Sample Reduction for Physiological Data Using Principal Component Analysis in Artificial Neural Network

Cid Mathew Adolfo, EcE, MSc, IEEE, PGCe

School of Computing and Engineering
University of Gloucestershire

This dissertation is submitted for the degree of
Doctor of Philosophy

Park Campus, Cheltenham October 2024

I would like to dedicate this dissertation to God and my loving parents …

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 50,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 140 figures.

Cid Mathew Adolfo, EcE, MSc, IEEE, PGCe
October 2024

DOI: 10.46289/8MM26HP8

Acknowledgements

I would like to fully recognize and express my sincerest appreciation to the following who made this study possible:

To my advisers, Dr. Hassan Chizari, Dr. Thomas Win, and Dr. Salah Al-Majeed, for their untiring patience, guidance and support. Their immense knowledge and expert advice helped me improve my research writing knowledge and skills. Without them, I would not be able to accomplish this remarkable feat;

To my panel members, Professor Hoshang Kolivand and Dr. Abu Alam for their insightful comments and suggestions for the enhancement of my research study;

To Republic of the Philippines - Department of Science and Technology - Science Education Institute (DOST-SEI) for granting me the Foreign Graduate Scholarship Program, which greatly financed my studies;

To my family and friends, for the belief, love and support, and for being my source of strength and inspiration;

To Almighty God, for giving me the wisdom, courage and capabilities to be able to surpass the challenges that I faced and fulfils my goals. To Thy be the glory!

Abstract

In the realm of biomedical applications, the analysis of big data holds immense promise, particularly in aiding medical practitioners with crucial interpretations. However, ensuring the integrity of multi-dimensional datasets and achieving high classification accuracy are paramount challenges. The quality of data sources presents a significant threat, with issues like abnormal, random and biased sampling posing obstacles, especially in machine learning contexts. Addressing these challenges is essential, particularly in biomedical applications reliant on accurate classification and prediction, such as physiological signal analysis utilizing Artificial Neural Networks (ANNs). This study proposes a novel approach, utilizing Principal Component Analysis–Sample Reduction Process (PCA-SRP), to preprocess datasets and enhance ANN model accuracy. We discuss the theoretical underpinnings of this methodology, followed by empirical validation using publicly available physiological and L/R Motor Movement (MM)EEG datasets. The analysis demonstrates the efficacy of PCA-SRP in cleansing datasets and improving ANN classification performance, with significant enhancements observed across various performance metrics. Notably, our approach achieves up to a 7% increase in accuracy in classifying L/R motor movement EEG signals, as validated through Python implementation.

Table of contents

List of figures **xv**

List of tables **xxi**

1 Introduction **1**
1.1 Introduction 1
1.2 Motivation 4
1.3 Problem Statements 5
1.4 Dissertation Aim and Objectives 6
1.5 Contributions of the Dissertation 7
1.6 Dissertation Structure 7

2 Literature Review and Prior Work **11**
2.1 Data-Cleaning Industries Perception 11
2.2 Taxonomy of Dirty Data 12
2.2.1 Müller and Freytag's Data Anomalies 12
2.2.2 Rahm and Do's Taxonomy of Data Quality Problems 13
2.2.3 Kim et al.'s Taxonomy of Dirty Data 14
2.3 Big Data Cleansing Techniques 16
2.4 The Concept of Data Minimization in the context of Big Data 20
2.5 Data Cleansing and Machine Learning in Physiological Data Analysis . . . 21
2.6 Mathematical Modelling 22
2.7 Principal Component Analysis — Dimension Reduction in Artificial Neural Networks (ANNs) 25
2.8 Recent Advancement in Principal Component Analysis 27
2.9 Findings 29

3 Methodology **31**
3.1 PCA-SRP and ANN Methodology 32

3.2 Principal Component Analysis—Sample Reduction Process (PCA-SRP) . . 35
3.2.1 Software Tool Used 39
3.3 Selectivity (Sc) 41
3.3.1 Test Selectivity (Sc_{test} and Recommended Selectivity (Sc) 42
3.4 Cross Validation 45
3.5 Classification Model Evaluation Metrics 46
3.6 Bootstrapping in Artificial Neural Networks (ANNs) 55
3.7 Effect of Model Accuracy by Injected Additional Random Samples 57
3.8 Conclusions and Contributions 57

4 Principal Component Analysis - Sample Reduction Process (PCA-SRP) in Physiological Datasets 59
4.0.1 PCA-SRP Implementation in ANN 59
4.1 PCA-SRP and ANN Methodology 60
4.2 Multidimensional Physiological Datasets 61
4.2.1 Heart Disease Dataset 64
4.2.2 Gender Voice Recognition Dataset 65
4.2.3 Breast Cancer Classification Dataset 67
4.2.4 Lung Cancer Patients Dataset 68
4.3 PCA - SRP Results in Physiological Datasets 70
4.3.1 Sensitivity vs Specificity Testing 70
4.3.2 Receiver Operating Characteristic (ROC) Curve Testing 72
4.3.3 PCA-SRP + ANN Comparison Accuracy Testing 72
4.3.4 Model Accuracy vs Injected Additional Random Samples Testing . 76
4.4 Contributions and Conclusions 76

5 Introduction and Previous Work on EEG Signal Acquisition and Analysis 79
5.1 Brain - Computer Interface (BCI) 80
5.1.1 10-20 System 82
5.1.2 EEG Frequency Bands 83
5.1.3 EEG Signal Analysis 85
5.2 Power Spectrum Distribution (PSD) 87
5.2.1 Short-Time Fourier Transform (STFT) Analysis - Amplitude vs Frequency 88
5.2.2 Simpson's Estimation Method and Welch's PSD Calculation 90
5.3 EEG Applications, Tools, and Libraries 92
5.4 EEG -BCI Current Challenges 95

5.5 Summary 97

6 PCA-SRP Implementation on Physionet's EEG Motor Movement (MM) Dataset 99
6.0.1 EEG Motor Movement (MM) Dataset 99
6.1 Experimental Framework 102
6.1.1 EEG Datasets Comparative Visualization 104
6.2 Discussion and Result 106
6.2.1 Brain's Electro-physiological Behaviour 106
6.2.2 PCA - SRP in L/R Motor Movement (MM) 116
6.2.3 Artificial Neural Networks (ANNs) Bootstrapping 117
6.3 Conclusion 130

7 PCA-SRP in Physionet's EEG L/R Motor Movement (MM/I) with P300 Oddball Random Auditory Dataset 133
7.1 Experimental Framework 133
7.1.1 EEG L/R Motor Movement and Imagery (MM/I) Dataset 133
7.1.2 P300 Oddball Paradigm EEG Dataset 134
7.1.3 Experimental Procedure 139
7.2 Result and Discussion 139
7.2.1 Evaluation by Visualization 140
7.2.2 Confusion Matrix Table -TP, FP, FN, and TN Evaluation 142
7.2.3 Receiver Operating Characteristic - Area Under The Curve (ROC-AUC) 152
7.3 Conclusion 153

8 Conclusion and Future Work 161
8.1 Contribution 161
8.2 Discussion and Conclusions 162
8.3 Future Work and Limitations 163
8.4 Published Paper 164

References 165

Appendix A Python Implementation 177
A.1 Cross-Validation Split 182
A.2 Simpson's Estimation Method and Welch PSD Calculation 191

Appendix B Classification Performance Metrics Graph 203

Appendix C PCA and ANN Supplementing Mathematical Discussion 217
C.1 Mathematics of Principal Component Analysis (PCA) 217
C.1.1 Standard Deviation (σ or SD) 218
C.1.2 Variance ($var(X)$) and Covariance ($cov(xy)$) 218
C.1.3 Correlation (ρ) 219
C.1.4 Standardization ($\dot{X}$) 219
C.1.5 Determinant and Generalized Variance 220
C.2 Ideas about Principal Component Analysis 220
C.2.1 Eigenvalue (λ) and Eigenvector (v) 221
C.2.2 Spectral Theorem 222
C.3 PCA Algorithm 222
C.3.1 Procedure in Performing PCA - Example 224
C.3.2 Explained Variance Ratio and F-Distribution 227
C.3.3 PCA Typical Python Implementation 230
C.4 Artificial Neural Network (ANN). 233
C.4.1 Mathematical Derivation of Artificial Neural Network 234
C.4.2 Feed Forward Neural Network 235
C.4.3 Back Propagation 238
C.4.4 Artificial Neural Network - Python Implementation 240

Appendix D Relevant Tables and Figures 245
D.0.1 Performance Metric Table Summary 246
D.0.2 Performance Metric Details 247
D.0.3 MM dataset - PCA-SRP Summary (Sc = PC_1 , n = 4927) 250

List of figures

1.1 Artificial Intelligence Popularity 2
1.2 Dirty data. 5
1.3 Thesis Structure. 9

2.1 Mathematical Modelling. 22
2.2 Data Modelling. 26
2.3 Data Cleaning - Thought Process Paradigm 29

3.1 Proposed Mathematical Model for Data Cleaning of Artificial Neural Networks. 32
3.2 Procedural Conceptual Framework 32
3.3 Typical Loss vs Learning Rate (LR) Behaviour. 34
3.4 Data Cleaning - Dataset - Evaluation 34
3.5 Covariance Interpretations. 36
3.6 PCA 2D Data Reconstruction. 38
3.7 PCA and Eigenstructures. 39
3.8 Python Programming Language and some supported libraries. 40
3.9 Sc—accuracy testing representation. 43
3.10 Principal Component Representation in Loading Score. 43
3.11 Frequency Distribution Diagram 44
3.12 Cross Validation Approach. 45
3.13 Train - Test Dataset. 46
3.14 Confusion Matrix 47
3.15 Illustration of Confusion Matrix. 48
3.16 Detailed Sensitivity vs specificity testing procedure. 48
3.17 ROC curve. 53
3.18 ROC curve interpretation. 53
3.19 ANN Bootstrapping 56
3.20 Bootstrapping - Histogram 56

3.21 Injected Additional Random Samples 57

4.1 Procedural Conceptual Framework 60
4.2 Typical Loss vs Learning Rate (LR) Behaviour. 62
4.3 Pre-processing an application of multidimensional datasets in the proposed approach. 63
4.4 Scree plot of the heart disease dataset. 66
4.5 Scree plot of gender voice recognition dataset. 66
4.6 Scree plot of breast cancer classification. 67
4.7 Scree plot of cancer patients dataset. 69
4.8 Sensitivity vs specificity diagram. 71
4.9 Receiver operating characteristic (ROC) curve. 73
4.10 Validation model accuracy of given datasets. 75
4.11 ANN accuracies in additional (+)%Randomness Response 77

5.1 Branches of BCI. 81
5.2 10-20 System. 82
5.3 Brain Lobes Anatomy. 84
5.4 EEG Signal - Spectral Waveform 84
5.5 Time Epoch or Sweeping 86
5.6 Power Spectrum Distribution (PSD). 88
5.7 Extraction and Filtering of EEG Signal Bands 89
5.8 Euler's Diagram 90
5.9 Amplitude vs Frequency Domain of a signal. 91
5.10 EEG Applications, Tools, and Libraries and its Timeline 93
5.11 Emotiv spokesperson conversation. 97

6.1 EEG Standard 64 Channel Electrodes Positioning Notation. 100
6.2 MM/I Experimental Dataset. 102
6.3 Experimental Structured Framework 103
6.4 MM - EEG Signal Visualisation 103
6.5 Subject 40 - Left-Right Side Motor Movement: Signal Channel Plot 106
6.6 Subject 40 - Left-Right Side Motor Movement: Event-ID Plot 106
6.7 Subject 40 - Left-Right Side Motor Movement: PSD Plot 107
6.8 Subject 40: Region of Interest (ROI)- Signal Intensity Plot 108
6.9 Subject 40- Left-Right: Epoch-Time Signal Intensity at Channel F7 (29) and F8 (37) 109
6.10 Subject 40 - Left-Right: Average Signal Intensity Plot 110

6.11 Subject 40 - Left-Right: Channel Signal Intensity Plot 110
6.12 Subject 40: Cranial Topography and Butterfly Signal Intensity Plot 111
6.13 Subject 40: Left - Right: Channels Signal Plot 112
6.14 Subject 83: Cranial Topography and Butterfly Signal Intensity Plot 114
6.15 Subject 109: Cranial Topography and Butterfly Signal Intensity Plot 115
6.16 MM Dataset Scatter Plot 116
6.17 L/R MM Dataset Continuous Parallel Plot 118
6.18 Selectivity (Sc) and Number of samples Removed (%) 118
6.19 MM Dataset - EEG Channels: ANN Model Validation Accuracy Increased . 120
6.20 Bootstrapping - Accuracy: Channel 1 to 8 122
6.21 Bootstrapping - Accuracy: Channel 9 to 16 123
6.22 Bootstrapping - Accuracy: Channel 17 to 24 124
6.23 Bootstrapping - Accuracy: Channel 25 to 32 125
6.24 Bootstrapping - Accuracy: Channel 33 to 40 126
6.25 Bootstrapping - Accuracy: Channel 41 to 48 127
6.26 Bootstrapping - Accuracy: Channel 49 to 56 128
6.27 Bootstrapping - Accuracy: Channel 57 to 64 129
6.28 Cranial Topography - ANN Bootstrapping Result 130

7.1 P300 Oddball Experimental Setup. 135
7.2 P300 Event-Related Potentials. 136
7.3 Signal Waveform: L/R Motor Movement (MM) and Oddball EEG Signals . 137
7.4 Power Spectral Density (PSD): L/R Motor Movement (MM) and Oddball EEG Signals 138
7.7 FC_5: Scatter Plot with Different Selectivity (*Sc*). 142
7.11 "Rightness" Evaluations - PPV and TPR (Sc = PC_1) 145
7.12 "Wrongness" Evaluations - FPR and FDR (Sc = PC_1) 146
7.13 F-Score Rate (F1) vs increasing Randomness (Sc = PC_1) 147
7.14 Geometric Mean (Gmean) Rate vs Selectivity (Sc) 148
7.15 Geometric Mean (Gmean) Rate vs increasing Randomness (Sc = PC_1) . . . 149
7.16 Cohen's Kappa Statistic Rate (khat) vs increasing Randomness (Sc = PC_1) . 150
7.17 Reciever Operation Characteristic - Area Under the Curve (ROC-AUC) Rate (Sc = PC_1) 152
7.19 Additional Randomness Threshold by Metric Evaluations 154
7.5 Event ID Diagram : L/R Motor Movement (MM) and Oddball EEG Signals 155
7.6 Experimental Methodological Procedure. 155
7.8 EEG Datasets - Parallel Coordinate Plot with Different Selectivity (*Sc*). . . 156

7.9 EEG Datasets - Scatter Matrix Plot with Different Selectivity (*Sc*) 157
7.10 Accuracy Rate vs increasing Randomness (Sc = PC_1) 158
7.18 Selectivity Threshold by Metric Evaluations 159

B.1 Accuracy vs increasing Randomness (Sc = PC_1) 204
B.2 Error Rate vs increasing Randomness (Sc = PC_1) 204
B.3 PPV vs increasing Randomness (Sc = PC_1) 205
B.4 TPR vs increasing Randomness (Sc = PC_1) 205
B.5 True Negative Rate (TNR) vs increasing randomness 206
B.6 Balanced Accuracy (BA) Rate vs increasing Randomness (Sc = PC_1) 206
B.7 F-score vs increasing Randomness (Sc = PC_1) 207
B.8 G-mean vs increasing Randomness (Sc = PC_1) 207
B.9 Cohen's Kappa Statistics vs increasing Randomness (Sc = PC_1) 208
B.10 Matthew's Correlation Coefficient (MCC) vs increasing Randomness (Sc = PC_1) 208
B.11 Fowles - Mallows (FM) Index vs increasing Randomness (Sc = PC_1) 209
B.12 Bookmaker (BM) Informedness Rate vs increasing Randomness (Sc = PC_1) 209
B.13 Positive Likelihood Ratio vs increasing Randomness (Sc = PC_1) 210
B.14 Negative Likelihood Ratio vs increasing Randomness (Sc = PC_1) 210
B.15 Negative Predictive Value (NPV) vs increasing randomness 211
B.16 False Positive Rate (FPR) vs increasing randomness 211
B.17 False Negative Rate (FNR) vs increasing Randomness (Sc = PC_1) 212
B.18 False Detection Rate (FDR) vs increasing Randomness (Sc = PC_1) 212
B.19 False Ommission Rate (FOR) vs increasing Randomness (Sc = PC_1) 213
B.20 Prevalence (Prev) Rate vs increasing Randomness (Sc = PC_1) 213
B.21 Prevalence Threshold (PrevT) Rate vs increasing Randomness (Sc = PC_1) . 214
B.22 Critical Success Rate (CSI) Rate vs increasing Randomness (Sc = PC_1) . . 214
B.23 Markedness (MK) Rate vs increasing Randomness (Sc = PC_1) 215
B.24 AUC-ROC vs increasing Randomness (Sc = PC_1) 215

C.1 PCA in 3-Dimensional Space 223
C.2 Frequency Distribution Diagram 229
C.3 Example output of the PCA Implemenattion 231
C.4 Example Explained Variance Ratio 231
C.5 Example Scree Plot 232
C.6 Example PCA 2D Space Graph 232
C.7 Artificial Neural Network 234

C.8 One Layer Neural Network. 235
C.9 Artificial Neural Networks (ANNs) Activation Functions 237
C.10 Deriving the Total error 239

List of tables

2.1 Müller and Freytag's Dirty Data Taxonomy 13
2.2 Rahm and Do's Taxonomy of Data Quality Problem 14
2.3 Kim et al.'s taxonomy of dirty data 15
2.4 Data-cleansing methods for big data. (Not Exhaustive List) 18
2.5 Currently used data-cleaning technique. 19
2.6 Summary of Mathematical Model 24

3.1 PCA-SRP and ANN parameters. 33
3.2 Interpretation of Kappa value 52

4.1 PCA-SRP and ANN parameters. 61
4.2 Datasets used and their metadata. 61
4.3 Sorted loading scores of heart disease dataset. 64
4.4 Sorted loading scores of gender voice recognition dataset. 66
4.5 Sorted loading scores of breast cancer classification dataset. 67
4.6 Sorted loading scores of cancer patients dataset. 69
4.7 Datasets sample status after PCA—sample reduction process. 70
4.8 Sensitivity and Specificity Result. 72
4.9 Area under the Dataset's Curve (AUC). 74
4.10 Dataset noise sample accuracy. 74
4.11 Dataset random sample accuracy. 74

5.1 Brain's Anatomy - Lobes and its Function 83
5.2 EEG Frequency Bands and its corresponding Activities 83

6.1 ANN Model Validation Accuracy Increase ($PCA_{SRP} + ANN$ vs ANN only) . 119

7.1 FC_5: Sorted Loading Score of MM/I and P300 Oddball (Random) Datasets 140
7.2 Confusion Matrix Table Vs Increasing Sc (with 10% Randomness - 435 samples) 143

7.3 MM/I and P300 Oddball Random Datasets - Performance Metrics Table Summary 151

C.1 Covariance vs Correlation. 219
C.2 ANN parameters. 241

D.1 Performance Metrics Table 246
D.2 Performance Metrics Summary 247
D.3 MM dataset - PCA-SRP Summary (Sc = PC_1 , n = 4927) 250

Chapter 1

Introduction

1.1 Introduction

A data explosion is currently underway, with projections indicating a fivefold increase to 160 ZB over the next five years. By 2025, we anticipate generating 130 ZB more data than in 2020, surpassing the cumulative data produced until 2019 [1]. This data surge would pose significant challenges if not addressed effectively.

The growth in data is driven by various developments, both commercially and domestically. Numerous industries, including biomedical healthcare, education, government, finance, and technology, are experiencing a massive increase in data volume, velocity, and variety. Social media platforms like Facebook, Instagram, and Twitter, alongside platforms like Google Cloud Public Datasets and Kaggle, are significant contributors to this data proliferation.

Efforts to manage data are concentrated at its source due to its widespread generation. Many social media companies and collaborative platforms, as well as government entities and institutions, share vast amounts of public and personal data online.

However, the integrity of data poses challenges. Acquisition and movement of data require substantial resources, and data quality issues such as noise, inaccuracy, and bias are prevalent. Like oil, data requires thorough refinement before utilization, as highlighted by the phrase "Data is the new oil" [2].

This research leans more towards the interpretivist paradigm, emphasizing descriptive and non-numeric methodologies [3]. It discusses technical challenges in maintaining extensive information collection, cataloguing, and reporting projects, leveraging advancements in computer and information science to address data-related issues.

Recent years have witnessed significant popularity in machine learning, particularly in deep learning and neural networks [4]. This approach, inspired by the functionality of the

human brain, employs both empirical and a priori methodologies to observe and analyze complex systems.

In conclusion, while the data explosion presents opportunities, addressing data-related challenges requires a multidisciplinary approach, incorporating technological advancements and rigorous methodologies.

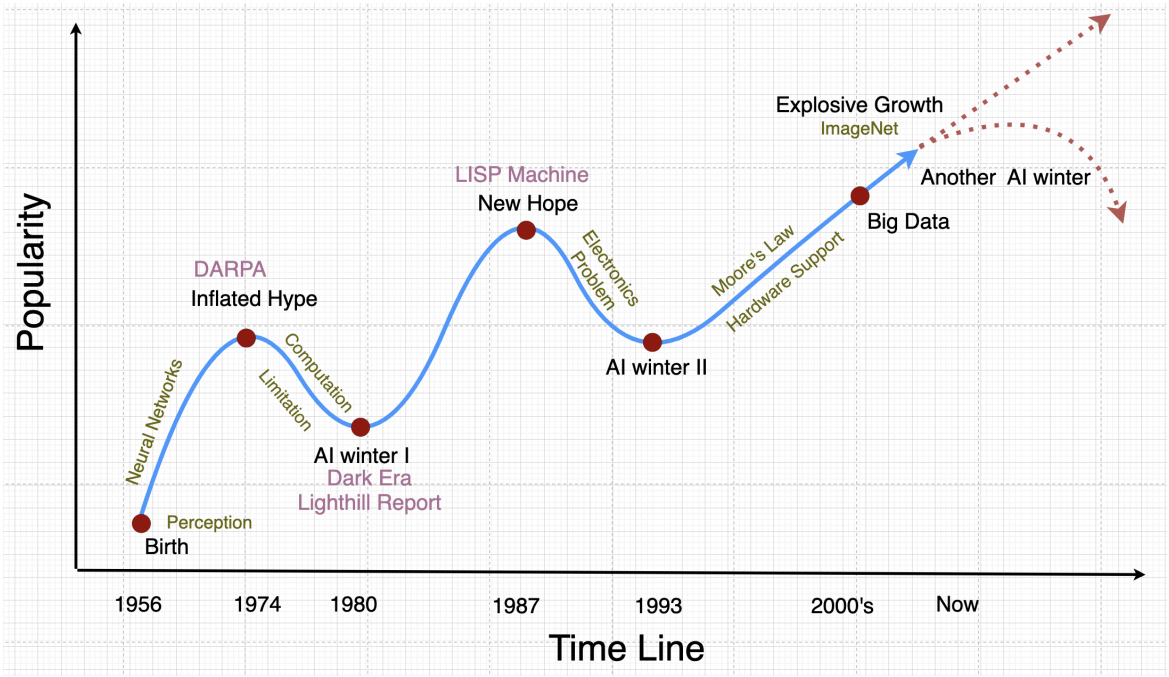


Fig. 1.1 Artificial Intelligence Popularity

Since the 1940s, as depicted in Figure 1.1, Artificial Neural Networks (ANNs) have improved tremendously, with notable milestones including [5]:

- The perception of Minsky and Papert, pioneers of the Neural Network concept.
- The NYTimes video 'Thinking Machine', which popularized the idea and gained mass acceptance.
- The US Government's DARPA (Defense Advanced Research Projects Agency) Initiatives played a central role in launching the Information Revolution.
- The US ALPAC and UK Lighthill Report, which marked the onset of the first AI winter in the 1960s. These reports failed to fulfill the promises of AI, leading to a drastic decline in research funding.

- Hopfield's paper in the 1980s, titled "Neural networks and physical systems with emergent collective computational abilities," which laid the foundation for collective computational properties in biological structures.
- John McCarthy's invention of the LISP machine introduced the first AI programming languages.
- Insufficient advancement in electronics hardware to support AI calculation led to the second AI winter in the 1990s.

During this period, computational challenges were often addressed through backpropagation, the primary learning technique for artificial neural networks, sparking renewed interest in the field. Researchers became more cautious following the lessons learned from the AI winters, leading to a significant conservative shift in the late 1980s and early 1990s towards established mathematical and statistics-based approaches. As a result, terms like "informatics" or "analytics" were preferred over "AI" [6]. Thus, advancements in mathematics paved the way for AI's complex computation.

Another milestone was the development of electronic hardware supporting complex computations, driven by Moore's Law, which led to the widespread use of computers across various industries, marking the beginning of the Era of Big Data. This era witnessed the emergence of Autonomous Generation, combining technologies to automate various human tasks, facilitating the gathering and acquisition of data.

Additionally, the conservative movement in AI led to the creation of public benchmark datasets and related contests, serving as crucial tools for evaluating developments in AI research. Notable benchmark datasets include:

- The Letter Dataset [7]
- Yale Face Database [8]
- MNIST dataset [9]
- Spambase Dataset [10]
- ISOLET Dataset [11]
- TIMIT [12]
- JARtool experiment Dataset [13]
- Solar Flare Dataset [14]
- EEG Database [15]
- BCI EEG Signal - Motor/Imagery [16]
- Breast Cancer Wisconsin (Diagnostic) Dataset [17]
- Liver Disorders Dataset [18]
- Thyroid Disease Dataset [19]

- Abalone Dataset [20]
- UCI Mushroom Dataset [21]
- Heart Disease Dataset [22]
- Gender Voice Recognition Dataset [23]
- Breast Cancer Classification Dataset [24]
- Lung Cancer Patients Dataset [25]

Furthermore, Fei Fei Li's promotion and development of large-scale databases like ImageNet have enabled computers to process big data through machine learning [26]. However, the excitement surrounding the attainment of human-level intelligence has led to exaggerations and media attention uncommon in other technological fields. Several crucial considerations are necessary to prevent another AI winter, including addressing issues related to "dirty data," which encompasses low-quality data that may compromise research outcomes [27]. AI still faces significant obstacles on the path to achieving Artificial General Intelligence, necessitating solutions to address problems with significantly fewer data.

1.2 Motivation

Given the earlier-mentioned issues, providing researchers and non-professionals with a common ground for comprehending the theoretical aspects of enhancing data-cleaning capabilities in physiological datasets would be beneficial. This involves providing a method for subject matter experts, such as data analysts, to derive meaningful information and data in a manner conducive to data analysis and classification.

On the computing side, neural networks are often described as black boxes, as their model structure and parameters are unknown, making it challenging to understand the rationale behind certain decisions.

However, progress is being made in this research study, particularly in the selection of training sets for Artificial Neural Networks (ANNs), which would benefit data analysts in the fields of data cleansing and big data analytics if knowledge representation could align more closely with advancements in data cleaning. Improved data cleansing has various practical advantages, including identifying different classes of disorders, potentially improving data quality, enhancing the performance of artificial neural networks to aid medical professionals in interpreting patients' medical results, and contributing to the accuracy of physiological data acquisition.

1.3 Problem Statements

Understanding data requires understanding its composition, which comprises valid and dirty data. While accurate data contain reliable information with predictive power, dirty data [28] contain misleading information, including noisy or random and erroneous data, such as pragmatic contexts and semantic- and syntactic-biased errors shown in Figure 1.2.

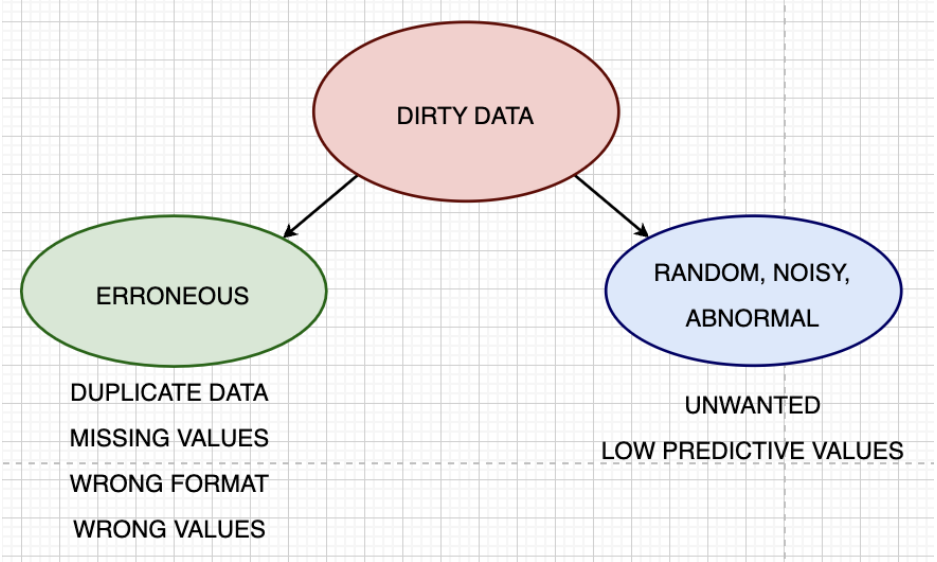


Fig. 1.2 Dirty data.

Dirty and Imprecise Data. A collection of facts, including numbers, words, measurements, observations, or descriptions, always contains erroneous data. It may also refer to data that lacks coherence or is randomly generated. Data can be considered dirty and imprecise if it includes:

- Misleading item data
- Duplicate data
- Incorrect data
- Inaccurate data
- Non-integrated data
- Data without standardized formatting
- Incorrectly punctuated or spelled data

Noisy data are small fluctuations considered unwanted, while **random (abnormal) data** are disruptive, seemingly "good" data without predictive information that hinders generalization across the dataset.

Data cleansing involves identifying and correcting errors in the data before modeling. Common techniques include removing duplicates, irrelevant data, standardizing capitaliza-

tion, converting data types, clearing formatting, fixing errors, searching for missing values, and translating language.

Various industrial facts about data cleansing [29] [30] include:

- Over 25% of revenue reported being subjected to data quality issues.
- Data analysts spend over 50% of their time on data quality.
- It takes data analysts around 4 hours to detect a data quality issue.
- Data cleansing and organization occupy 60% of data scientists' work.
- Data scientists spend approximately 80% of their time organizing and preparing data for analysis, with data collection occupying the remaining 19%.
- Data preparation is viewed as the least enjoyable aspect of a data scientist's job by 76% of respondents.
- Obtaining data sets is considered the least enjoyable aspect of a data scientist's job, while 57% find cleaning and organizing data to be the least pleasurable.

In recent decades, flexible and wearable body sensors have been developed for medical applications, showing significant potential for future biomedical and healthcare applications. However, data accuracy, sensitivity, selectivity, and data preparation play critical roles in fulfilling clinical assessment needs or commercialization requirements in medical facilities. Therefore, data cleansing techniques need improvement to address random data acquired via biomedical sensors and human input errors.

While most data cleansing techniques help improve and minimize errors in gathered data, they often fail to measure the predictive value of samples. To accurately analyze data and determine predictive power, datasets need to be free from noise and random data, which significantly impact classification, regression, and prediction tasks in Machine Learning. Technological solutions [31] have highlighted relevant control algorithms statistically and mathematically to address these challenges.

1.4 Dissertation Aim and Objectives

The dissertation aims to propose a mathematical model for handling general and raw signals in Big Data, particularly in physiological datasets and motor movement EEG signals.

1. Identify the key components of multi-dimensional data and explore mathematical equations to determine the magnitude of their predictive value.
2. Examine the proposed mathematical model using different classification metrics.
3. Evaluate the effectiveness of the proposed mathematical model in basic artificial neural networks through model accuracy comparison, bootstrapping, and resilience testing.

1.5 Contributions of the Dissertation

This dissertation focuses on reinforcing the ANN methodology using the Principal Component Analysis - Sample Reduction Process (PCA-SRP) proposed herein to determine system accuracy and performance with varying sizes of multidimensional biomedical datasets. Furthermore, it investigates heart disease, voice and speech analysis for gender recognition, breast cancer classification, cancer patient datasets, and EEG signals for BCI application to demonstrate the versatility and flexibility of the proposed PCA-SRP data-cleansing technique.

The following aspects of the dissertation are believed to be original contributions:

1. The presentation of a mathematical model for data cleaning in Chapter 2.6, as depicted in Figure 2.1.

2. The novel use of Principal Component Analysis as Sample Reduction Process (PCA-SRP) in data cleaning for Artificial Neural Network classification problems in physiological data analysis, discussed in Chapter 3.2, including the identification of the Selectivity Threshold for highly predictive samples.

3. The application of the proposed mathematical model using PCA-SRP in Physionet's EEG Motor Movement (MM) Dataset as a data-cleaning process, presented in Chapter 5.

4. The bootstrapping process of ANN using the proposed mathematical model with PCA-SRP in Physionet's EEG Motor Movement (MM) dataset as a data-cleansing technique, demonstrating a significant increase in model accuracy, as shown in Chapter 6.2.3.

5. The classification and identification of randomness and minority EEG samples in the P300 Oddball Random Auditory Dataset in Physionet's EEG Motor Movement (MM), utilizing the proposed mathematical model with PCA-SRP as the data-cleaning process, detailed in Chapter 6.

1.6 Dissertation Structure

The dissertation consists of eight chapters, covering eight research topics, excluding the Introduction, Conclusion, and Future Works chapters. These chapters are:

- Chapter 2: **Literature Review and Prior Works**, discussing usability-related factors investigated through an extensive systematic review of existing literature on Data Cleaning, Taxonomy of Dirty Data, Machine Learning, and Mathematical Modelling.

- Chapter 3: **Methodology**, explaining the data cleansing method, theoretical mathematics behind the proposed process, identification of the Selectivity Threshold, bootstrapping, performance metrics, limitations, and interpretations.
- Chapter **??**: **Principal Component Analysis - Sample Reduction Process (PCA-SRP) in Physiological Dataset**, covering attributes of physiological datasets, performance metrics, and results of PCA-SRP in various datasets.
- Chapter 4: **Introduction and Previous Work on EEG Signal Acquisition and Analysis**, discussing RAW signals, Feature Extraction and Classification, EEG applications, tools, libraries, and current EEG-BCI challenges.
- Chapter 5: **PCA-SRP Implementation on Physionet's L/R EEG Motor Movement (MM) Dataset**, focusing on identifying anomalies in EEG signals, model accuracy, and the impact of PCA-SRP on system performance.
- Chapter 6: **PCA-SRP in Physionet's L/R Motor Movement (MM) EEG Signals with P300 Oddball Random Auditory Dataset**, investigating randomness and minority samples in EEG datasets using PCA-SRP and analyzing performance metrics.

Figure 1.3 illustrates the thesis structure, with each chapter addressing related research questions, and bold blocks indicating chapters containing original contributions.

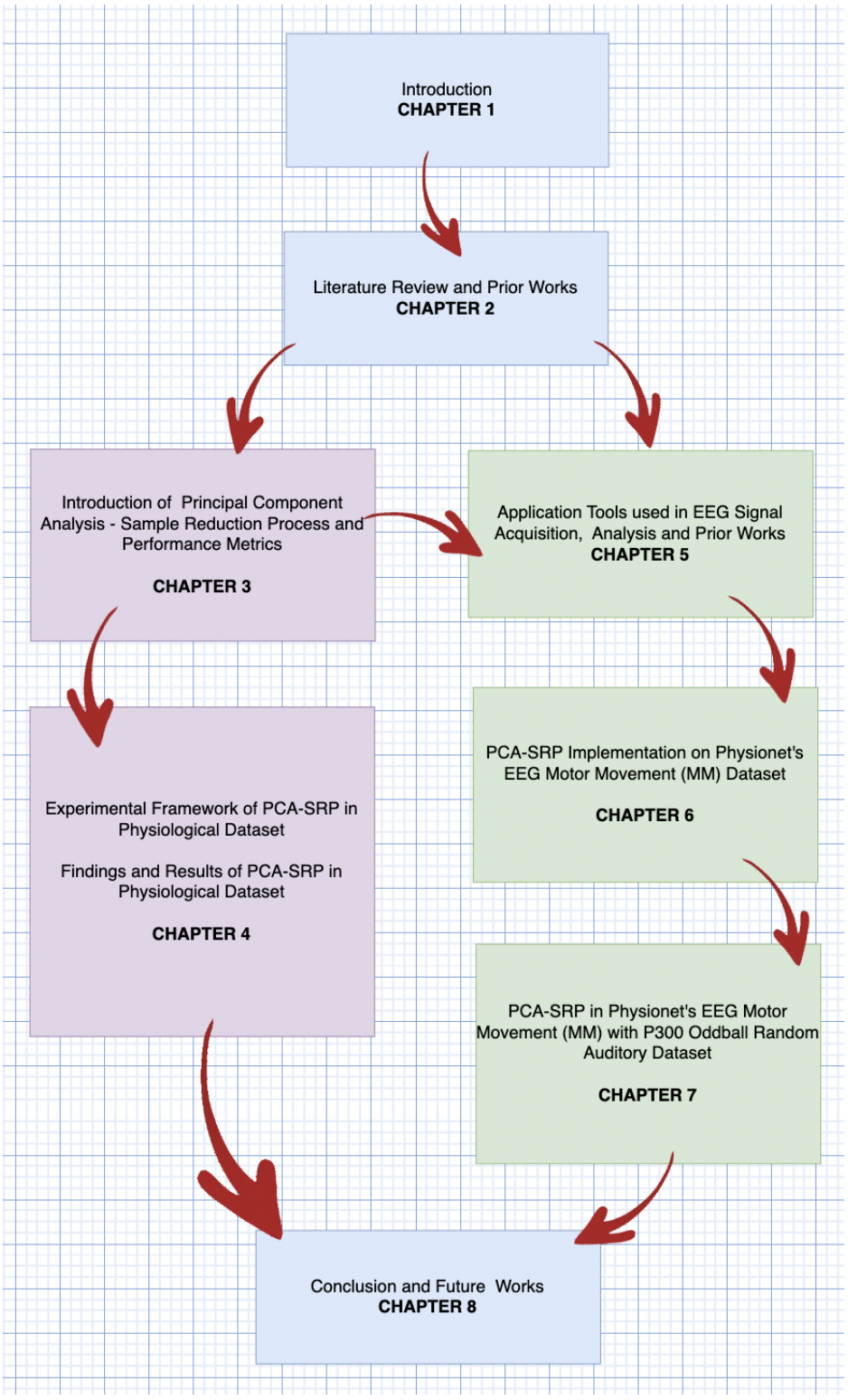


Fig. 1.3 Thesis Structure.

Chapter 2

Literature Review and Prior Work

This chapter reviews the industry perspective on data cleaning, existing taxonomies of dirty data types from the literature, and data cleaning methodologies and approaches. This provides the foundation for developing the proposed mathematical modeling for the data-cleaning process reviewed in the final part of this chapter.

2.1 Data-Cleaning Industries Perception

The volume of data collected nowadays is vastly increasing, and since most data acquired are polluted, the dependability of the data is declining. Various data-cleaning methodologies are available to rectify this issue, but data cleansing still needs to be improved when working with large data requirements. Data cleaning, also known as data cleansing, is no longer a recent area of research. It aims to increase data quality by detecting and eliminating errors and inconsistencies [32]. Meanwhile, due to the need for more confidence in collected data, end users who rely on it must spend more time confirming its validity, further reducing speed and productivity. When the number of dirty data rises, adding more manual procedures causes an increase in errors, biased and discrepancies.

Data cleansing is a crucial part of the overall data management process and one of the core components of data preparation work that readies data sets for use in business intelligence (BI) and data science applications [33]. Furthermore, data cleansing is a critical step in any data analysis project. It involves identifying and correcting errors in the data prior to modelling, including, but not limited to, outliers, missing values, and much more. Data cleansing improves data quality and helps provide more accurate, consistent, and reliable information for decision-making in an organisation. Basic data cleaning operations should always be performed on a dataset before jumping to more sophisticated methods.

2.2 Taxonomy of Dirty Data

In practice, detecting and cleaning all the dirty data in all data sources is quite expensive and unrealistic. The cost of cleaning dirty data needs to be considered for most enterprises. This problem needs to attract more attention from researchers [34]. In order to understand the data cleaning process, one needs a comprehension of the classification and taxonomy of dirty data - to apprehend the existing framework for understanding how dirty data arise, manifest themselves, and may be cleansed to ensure proper construction of data warehouses and accurate data analysis.

Here are the current classifications or taxonomies of dirty data:

2.2.1 Müller and Freytag's Data Anomalies

Müller and Freytag [35] described dirty data as anomalies that define a property of data values that renders them a wrong representation. In the pre-defined data model prescribed by the authors, data acquired that does not conform to the constraints of the data model is considered a data anomaly and classified broadly data anomalies into three groups as follows:

- **Syntactical Anomalies** consider dirty data from data's representation perspective. There are three dirty data types: lexical errors, domain format errors and irregularities. Lexical errors highlight the structural differences between the data items and the specified format. Domain format errors indicate that the attribute value for an attribute does not follow the expected domain format. The problem of non-uniform use of values, units and abbreviations deals with irregularities.
- **Semantic Anomalies** are data that violate the integrity requirements and duplicate data. Integrity constraints are used to specify the rules for representing knowledge about the domain and the values allowed for representing certain facts. Duplicate data here represents two or more tuples representing the same entity.
- **Coverage Anomalies** describe the dirty data due to missing values or tuples.

Table 2.1 shows the dirty data types classified in this work.

Table 2.1 Müller and Freytag's Dirty Data Taxonomy

Dirty Data Type
Lexical error
Domain format error
Irregularities error
Duplicate records
Missing values
Missing tuple
Invalid tuple

2.2.2 Rahm and Do's Taxonomy of Data Quality Problems

"Dirty Data" has been replaced by "Data Quality Problem" by the authors in their defence [36]; database systems enforce restrictions of the specific data model and application integrity constraints. It was observed that data quality problems are divided into two categories – single-source problems and multi-source problems, based on the plurality of their sources.

It mentioned the existence of overlapping data that causes the problem of duplicate records within the multi-data sources and contradicting records among multiple data sources. The "Data Quality Problem" Taxonomy, the authors introduced is shown in Table 2.2.

Table 2.2 Rahm and Do's Taxonomy of Data Quality Problem

Dirty Data Type
Illegal values due to invalid domain range
Violated attribute dependences at schema level
Uniqueness violation
Referential integrity violation
Missing values (null allowed)
Cryptic values, Abbreviations
Misspellings
Embedded values
Misfielded values
Violated attribute dependences at the instance level
Word transpositions
Duplicated records in single data source
Contradicting records in single data source
Wrong references
Naming conflicts
Structural conflicts
Data conflicts in multiple data sources
Duplicate records in multiple data sources
Contradicting records in multiple data sources

2.2.3 Kim et al.'s Taxonomy of Dirty Data

The authors defined and categorised the dirty data as either missing data, wrong data, or non-standard representations [37], and put more distinction between different types of dirty data. As shown in Table 2.3, within the three (3) categories, fourteen (14) dirty data types have been added.

Classifying the given dirty data's taxonomies seemed good, noisy and random data samples were vague. However, this might identify in the following, but it is not quite clear and defined:

1. **Irregularities Error** of Müller and Freytag's Data Anomalies,
2. **Referential Integrity Violation** for Rahm and Do's Taxomony of Data Quality Problems

Table 2.3 Kim et al.'s taxonomy of dirty data

Category	Dirty Data Type
Missing Data	Missing data (null value allowed)
Not Missing, but Wrong Data	Use of wrong data type including value range
	Dangling data
	Violation of uniqueness constraint data
	Mutually inconsistent data
	Dirty data due to failure of transaction management facility
	Wrong categorical data
	Outdated temporal data
	Inconsistent spatial data
	Errorneous entry
	Misspelling
	Extraneous data
	Entry into wrong fields
	Wrong derived-field data from stored data
	Inconsistency across multiple tables/files due to integration constraint problem
Not Missing, Not Wrong, but Unusable Data	Different data for the same entity across multiple databases
	Ambiguous data due to use of abbreviation
	Ambiguous data due to incomplete context
	Different representation for non-compound data due to use of abbreviation
	Different representation for non-compound data due to use of Alias/ nickname
	Different representation for non-compound data due to encoding format
	Different representation for non-compound data due to different representations
	Different representation for non-compound data due to measurement units
	Different representation for compound data due to abbreviation
	Different representation for compound data due to use of special characters
	Different representation for compound data due to different ordering
	Different representation for hierarchical data due to abbreviation
	Different representation for hierarchical data due to use of special characters
	Different representation for hierarchical data due to different ordering

3. **Mutually Inconsistent Data** and **Inconsistent Spatial Data** for Kim et al's Taxonomy of Dirty Data.

Among the given dirty data taxonomies, the authors classify more erroneous data and barely hard to define the dirty data as seemingly good, noisy, and random data, which is mathematically and statistically disruptive and least predictive. Generally, **first thing to solve the problem, knowing the problem, in a sense, hard to classify the random data, and how much more to identify it, most likely hard to solve and remove it**.

2.3 Big Data Cleansing Techniques

Data cleaning in the realm of big data is a crucial process aimed at enhancing data quality by addressing the presence of dirty data. Dirty data, which is inevitable in big data environments, can significantly impact data quality [38]. Various techniques and frameworks have been proposed to tackle data cleaning challenges in big data. While specific field-oriented methods exist, there is a recognized need for more general approaches to data cleaning in the context of big data [39].

Currently, there are two classifications of data cleansing: **traditional data cleansing** and **data cleansing for big data**. Traditional data cleansing techniques are so-called because they are not used to manage massive volumes of data, such as Potter's Wheel and Intelliclean [40]. It is often assume that all data can be loaded into memory simultaneously, which is impractical for big data due to its scale [41]. As the volume of data in big data continues to grow rapidly, cleaning techniques face the challenge of scaling data capacities efficiently [42]. Sampling methods have been explored to address the multi-class imbalance issue in big data, with a focus on heuristic sampling methods and their impact on deep learning neural networks [43].

One of the perennial problems in data analytics is identifying and restoring dirty data, and failure to do so will result in faulty analytics and unreliable decisions. New abstractions and scalability are among the various facets of this issue and the concern when developing data-cleaning methods to cope with the amount and diversity of data [44–46]—see Table 2.5. Given the significant amount of data, it needs time to be processed to be suitable for Big Data Analysis and decision-making. The data's volume, veracity and velocity (3Vs) must also be considered when analysing the proposed approaches; however, the researcher [40] mentioned that **"Data analytics is not about having the information known, but about discovering the predictive power behind the data collected"**.

In the context of security big data ecosystems, data cleaning techniques involve

- **Missing Data Handling:** Big data often contains missing values due to various reasons such as sensor failures, human errors, or data corruption during transmission. Imputation techniques such as mean, median, mode imputation, or more advanced methods like k-nearest neighbors (KNN) or regression imputation can be applied to fill in missing values.

- **Outlier Detection and Treatment:** Outliers are data points that deviate significantly from the rest of the dataset and can skew analysis results. In big data, identifying outliers becomes more challenging due to the sheer volume of data. Statistical methods like z-score, interquartile range (IQR), or machine learning algorithms like Isolation Forest or Local Outlier Factor (LOF) can be employed to detect and handle outliers.

- **Noise Reduction:** Big data may contain noise introduced during data collection or transmission. Noise reduction techniques such as smoothing, filtering, or data transformation methods like Fourier Transform or Wavelet Transform can be applied to mitigate noise and extract meaningful patterns from the data.

- **Inconsistency Resolution:** Inconsistencies in big data can arise from data integration processes where data from multiple sources are combined. Cleaning inconsistencies involves identifying and resolving conflicts in data values, formats, or representations to ensure data consistency across the dataset.

- **Normalization and Standardization:** Data normalization and standardization techniques are applied to scale the features of the dataset to a uniform range, making it easier for machine learning algorithms to converge during model training. Normalization methods include Min-Max scaling, z-score normalization, or robust scaling, while standardization involves transforming data to have a mean of zero and a standard deviation of one.

- **Duplicate Detection and Removal:** Big data often contains duplicate records due to data entry errors or replication during data collection processes. Identifying and removing duplicates is essential to maintain data quality and prevent redundancy in analysis. Techniques such as hashing, record linkage, or similarity measures can be employed to detect and eliminate duplicates efficiently.

- **Data Type Conversion:** Big data may contain heterogeneous data types that need to be converted into a consistent format for analysis. Converting categorical variables into numerical representations using techniques like one-hot encoding or label encoding, or

parsing date and time data into a standardized format, are common data type conversion tasks in big data cleaning.

- **Data Validation:** Data validation involves ensuring that the data adheres to predefined quality constraints or business rules. Validation checks may include range validation, format validation, or referential integrity checks to identify and correct erroneous data entries.
- **Data Deduplication:** In addition to duplicate detection and removal, data deduplication involves identifying and eliminating redundant information within the dataset. Deduplication techniques such as record linkage or entity resolution are used to merge duplicate records and create a single, clean representation of the data.
- **Text Preprocessing:** In big data analytics, textual data is abundant, but it often requires preprocessing to extract meaningful insights. Text cleaning techniques such as tokenization, stop word removal, stemming, or lemmatization are applied to prepare textual data for analysis tasks like sentiment analysis, text classification, or topic modeling.

This big data cleaning techniques are quantifying the user-level error that has been demonstrated through case studies[47], highlighting the importance of meticulous data cleaning practices in big data scenarios. Meanwhile, the big data techniques in Table 2.4 such as Cleanix [44], SCAREd [45], KATARA [48], and BigDansing [46] is explicitly developed for big data [49]. Regarding the emerging trends in data-cleaning techniques, one of the new challenges researchers face is scalability.

Table 2.4 Data-cleansing methods for big data. (Not Exhaustive List)

Methods	**Key Features**	**Approach**
Cleanix	Scalability, unification, and Usability	Rule Selection
SCAREd	Scalability	Machine Learning Technique
KATARA	Easy Specification, Pattern Validation Data Annotation	Knowledge-base and Crowdsourcing
BigDansing	Efficiency, Scalability and ease of use	Rule Specification

Cleanix, SCAREd, and BigDansing focus on the scalability issue in the data cleansing process. Moreover, SCAREd and BigDansing do not require any human-domain expert in the cleansing process. SCAREd needed an extensive set of rules to update the dataset; however, no expert was present. Nevertheless, the process is expensive. If the administrator fails to identify the correct fixes for the dirty dataset, it will result in the redundancy of training data and a threshold machine learning parameter that is hard to set precisely [48]. Furthermore, BigDansing also requires a set of data-quality rules for optimization of the cleansing process that requires too many regulations to calibrate and put into place before the start of the cleaning process, as shown in Table 2.5 below; however, it needs no human-domain expertise to monitor the whole process, although adjusting such parameters is crucial in maintaining the essence of the information in the datasets [40]. These data-cleansing techniques support various data-cleaning tasks such as abnormal value detection and correction, incomplete data filling, de-duplication, and conflict resolution (Cleanix), value modification (SCAREd), identification of correct and incorrect data and generating top-k possible repairs for inaccurate data (KATARA), and rules into a series of transformations that enable distributed computations and several optimizations (BigDansing).

Table 2.5 Currently used data-cleaning technique.

	Volume	**Veracity**	**Velocity**
Cleaning Technique Data	**Scalability**	**No need for Extensive Data Quality Rule Optimisation**	**No need for Human Domain Expert**
Cleanix	x		
SCAREd	x		x
KATARA		x	x
BigDansing	x		x

There are two kinds of dirty data: erroneous and noisy or random—see Figure 1.2. The current big-data-cleansing techniques—Cleanix SCAREd, KATARA [48], and BigDansing —are focused chiefly on erroneous dirty data such as a solution in duplicate entries, missing values, wrong values and wrong formats.

Cleanix, KATARA and BigDansing are not able to predict the significant sample values and cannot determine how to eliminate the mixed random data; however, the SCAREd technique, though constructed via machine learning, can only replace the missing values with

the most precise value, but not when dealing with random data and knowing the predictive power of the samples.

Overall, data cleaning in big data involves a range of methods and technologies, including machine learning paradigms, continuous data cleaning approaches, and interactive deterministic data cleaning processes [50] [51]. These techniques are essential for ensuring high-quality data, which is fundamental for effective decision-making and analysis in the big data domain. By leveraging advanced data cleaning methods, organizations can enhance the accuracy and reliability of their data analytics processes, leading to more informed insights and strategic outcomes.

2.4 The Concept of Data Minimization in the context of Big Data

Artificial intelligence (AI) and Machine Learning (ML) have gained popularity in the computer sector. It's a branch of AI, that lets robots learn from data and decide without explicit programming. Stated differently, through data analysis and pattern recognition, machine learning algorithms may be trained to learn and get better over time. Any machine-learning algorithm starts with training data. The system is trained to identify patterns and provide predictions using the data. If the training data is biased or incomplete, the machine learning algorithm will make inaccurate predictions. The correctness and dependability of the machine learning algorithm are directly impacted by the caliber of the training samples.

For a variety of purposes, it can easily gather an increasing amount of data in any manner imaginable. "The more, the merrier (better)", it is absolutely true, since we can at least obtain what we would require. But at a certain point, having too much data becomes equivalent to having none at all if there's no practical way to retrieve it as a notion **"More is less, and less is more"** comes into place [52]. Worse, as the data accumulated, more random data samples was injected into the system as well.

In potraying a different route, **Big Data** is a methodology rather than a particular selection of attribute in device. The objective is to find correlation of patterns that reveals its behaviour. Moreover, the characteristics of Big Data (5 Vs) i.e., volume (large quantity), veracity (real-time data), velocity (speed), variety (data fusion), and value (worth). it is more safe to define Big Data as "Mixed Data" [53]. For example, extracting the brain wave signal in the subject for particular mental action, but acquiring signals may include different mental action, that poses security threat and privacy of an individual. Furthermore, Articles 5, 25, 47, and 89 of the 2018 General Data Protection Regulation (GDPR) prominently include the idea of

data minimization . The idea itself is at the core of the regulation, serving as the regulatory framework's ethos and one of the seven fundamental data protection principles under EU data protection law [GDPR, Chapter 2, Article 5 (1) (c)]. The GDPR of 2018 states that personal data must be "adequate, relevant, and limited" to what is "necessary" for the "purposes" for which it is processed (data minimization) [54]. In actuality, the data reduction notion compels to be more deliberate about the acquired samples. Therefore, if the Big Data is mixed in nature, requires us to be more conscious about the collected data , and yet "the actual or quantifiable measurement of Big Data are still not yet known [55].

2.5 Data Cleansing and Machine Learning in Physiological Data Analysis

Given all the above difficulty data-cleaning techniques and more conscious about the legality as part of the discussion, removing the seemingly sound, noisy, random or samples (minority) not part of the the dataset is called **sample reduction**. It holds a significant part that is vital in biomedical data signal analysis acquired in wearable biomedical sensors due to its subjectiveness in a different environment- often filled with varying sorts of noises, such as thermal and acoustic noises and interference. When a sufficient amount is added to the signals, even if it is a good signal, it can alter its totality. It can change its attributes as well in seemingly significant sampled signals. Moreover, since most of the wearable biomedical sensors are low-power, the unsupervised classification is ineffective under a low signal to noise ratio (SNR) by suppressing these kinds of noises through signal processing by applying the filtering threshold method. When the spectral characteristics of the noise are so similar or near to that of the sensor-received signal, the detection performance may be degraded [56]. In addition to that, in wearable biomedical sensors, physical sensor fitting also affects the electrical properties of the electrodes in moving body parts. Furthermore, electro-physiological data are seemingly random and vary differently from other individuals electrically; and have difficulty in identifying which are those highly predictive and random signals, just like samples in electroencephalogram (EEG) sensors.

Complicated and straightforward mistakes of low-quality data collection are unavoidably present in data input and acquisition. Although much effort could be expended into this front-end procedure to reduce entry mistakes, the truth remains that mistakes in massive datasets are prevalent. Moreover, this rate still needs to be lowered, which might lead to erroneous interpretation and decision-making [57]. Understanding the effects of these inconsistencies is crucial because it helps with understanding various practical implications. When it applies to

the classification problem of an artificial neural network (ANN), obtaining the correct values through comprehensive and extensive quantisation in data-signal processing is essential. Still, it is insufficient to identify which data is gathered without its predictive power. Seemingly correct data values do not guarantee that it holds a valid value in predicting and summarising the totality of multi-dimensional datasets [40]. Mining those data with predictive values is integral to data mining in every Machine Learning and Big Data Analysis.

The current data-cleaning solutions need a more straightforward interface, and automatic random detection will be an integrated approach that suggests the optimal stopping point for the data to be cleaned. Moreover, methods that are based on the analysis of groups of correlated fields (e.g., based on statistical correlation) should also prove powerful [58]. As a result, practitioners rarely reach the equilibrium required to ensure the desired level of clean data in the training set. The better approach in applications with Data Cleaning is the aid of **Mathematical Modelling** of data visualisation for optimisation, and combining data programming [59] intervention to address the identification, visualisation and their suppression of random samples as well as to optimise the dataset in the training of the machine learning.

2.6 Mathematical Modelling

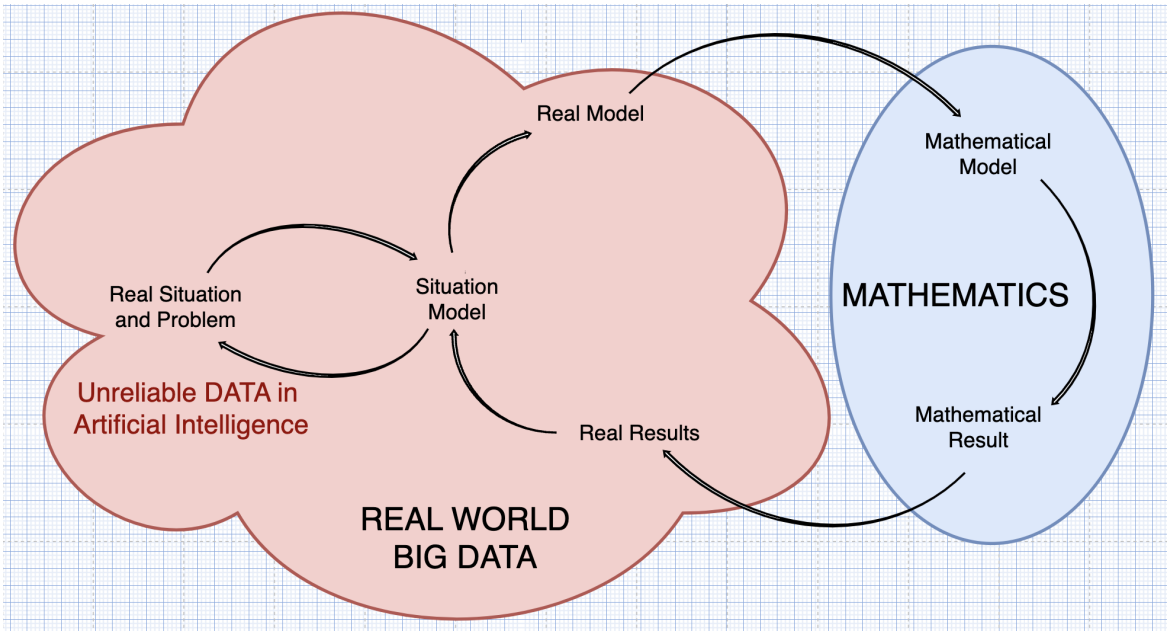


Fig. 2.1 Mathematical Modelling.

Mathematics is a crucial component in the modelling and design of control systems. The mathematical model's goal is to simplify the crucial elements of the system under analysis by simulating reality in a reduced manner. Real-world occurrences are converted into a conceptual universe through Mathematical Modelling as shown in Figure 2.1 [60]. This procedure is started by observing the phenomenon, modelling it mathematically, then simulating its behaviour and predicting its behaviour through simulation.

There are two main categories of Mathematical Modelling [61]:

- **Theoretical Modelling** based on the equations drawn from science are used to explain the system. A few simplifications must be used to model the system in this manner. Systems modelled entirely based on physical principles (equations) are called white-box models, in which the user has all the details concerning how the system works.
- **Experimental Modelling**, based on measurements, is known as system identification. The system's mathematical model is constructed from several sets of measurements, each of which records the response (output) of the system to various stimuli and disturbances (inputs). Black-box models are systems based on experimental data (input-output measurements). It implies that while the user can observe the model's reaction (output) to a specific stimulus (input), they are unaware of the internal mechanism's (principles) workings.

The stages that follow establish a broad methodology for the mathematical modelling process, even though situations may call for many varied approaches to the solution:

1. Define the problem and develop diagrams as needed.
2. Start with a basic model and state the presumptions on different phenomena' facets.
3. Determine significant constants and variables and ascertain their interrelationships.
4. Create the equation(s) that represent the interactions between the constants and variables.

Table 2.6 shows the summary of the characteristics of each type of mathematical model [61].

Mathematical modeling in the realm of machine learning and artificial intelligence (AI), highlighting its importance in extracting insights, making forecasts, and optimizing complex systems [62]. Mathematical modeling involves creating mathematical representations of real-world phenomena to understand and analyze behavior, often using differential equations to describe dynamic systems.

Table 2.6 Summary of Mathematical Model

Model type	Characteristics	Examples
White-box	governing physical laws know, parameters know	linear/non-linear differential equations
Light-gray-box	some of the physical governing laws known, model structure known, parameters unknown	linear/non-linear differential equations transfer function state-space model
Dark-gray-box	some of the physical governing laws known, model structure unknown, parameters unknown	neuro-fuzzy models
Black-box	model structure unknown, parameters unknown	artificial neural networks

In the context of machine learning, mathematical modeling finds applications in various domains:

- **Predictive Analytics:** Developing algorithms for forecasting future outcomes based on historical data, employing techniques like linear regression and decision trees.
- **Optimization:** Utilizing mathematical models such as linear programming to optimize resource allocation and production processes.
- **Neural Networks:** Leveraging artificial neural networks, inspired by the human brain's structure, for analyzing complex patterns in large datasets.
- **Natural Language Processing (NLP):** Employing mathematical models like word embeddings for sentiment analysis and language translation tasks.
- **Recommender Systems:** Using collaborative filtering and matrix factorization for personalized recommendations.
- **Image and Speech Recognition:** Revolutionizing tasks like facial recognition and speech-to-text systems through convolutional neural networks.

Mathematical modeling enhances AI algorithms' performance by fine-tuning parameters, improving accuracy, and enabling real-time decision-making capabilities. The integration of mathematical modeling with cutting-edge algorithms drives innovation, transforming industries and enriching lives. Embracing mathematical modeling is essential for organizations aiming to unlock the full potential of machine learning and AI in the digital age.

2.7 Principal Component Analysis — Dimension Reduction in Artificial Neural Networks (ANNs)

Principal Component Analysis (PCA) is fundamental to studying multivariate data. Its applications are usually used for data visualisation and dimension reduction. PCA converts the features of datasets from high-dimensional data and can to low-dimensional data with the aid of covariance, eigenvalues, and eigenvectors [63]. It is an instrument for identifying high-dimensional data patterns and a feature extraction method. In addition, the components of the transformed vector are arranged so that the first and second components have the greatest and next most significant variance. It allows for extracting critical feature vectors from multidimensional datasets to be more visualised.

The classification problem is based on the rating or measurement of objects, called features or dimensions. The problem in multidimensional data in performing the classification problem is having relatively few or high training samples available [64]. It is hard to pinpoint if all features or dimensions are needed in classification or prediction. When using it as a pre-processing method in a forecasting model, the paper explains particular aspects concerning Principal Component Analysis (PCA). An effective used statistical technique for dimensional reduction and the PCA to decorrelate the input data before a neural network architecture is trained. This approach in the PCA transformation matrix shows essential regularities that can enhance the entire Artificial Neural Networks (ANNs) model.

In often cases of the utilisation of PCA via biomedical sensors, most researchers use principal component analysis, such as reduction in dimensionality or feature space [65–69], feature extraction in different data visualisation [70–72], feature selection tools in the machine and deep learning applications [73–77] for machine learning application.

In summary, the data modelling of the analysing the data shows in Figure 2.2. After acquiring the data from sensors or surveys, then pre-processing, then data integration, modelling, to post-processing - that includes the evaluation and interpretation of visualised graphs and diagrams. PCA falls in the pre-processing section in data cleaning and reduction, a vital part of data integration.

With the implementation of PCA in many fields, it is highly flexible. It continues to be the subject of many studies, ranging from modern model-based methods from algorithmic ideas to neural networks, while it is one of the earliest multivariate techniques. By training a multilayer neural network with a small central layer to recreate high-dimensional input vectors, the dimensions of datasets can be reduced, and high-dimensional data can be transformed into low-dimensional codes [63]. Used during the pre-processing stage of

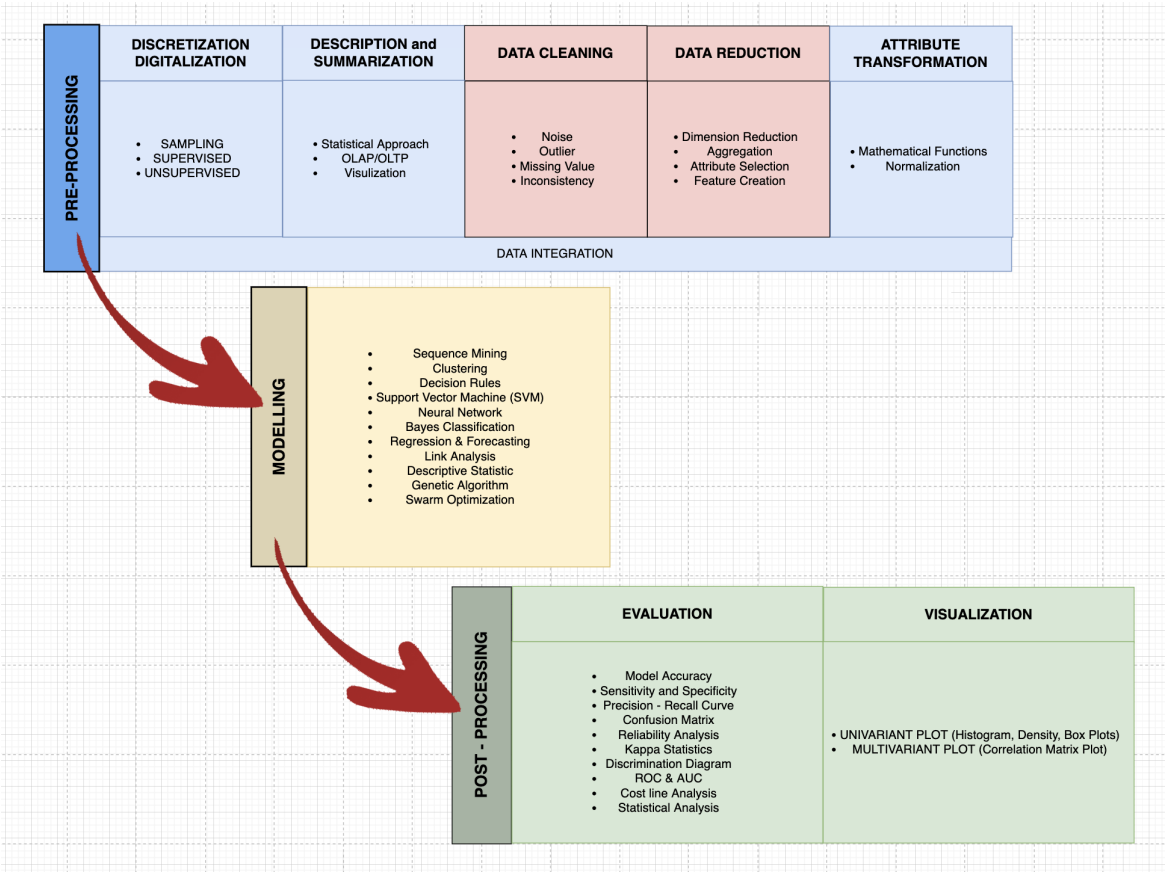


Fig. 2.2 Data Modelling.

a machine learning endeavour, PCA allows for extracting crucial feature vectors from multidimensional datasets and has been used extensively in machine learning.

PCA typically retains the first few components of the transformed vector, which make up the original vector's most significant variance. The results show a substantial decrease in the input vector dimension if the original components are strongly correlated. The primary strength of PCA on artificial neural networks is the acquisition of non-linear functions that enable non-linear and multidimensional dependencies to be treated. Therefore, before adding the inputs to the neural network, the use of PCA to reduce the input dimensions must be introduced. Although there are a lot of significant articles stating that PCA-Dimension Reduction gives a significant increase in quality or accuracy, it is conclusive. Nevertheless, the pitfall of implementing PCA is that only the linear relationships between the input vector components are considered if not adequately implemented and lose some crucial information along the process [78]. In some cases, it could also contribute to the loss of some non-linear information if the dimension of a vector using a linear transform decreases, and independent samples become less interpretable dimensionally, which is a considerable

disadvantage, primarily if it is used extensively in machine learning such as Artificial Neural Networks(ANNs).

2.8 Recent Advancement in Principal Component Analysis

Recent developments in PCA research have focused on improving the technique's performance and adaptability to different data types and structures, such as sparse PCA, which aims to identify the most critical variables in a dataset while ignoring the noise [79]. The robust PCA is designed to handle outliers and anomalies in the data. In addition, there has been research on extending PCA to handle non-linear relationships between variables, resulting in techniques such as kernel PCA [80]. PCA has several adjustments that make it practical for a broad range of scenarios and data types in different fields, even though it is a frequently used and flexible descriptive data analysis technique in its basic form.

There have been suggestions for PCA modifications for various data types, including binary, ordinal, compositional, discrete, symbolic, and data with unique structures like time series or datasets with similar covariance matrices. Other statistical techniques like linear regression (including principle component regression) and even concurrent clustering of persons and variables have significantly benefited from using PCA or PCA-related methodologies. However, as they are founded on factorial decompositions of specific matrices, methods like correspondence analysis, canonical correlation analysis, and linear discriminant analysis have a standard methodology with PCA[79]. The PCA literature is extensive and cross-disciplinary. Applications, methodological innovations, and new adaptations continue to emerge.

PCA has various applications in various fields, and new applications emerge as the technique develops. Some of the emerging applications of PCA in research include:

- **Healthcare**: PCA reduces the dimensions of healthcare data, making it easier to analyse and interpret. It can also identify patterns in patient data and improve diagnosis and treatment.
- **Image resizing**: PCA can help resize an image by identifying the most critical features and reducing the dimensionality of the picture [81].
- **Finance**: PCA is used to analyse stock data and forecast returns. It can also identify patterns in financial data and improve investment strategies [81].

- **Data visualisation**: PCA can visualise data in two or three dimensions by projecting high-dimensional data onto the first few principal components. It can aid in locating data patterns or clusters that may not have been visible in the initial high-dimensional space [81].
- **Noise reduction**: By locating the underlying signal or pattern in the data, PCA can also lessen the impacts of noise or measurement errors [81].
- **Colour technology**: PCA has been an essential mathematical tool in colour technology since the 1960s. It is used to analyse colour data and identify patterns in colour images [82].

PCA has various emerging applications in multiple fields; new applications will likely occur in other areas as the technique develops.

Another recent development uses deep learning techniques to perform PCA, resulting in processes such as deep PCA [83]. Deep PCA is an extension of traditional PCA that combines deep learning techniques with PCA to learn non-linear mapping from high-dimensional data to a lower-dimensional representation. Here is how Deep PCA works:

1. The input data is first transformed into a lower-dimensional space using a deep neural network.
2. The principal components are then computed in the lower-dimensional space, which captures the most variance in the data.
3. The principal components are transformed back into the original high-dimensional space using the inverse of the deep neural network.

By leveraging the power of deep learning, Deep PCA can provide better representations and capture more complex relationships between its features in the data compared to traditional PCA. It has shown promising results in tasks such as image compression, feature extraction, and anomaly detection. **However, most further studies and recent research regarding PCA are connected more to reducing, lowering, and selecting certain features or dimensions. Discovering the potential of PCA by selecting the highly predictive samples and removing the randomness is still undiscovered and unexamined.**. Figure 2.3 shows the general thought process in solving the data cleaning problem by introducing the other attributes of Principal Component Analysis - Sample Reduction Process

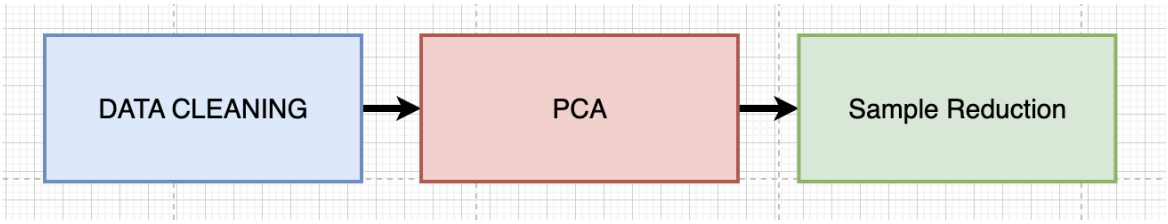


Fig. 2.3 Data Cleaning - Thought Process Paradigm

2.9 Findings

Data cleaning is widely acknowledged as a critical yet time-consuming task across industries, requiring significant labor and consuming a substantial portion of data scientists' time. Identifying and classifying the types of dirty data within large datasets presents a formidable challenge, particularly when dealing with massive volumes of data. The proliferation and accessibility of data from diverse sources, facilitated by rapid advancements in electronics, telecommunications, and the Internet, have exacerbated issues of unreliability and poor data quality.

Therefore, the essence of data cleaning cannot be overstated in industries aiming to minimize data discrepancies and ensure the reliability of analytical outcomes. This involves quantifying the appropriate volume of data to be utilized, a task of paramount importance. However, among the literature surveyed concerning data cleaning for big data and its application in AI, a notable gap emerges: the absence of a quantifiable metric to assess the predictive power of individual data points.

This gap can be addressed through the utilization of mathematical modeling, which offers a solution to the complexity of data and its predictive potential. This is particularly crucial in contexts such as the analysis of physiological datasets, where accurate classification and prediction of physiological behavior are essential. The commercialization of wearable biosensors, despite their cost-effectiveness, introduces challenges related to accuracy and sensor fit, which can degrade the electrical attributes of acquired data. In such scenarios, mathematical modeling plays a pivotal role in enhancing predictive accuracy and facilitating informed decision-making in biomedical practices.

Chapter 3

Methodology

The proposed framework addresses the limitations of the ANN in processing multidimensional datasets through the use of PCA with the sample reduction process (PCA-SRP) [84]. PCA is used to analyse the multidimensional data and if a significant relationship exists between the features of a dataset. It is arranged from the most significant samples to the least to be visualised and put the multidimensional data into perspective. It focuses on implementing dimension reduction by removing data from the multidimensional dataset feature columns that do not have high inter-feature covariance; however, the proposed framework using the loading scores can dissect all the samples or rows in the multidimensional dataset. Figure 4.1a provides a graphical description of the proposed PCA-SRP-based ANN solution.

The PCA-SRP Python subprogram as shown in Appendix A - **Subprogram - PCA - SRP** generates a new set of multidimensional data with fewer samples based on the screen plot to identify the most significant samples. It is used in the Python ANN program as an input to ensure correct classification.

For this research, it is more of light-grey box for the non-linearity attributes of the datasets and black box for Artificial Neural Networks (ANN). The proposed Mathematical Model for Data Cleaning approached in training the Artificial Neural Networks, as shown in Figure 3.1.

The best way to clean big data is to use a "Mathematical Model" by converting the multidimensional data into one-dimensional metrics that generalize its features, then sort it according to its importance and make a threshold bias that limits and removes the unnecessary samples.

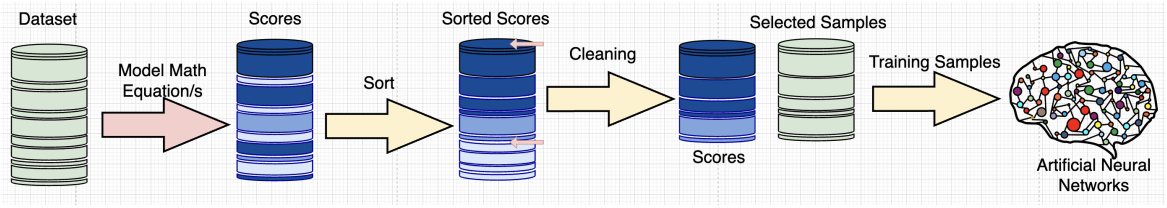
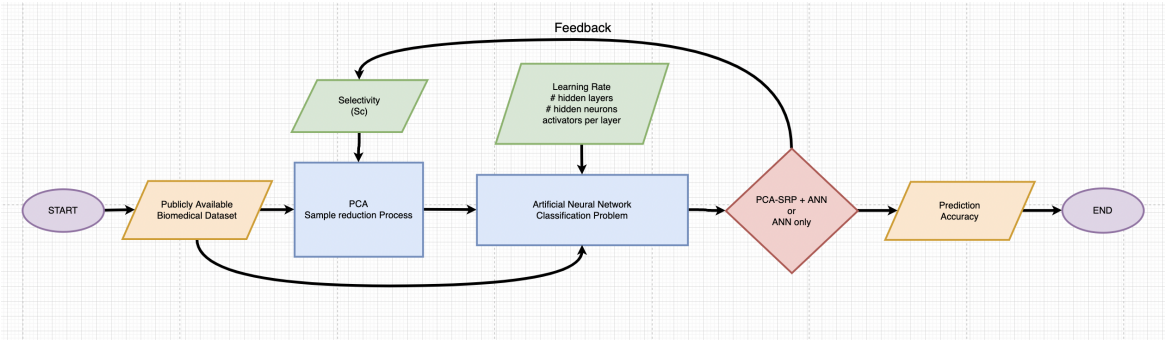
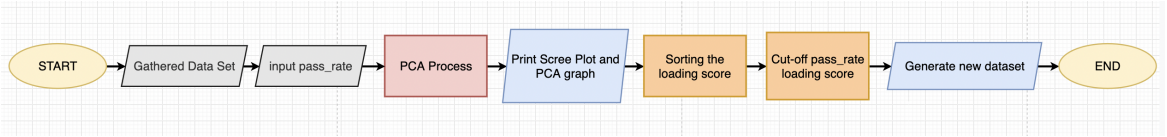


Fig. 3.1 Proposed Mathematical Model for Data Cleaning of Artificial Neural Networks.

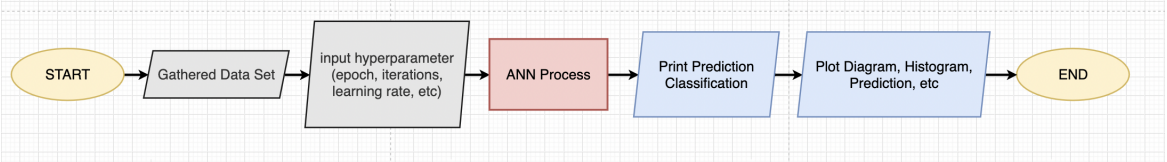
3.1 PCA-SRP and ANN Methodology



(a) Conceptual flowchart of the proposed approach.



(b) Detailed breakdown of the steps in PCA.



(c) Detailed breakdown of the steps in ANN.

Fig. 3.2 Procedural Conceptual Framework

Figures 4.1b and 4.1c show the algorithm used in Python to implement and analyse the concept. The implementation starts by extracting the data in .csv format and injecting it into the PCA-SRP process. Principal components are extracted through the abovementioned threshold based on the input dataset, resulting in a new set of multidimensional datasets used in ANN.

The proposed PCA-SRP utilised in the basic Feed Forward ANN model consists of two hidden layers consisting of 32 neurons and 16 neurons, respectively. The learning rate was set at 0.1 and trained for 100 epochs. The input parameters in PCA-SRP-based ANN are

Table 3.1 PCA-SRP and ANN parameters.

PCA- SRP and ANN Parameters	**Values**
Most significant sample size	$> Sc \times (PCA_{SRP}(M)_{max})$
Learning rate	0.1%
Epochs	100
Number of ANN neurons	2 hidden layers (32 and 16, respectively)
Activation functions used	ReLU in hidden layers SoftMax in final layer

shown in Table 4.1 and justified in Chapter 3.2. Then, the comparative performance of the Accuracy of the PCA-SRP-based ANN against a model trained using ANN alone on different datasets.

The ANN model is prescribed to be a basic Feed Forward ANN with 32 and 16 neurons, respectively, **using ReLU and SoftMax activators**, RELU is conventionally used as an activation function for the hidden layers in a deep neural network, and Softmax as output activation [85]. A **testing size of 20%, training size of 80%**, it is described that the $p \approx 0.8$ is empirically the best division into the training and the testing test, due to the decreasing $p(1 \dot{-} p)$ error measurement at $p \leq 0.8$ [86]. Moreover, a **learning rate of 10%**, for this reason, the loss starts decreasing significantly between LR 0.001 and 0.1 as shown in the Figure 4.2 [87] . One hundred (100) epochs are chosen to visualise the gradual changes and identify the behaviour of the performance metrics. Furthermore, $Sc = 0.98$ or PC_1 of the highest loading score will be used in PCA analysis.

The dissertation's discourse to meet the objective is designed as follows as shown in Figure 3.4:

1. Identify the mathematics that helps to solve the problem.
2. Utilize the mathematical modelling by incorporation of equation.
3. Validate the constructed mathematical model by the used of different small, publicly-available physiological (raw) dataset.
4. Test using massive raw dataset with different metrics to quantify its measurement
5. Validate using mixed random/abnormal raw signals and test check its behaviour.

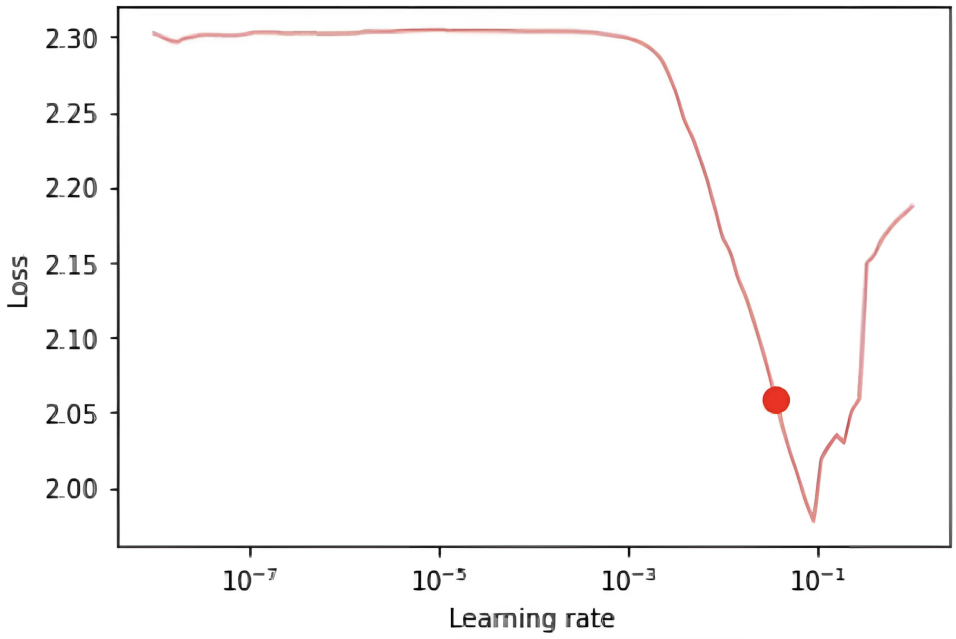


Fig. 3.3 Typical Loss vs Learning Rate (LR) Behaviour.

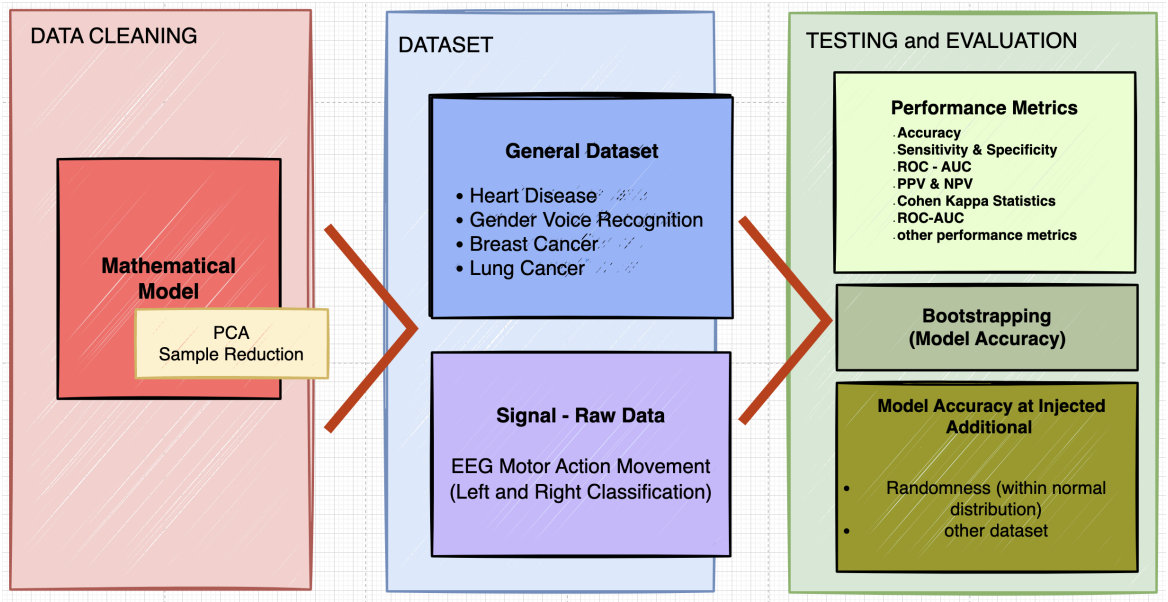


Fig. 3.4 Data Cleaning - Dataset - Evaluation

3.2 Principal Component Analysis—Sample Reduction Process (PCA-SRP)

Principal Component Analysis (PCA) is usually utilised in dimension or feature reduction and significantly increases accuracy and efficiency along with other machine learning techniques in many applications aside from biomedical application [88–92] to a different field. Nevertheless, some of the information is lost and needs to be recovered and reversed during the process of dimension reduction [78].

This study used PCA, not as a dimension reduction, but as a sample reduction, to remove the unwanted random samples in multidimensional datasets before fetching in machine learning training processes, particularly in Artificial Neural Networks, by converting the high dimension dataset to its PC major axis, it collectively transform into more graspable dataset. Through this newly transform dataset, it can deliberately identify the predictive power values of each sample. Furthermore, measuring and quantifying the predictive values is the main criterion for sorting the samples based on the significance as a dataset. Hence, given these existing gaps' justification for using PCA as Sample Reduction Process (SRP) for the ANN classification problem, additional efforts must be made and a systematic investigation of the publicly available physiological datasets and the EEG mental activities.

Using publicly available physiological datasets, some of the benchmark example listed in Chapter 1 provides qualitative analysis - by observing the training samples that fed on the basic Feed Forward ANN and analysing its accuracy effect. Feed Forward ANN is used for this because of its basic random function approximation. Moreover, most benchmark datasets have been observed and analysed in complicated ANN classification models [93–96]. Testing the effectiveness of the classification in basic ANN Model such as Feed Forward ANN might show and help significantly improve the previous literature with complicated ANN structures by adding this concept in their research study.

The PCA reduces the dimensions or features vertically ; for the proposed methodology, it reduces the random samples horizontally. It emphasises that extracting features, such as covariance, eigenvalues, eigenvectors and dimension reduction is not a novel technique [97]; instead, sample data reduction . The proposed implementation of a Sample Reduction Process (SRP) [84] using Principal Component Analysis (PCA) in identifying the random samples that cause irregularity in the multidimensional datasets and omitting or reducing sample randomness of massive physiological dataset is the main focus of this study.

The observation and the application of the **PCA - sample reduction process (SRP)** in this study are utilised for data cleansing of the dirty multidimensional datasets to identify the behaviour of classification problems in Artificial Neural Networks (ANNs) and identify

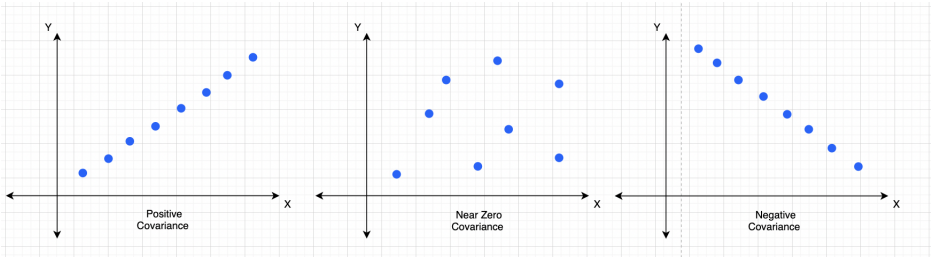


Fig. 3.5 Covariance Interpretations.

recommended threshold ranges are unique approach in data cleaning process. It discussed a specific technique for qualitative random detection and the omission of such. These techniques are explained with a motivating example highlighting artificial neural networks' deep learning classification problems using publicly available physiological datasets in biomedical applications [84]. The simplicity of this technique makes it portable and can apply to various tasks for fast and accurate classification of typical or commonly used artificial neural networks.

Principal Component Analysis (PCA) minimises the dimensionality of a dataset with many connected variables while keeping as much variance as feasible. The conversion of the new collection of uncorrelated variables known as principal components (PCs) preserves most of the variance included in the original variables. A new set of dimensions or orthogonal measurements are linearly independent and ranked according to the data covariance. It means the more crucial principal axis occurs first (more important = more covariance/strong relationship in other dimension). Understanding the PCA, variance, covariance, eigenvalues, and eigenvectors is essential in this concept [98].

Figure 3.5 shows large positive covariance, which means that x and y are entirely related, i.e., as x increases, y also increases. Negative covariance portrays the exact opposite relation. Though zero covariance means x and y have no relation.

Visualisation of the data is an excellent approach to understanding the patterns in multidimensional datasets. When information is placed in the horizontal and vertical axis (2-dimensional plane) as shown in Figure 3.6. Using PCA, it is straightforward to understand and discern its pattern; however, the difficulty of conceiving it visually in multidimensional data with many features to consider and performing the data analysis computation becomes complex. Principal Component Analysis prerequisites require discovering patterns between the datasets so that data are distributed across each dimension by first analysing the contributions of each feature in providing information to the overall dataset through eigenvector analysis. It then reduces dimension by keeping the feature columns with the highest eigenvalues.

The actual dataset has been shown in the Figure 3.6a in 2D space. Due to the PCA implementation, it rotates the principal axes (PC_1 and $PC-2$) in order to aligned the data samples to visualized and interpret easier as shown in the Figure 3.6b, which projects a set of multidimensional data onto a two-dimensional space, is used as an example to demonstrate the efficacy of PCA. Due to the high dimensional nature of the data points, it can be challenging to identify a linear correlation between them. as depicted in Figiure 3.6. The points are represented as column vectors before being aggregated into a matrix X.

When X is rotated and scales, it turns into matrix $\widehat{M}$ as shown in Equation 3.1 with corresponding eigenstructures such as eigenvalues U and eigenvector V.

$$X \rightarrow \widehat{M}_{(U,V)} \tag{3.1}$$

Given the PC space of the matrix X, when PCA is applied, it rotates the axis called PC_1 and PC_2 axes to where the most point samples that are covariant to each other lie.

When $X_{(n,k)}$ rotates to PC space, where matrix $\widehat{M}$ turns into matrix M.

$$X \rightarrow M_{(\nu,\lambda)} \tag{3.2}$$

Wherein, $\nu \in V$ and $\lambda \in U$. By definition of eigen-structure, eigenvalues λ is the scaling of dimension, while eigenvectors ν is the direction that contains covariance

Furthermore, Equation 3.2 is utilised to have the desired highest PC that yields an eigen-structure of λ and ν. ν_{PC_1} is generated using covariant eigenvector ν with the corresponding top two eigenvalues in $\lambda_{1,2}$, which best represent the data points, and are then selected before plotting it as discussed and shown in the Figure 3.7 [99].

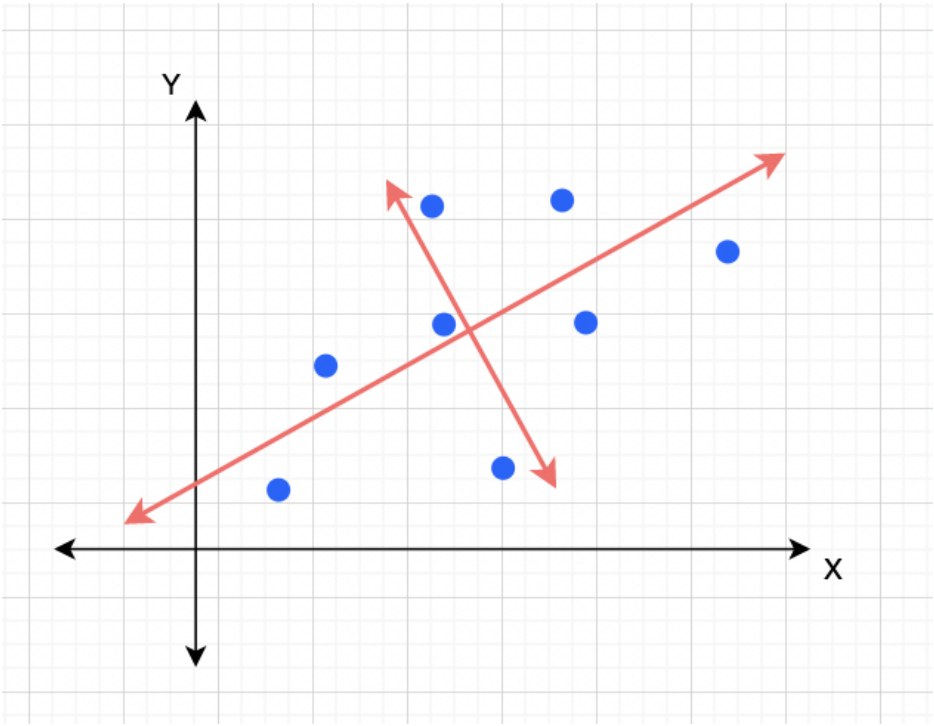
More discussion about basic knowledge of PCA and ANN in Appendix C.

The decomposition of the matrix $X_{(n,k)}$ into two matrices $U_{(k,k)}$ and $V_{(n,k)}$ are orthogonal - which means if the product of a matrix and its transpose gives an identity value. The V is usually the loading score matrix, and the U is the covariant dimension scores matrix. The matrix U are defined as the weight for each original dimension in the principal component, and the matrix V contains the covariant of original data in a rotated coordinate system.

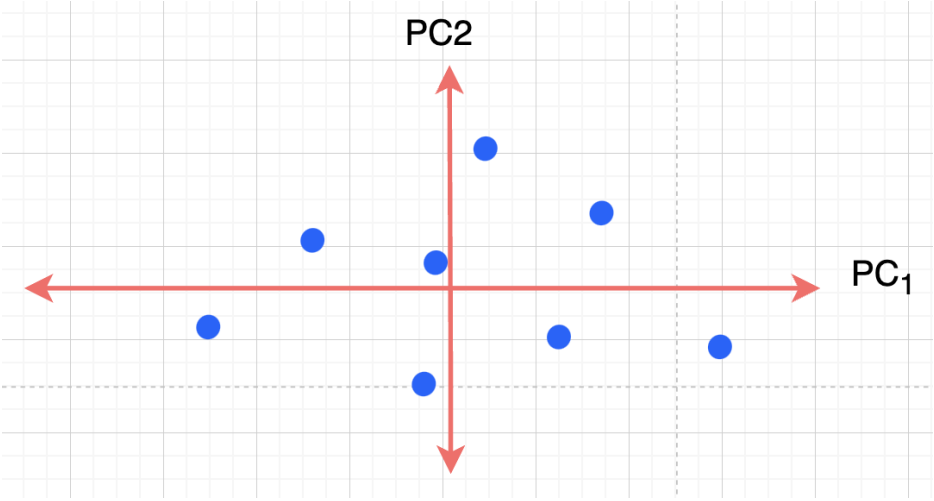
The implementation of Principal Component Analysis to identify the principal axis PC_1 and PC_2. Then it generates the highest eigenvalue ν_{PCA} and with corresponding eigenvector λ_{PCA}.

$$PCA(X) \rightarrow [\nu_{PCA}, \lambda_{PCA}] \tag{3.3}$$

The eigenvector ν_{PCA} is still unexplored and not much of the literature and its application as discussed in Chapter 2.8.



(a) 2-D Projection of data points.



(b) Newly constructed 2-D projection of data points.

Fig. 3.6 PCA 2D Data Reconstruction.

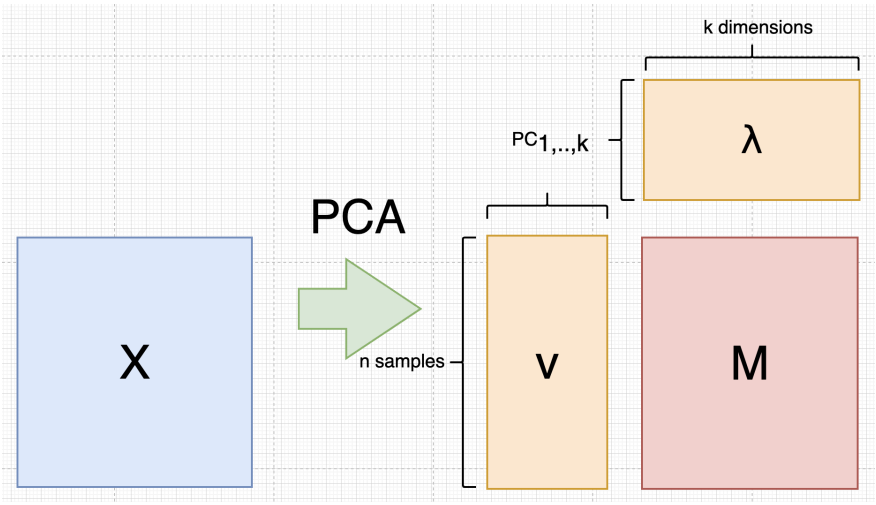


Fig. 3.7 PCA and Eigenstructures.

$$V_{PCA-SRP} = v_{PCA_1} \tag{3.4}$$

where $V_{PCA-SRP}$ is the loading score of each sample in dataset n in PC_1 eigenvectors of the samples in matrix X.

Appendix C.3.2 discusses that the explained variance ratio is equal to principal components, and the cumulative sum of all PC equals 1. Furthermore, if the PC_1 signifies a high variance ratio, the samples in PC_1 rank the highest in the order of importance. Therefore, it is assumed that the first few PC can be a reference as the limiting threshold as Selectivity (Sc) biased threshold.

$$S = \begin{cases} 1 & \text{if } V_{PCA-SRP} \geq Sc \times max(V_{PCA-SRP}) \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

3.2.1 Software Tool Used

Regarding software implementation, developers and data engineers prefer Python programming language. It is the best option for projects or programming involving AI and extensive data analysis as Python is a simple language, the abundance of a mature and supportive Python community, support from renowned corporate sponsors, an extensive and popular selection of libraries and can work with heavy-hitting frameworks shown in Figure 3.8 [100] such as the following:

- **TensorFlow** an end-to-end machine learning platform.

Fig. 3.8 Python Programming Language and some supported libraries.

- **Numpy** is an open-source Python library that facilitates efficient numerical operations on large quantities of data.
- **Pandas Dataframe** is a structure that contains nth-dimensional data and its corresponding labels.
- **MNE** is an open-source Python package for working with EEG and MEG data.
- **Sci-kit-learn** or sklearn, is a Python library to implement machine learning models and statistical modelling.
- **OpenCV** is a great tool for image processing and performing computer vision tasks.
- **Keras** is a powerful, easy-to-use free, open-source Python library for developing and evaluating deep learning models.
- **Matplotlib** is a comprehensive library for creating static, animated, and interactive visualisations in Python.
- and many more other libraries.

Therefore, the maximum utilisation of Python Programming is the better choice for implementing this research study for the reason given above.

3.3 Selectivity (Sc)

One of the challenges of the PCA is the identification of features; it reduces the dimension, but knowing which of the features in the dataset is hard to find; although it pins down the PC_1 and PC_2, it comprises different features. It is the same with the samples. It is hard to identify which of the samples is good and random. Selectivity is the key to data cleaning; since the data has randomness, it is difficult to assume the precise value of Selectivity (*Sc*) but only the approximate value.

This section provides a hypothetical assumption of *Sc* and the validation if the assumption is valid in Chapter 4.3.1. The accuracy testing was conducted for the specified dataset to identify the recommended *Sc* and to maximise the number of samples of the cleaned set *S* and maximise the number of samples of the removed set *R* in the PCA-SRP.

Assume two sets of data *D*—cleaned data *S* and *R*.

$$D = S + R \tag{3.6}$$

$$\%Random = R/S \tag{3.7}$$

$$R = \%Random \times S \tag{3.8}$$

Therefore,

$$D = S + \%Random \times S \tag{3.9}$$

$$D = S \times (1 + \%Random) \tag{3.10}$$

The **index** of the samples are the location of the sample in the dataset, Indexing the dataset *D* for identifier, where *n* is the number of cleaned data *S*. To identify the the known cleaned samples to random samples.

$$D = \begin{cases} S & \text{if } 0 \leq \text{index} \leq n \\ R & \text{if } n < \text{index} \leq n \times (1 + \%Random) \end{cases} \tag{3.11}$$

S cleaned data and *R* randomness are randomly distributed in the dataset, only maintaining its index as identification. Then, PCA—sample reduction process (SRP) is applied and sort it based on the loading score through the concept of covariance, eigenvalues and eigenvectors as discussed in Appendix C.2.

Where *Sc* is the Selectivity of data in the Table 4.1 that is used as cut-off biased to separate and find the True Positive (TP) and False Positive (FP) as well as True Negative (TN) and False Negative (FN).

The accuracy of *S* and *R* is based on the Equation 3.12 and 3.13.

$$S_{accuracy} = S_{set}/n \tag{3.12}$$

$$R_{accuracy} = R_{set}/(n \times \%Random) \tag{3.13}$$

Therefore, in finding the most efficient *Sc* is where $S_{accuracy}$ and $R_{Accuracy}$ lines meet.

3.3.1 Test Selectivity (Sc_{test} and Recommended Selectivity (Sc)

Sc_{test} is utilized to find the recommended Selectivity, which is vital and crucial that determined the efficiency of the dataset. The actual cleaned *S* set and random *R* set are needed to identify the accuracy by identifying the $S_{accuracy}$ and $R_{accuracy}$ lines and applying the Equation (3.14).

$$Sc_{test} = index[max(S_{accuracy} \times R_{accuracy})] \tag{3.14}$$

The minimum *Sc* is the Selectivity wherein all the *S* samples have been part of True Positive, while the maximum *Sc* is no random *R* samples included in the process, as shown in Figure 3.9. Nevertheless, this is the hard part to identify in the reality of the dataset. However, using Performance Metrics and knowledge-based ideas might approximate the Selectivity (*Sc*).

The cruciality of Selectivity, whether it loses or not important information as discussed in Chapter C.3.2 - **Explained Variance Ratio and F-Distribution** in the Equation C.42 give us an insight into how the *PC*'s arranged and sorted loading score in the dataset as shown in the Figure 3.10.

So, in Figure 3.11, most of the high loading score samples are in PC_1, sometimes also in PC_2.

Therefore, Equation 3.15 can be used as recommended Selectivity (*Sc*).

$$Sc_{recommended} = PC_1 \tag{3.15}$$

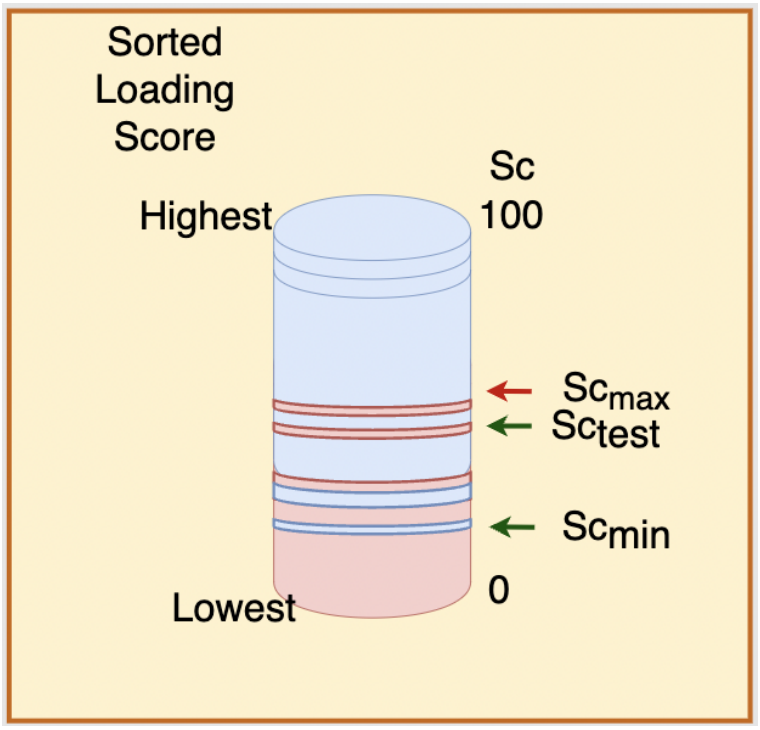


Fig. 3.9 Sc—accuracy testing representation.

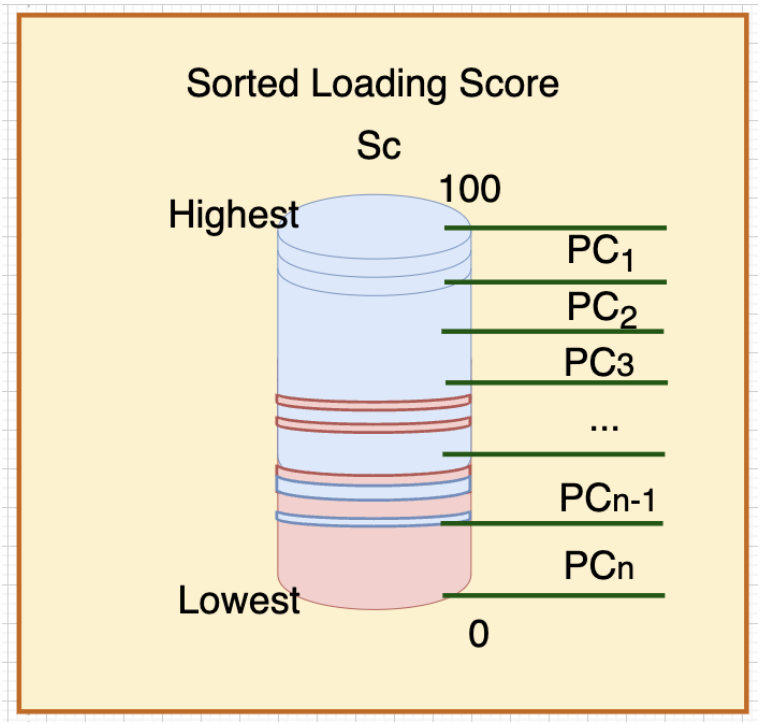


Fig. 3.10 Principal Component Representation in Loading Score.

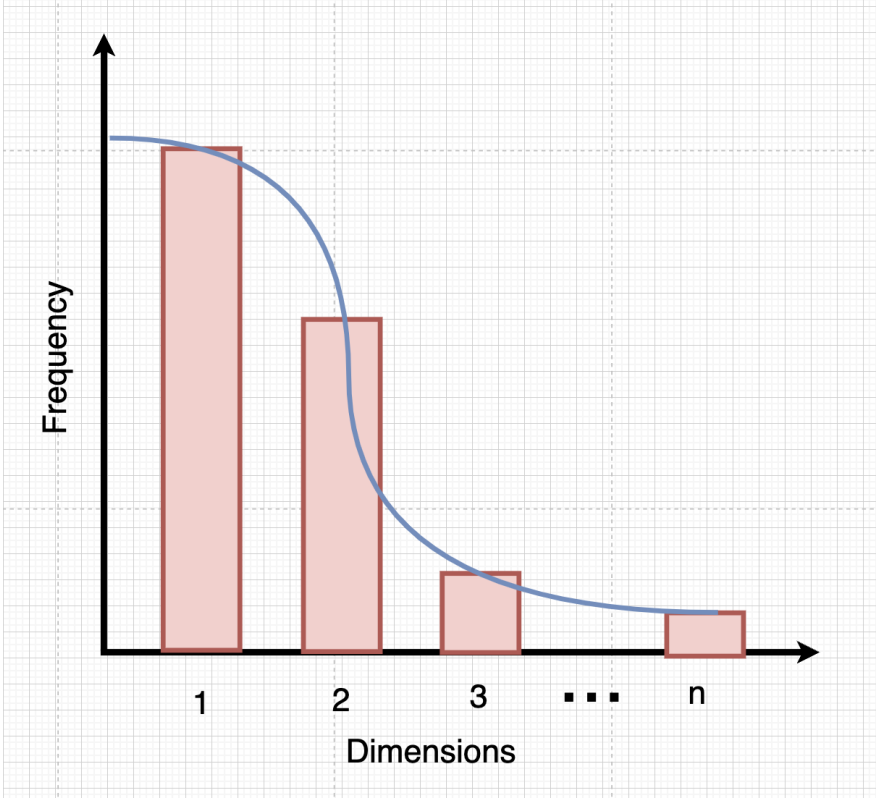


Fig. 3.11 Frequency Distribution Diagram

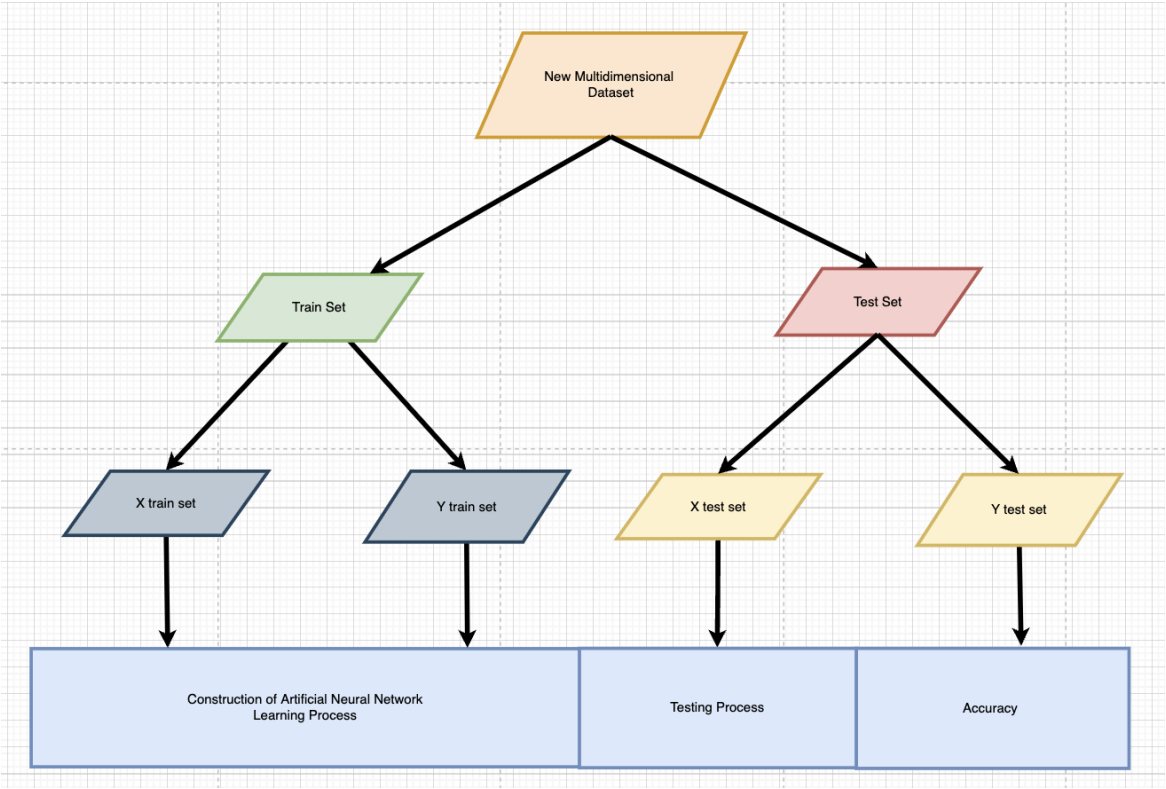


Fig. 3.12 Cross Validation Approach.

3.4 Cross Validation

The statistical approach of cross-validation is used to measure the skill of machine learning models. It is widely used in applied machine learning to compare and select a model for a specific predictive modelling issue since it is simple to comprehend, implement, and produces skill estimates with lower bias than other approaches. It may be used to estimate a classifier's performance and fine-tune the model parameters.

Suppose all accessible datasets are utilised for training the classifier during validation. After that, it is put to the test on the same dataset. It is prone to over-fitting; as a result, the classifier may perform well on current data but badly on future test data.

One particular solution method is the **K-fold cross validation** to overcome this drawback. It is a procedure used to estimate the skill of the model on new data, as shown in figure 3.12. The available dataset is split into two sets in validation: one for training and the other for testing the model, allowing the model to be tested on previously unknown data. The outcomes of this method are significantly reliant on the training/test split used. Most common K - fold cross-validation are 4-fold, 5-fold, and 10-fold cross-validation, which are 75/25%, 80/20%, and 90/10% train/test sample set, respectively.

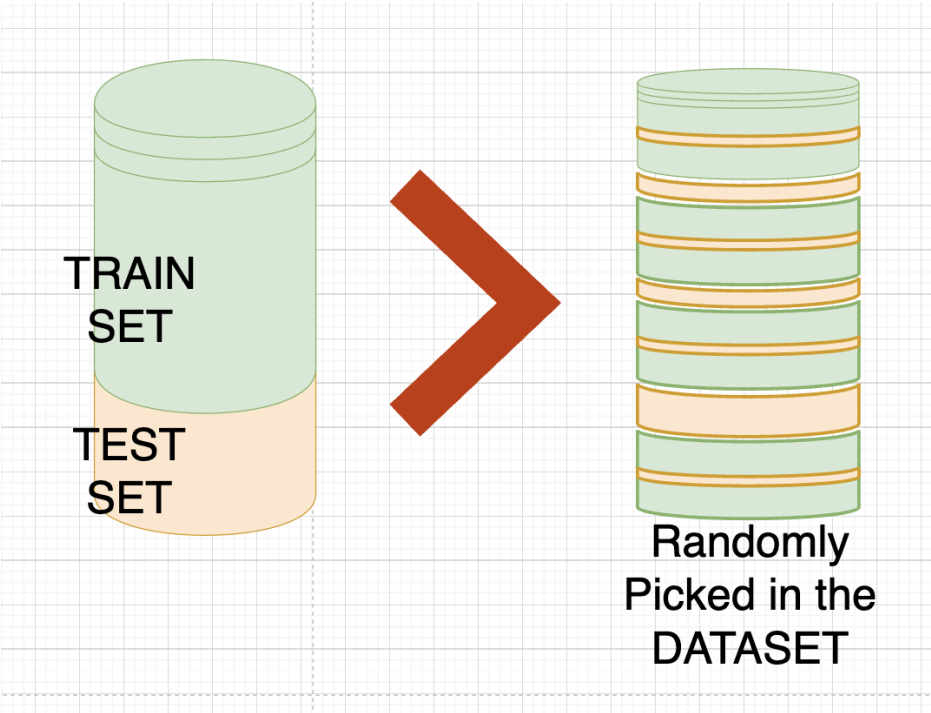


Fig. 3.13 Train - Test Dataset.

The test set's occurrences may need to be more complex or easier to categorise, skewing the findings. The cases in the test set, on the other hand, may be helpful in training, and if they are left out, the prediction performance may decrease, resulting in skewed findings shown in figure 3.13, so it is randomly selected in the dataset. Appendix A.1 shows the sub-program for split cross validation using Python Implementation.

3.5 Classification Model Evaluation Metrics

Results must be presented consistently to enable understanding and comparison across many research groups. Therefore, evaluation methods must be carefully chosen and specified. Performance metrics, error estimates, and statistical significance testing are all part of evaluating the classifier's performance, including accuracy, error rating, and additional measures derived from the Confusion Matrix, such as Recall, Specificity, Precision, and False Omission Rate (FOR) shown in Figure 3.14 [101].

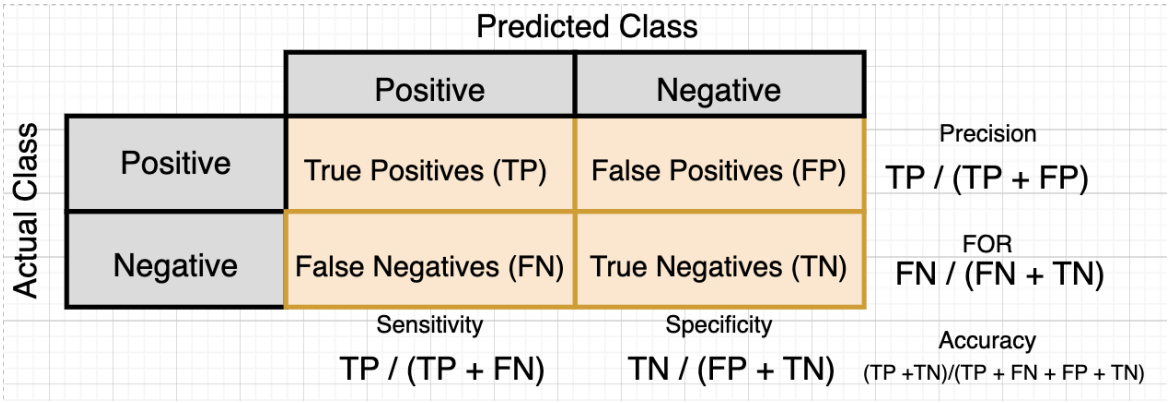


Fig. 3.14 Confusion Matrix

Performance Measures

After the classification, the model's performance should be evaluated using predefined metrics. The first primary metric that comes to mind is accuracy. The fraction of correctly identified cases is measured by accuracy. However, accuracy is only sometimes dependable because it might be deceptive if the dataset is uneven. It describes other metrics on a table called the confusion matrix similarly. The confusion matrix is a graphic representation of a classifier's performance shown in Figure 3.14. Each row of the matrix represents the actual classes of the data, while each column represents the projected classes.

The confusion matrix displays the number of accurate and incorrect classifications, identifies the incorrectly classified class, and provides information about the different mistakes made by the classifier (correct and incorrect predictions for each class). For reporting outcomes in M-class classification, it is an excellent choice. The data, however, could be clearer to contrast and analyse. It uses several parameters that were taken from the confusion matrix instead.

Given the binary classification problem, it can define certain terms of the confusion matrix as follows:

- **True positives (TP)** is an outcome where the model correctly predicts the positive class.
- **True negatives (TN)** is an outcome where the model correctly predicts the negative class.
- **False positives (FP)** is an outcome that indicates that the model predicts it has a specific condition when it does not have the condition.

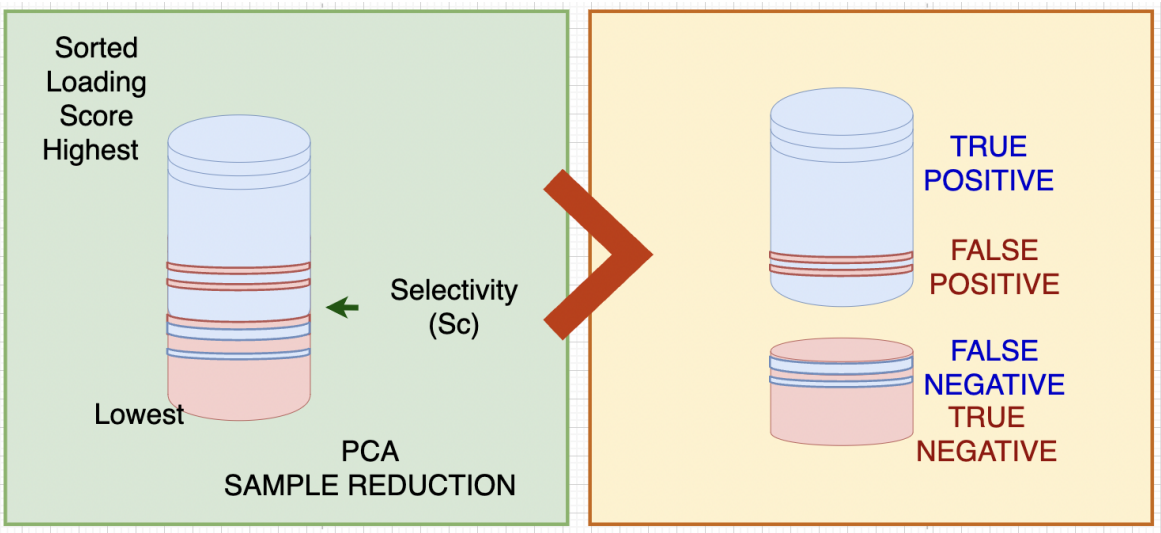


Fig. 3.15 Illustration of Confusion Matrix.

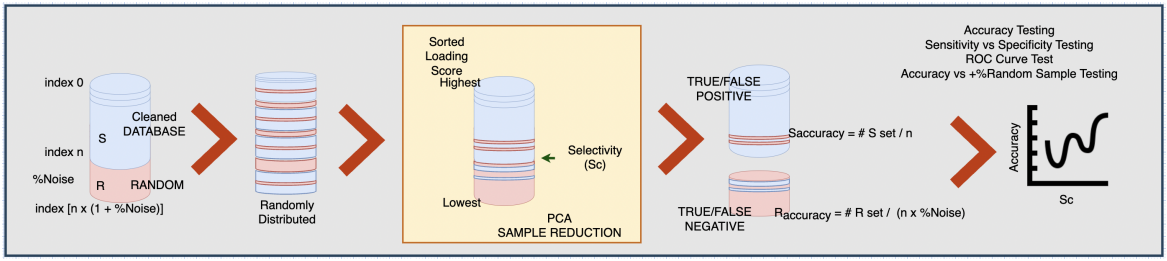


Fig. 3.16 Detailed Sensitivity vs specificity testing procedure.

- **False negatives (FN)** is an outcome which wrongly indicates that a particular condition or attribute is absent.

The most commonly used evaluation metrics are:

Accuracy

Measures the instances that are correctly classified. The chance of a proper classification over a specific number of repeated measurements is known as accuracy. The likelihood that an inaccurate classification was made is represented by the error rate, e = 1-p. It works well if the classes are balanced or if there are an equal amount of samples in each class. However, Accuracy and Error Rates do not consider whether or not the dataset is balanced. Even when the categorisation is underperforming, the assessment may show a high accuracy rating if one class happens more frequently. The quantity of classes and cases affects these parameters. The probability level in a 2-class problem is 50%. However, the confidence level varies based on the number of cases.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.16}$$

Precision (PPV)

A Positive Predicted Value (PPV) measures the proportion of correctly classified positive cases. Precision, also known as Positive Predicted Value, is determined by a 1 - False Detection Rate(F). The ratio of false positives to the total of both true and false positives is known as the false detection rate. The percentage of accurate classifications is measured. While accurate identification of positive samples is vital to the problem, Precision should not be employed when the positive class is more significant (imbalanced dataset).

$$Precision = \frac{TP}{TP + FP} \tag{3.17}$$

Sensitivity or hit rate (TPR)

Measures the proportion of the actual positives that are correctly classified. The True Positive Rate (TPR) for defining the accuracy of categorisation findings is identified by Sensitivity, also known as Recall. The ratio of correctly detected true positives to the total of true positives + false negatives is assessed. It gauges how frequently a classifier classifies a favourable outcome in the correct category. However, when the positive class is more significant (an unbalanced dataset), and the precise identification of positive samples is less crucial, the Sensitivity/Recall should not be employed.

$$Sensitivity = \frac{TP}{TP+FN} \tag{3.18}$$

Negative Predictive Value (NPV)

It is a measure of the proportion of correctly classified negative cases.

$$NPV = \frac{TN}{TN+FN} \tag{3.19}$$

Specificity (TNR)

Also known as True Negative Rate (TNR), it measures the proportion of the actual negatives that are correctly classified. The capacity to recognise a real negative rate is known as Specificity. It calculates the percentage of accurately determined true negatives relative to the total of True Negatives (TNs) and false positives (FPs). Therefore, 1 - Specificity equals the False Positive Rate (FPR). It gauges how frequently a classifier classifies a bad outcome accurately. Nevertheless, because it only considers one class, it could be more objective against that class.

$$Specificity = \frac{TN}{FP+TN} \tag{3.20}$$

F_1 - score or F - measure

The F1 score is a metric used to evaluate the performance of a classification model, particularly when dealing with imbalanced datasets. It combines both precision and recall into a single metric. Precision is the ratio of true positive predictions to the total number of positive predictions (true positives + false positives), while recall is the ratio of true positive predictions to the total number of actual positives (true positives + false negatives).

The F1 score is calculated as the harmonic mean of precision and recall, and it is given by the formula:

$$F = 2 \cdot \frac{Precision \cdot Sensitivity}{Precision + Sensitivity} = \frac{TP}{TP + \frac{1}{2} \cdot (FP+FN)} \tag{3.21}$$

The F1 score ranges from 0 to 1, where 1 is the best possible score, indicating perfect precision and recall, and 0 is the worst score. It's a useful metric when you want to balance both false positives and false negatives in your classification model evaluation.

G-mean

is formulated as the geometric mean of Precision and Recall.

$$G = \sqrt{Precision \cdot Sensitivity} \tag{3.22}$$

Cohen's Kappa Statistics (κ)

For qualitative (categorical) items, Cohen's kappa coefficient (κ) is a statistic used to assess inter-rater reliability. It is a metric often used to assess the agreement between two raters. Furthermore, it can be used to assess the performance of a classification model.

Kappa Statistic measures how well two nominal scales agree. This index gauges how closely a genuine class matches a classifier's output. A perfect agreement is a 1, while a chance agreement is a 0. Cohen's kappa provides the theoretical probability level of a classifier. This metric accurately assesses the classifier. Even with high accuracy values, the confusion matrix would be meaningless if k were to have a low value. Due to its usage of the whole confusion matrix, this coefficient provides more details than simply percentages of relationships. However, the correct interpretation of this coefficient is required. For a minimally acceptable degree of agreement, indicating the bias and prevalence of the k value and assessing the significance is essential.

Cohen's kappa is defined as the equation 3.23 [102]:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{3.23}$$

Where P_o is the model's overall accuracy and P_e is the measure of the agreement between the model predictions and the actual class values as if happening by chance.

In a binary classification problem, P_e is the product of P_{e1}, which represents the likelihood that predictions will coincide by chance with actual values from class 1 ("good"), and P_{e2}, which represents the likelihood that predictions will coincide by chance with actual values from class 2 ("bad"). These probabilities, P_{e1} and P_{e2}, are derived by multiplying the proportion of the actual class and the proportion of the expected class under the assumption that the two classifiers - model predictions and actual class values are independent.

$$P_e = P_{e1} + P_{e2} = P_{e1,actual} \times P_{e1,pred} + P_{e2,actual} \times P_{e2,pred} \tag{3.24}$$

Cohen's kappa, which counts the number of predictions the classifier produces that cannot be explained by a random guess, almost eliminates the likelihood of the classifier and a

Table 3.2 Interpretation of Kappa value

κ	**Interpretation**
< 0	No agreement
0 - 0.20	Slight
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Substantial
0.81 - 1.0	Perfect

random guess agreeing. Cohen's kappa also attempts to eliminate the assessment bias by accounting for the accurate categorisation made by chance.

Cohen's Kappa formula may be stated as follows in the conventional 2x2 confusion matrix used in machine learning and statistics to evaluate binary classifications:

$$\kappa_{binary} = \frac{2 \times (TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)} \tag{3.25}$$

Interrater reliability measures the degree to which different data collectors (raters) give the same variable the same score. The following scale is used to interpret Kappa values [103], shown in Table 3.2.

Receiver Operating Characteristic (ROC) Curve - Area Under the Curve (AUC)

A receiver operating characteristic curve (ROC) graph shows how well a model performs across all classification thresholds. Two parameters are shown in Figure 3.18.

The Sensitivity plot of the ROC curve is a function of the false positive rate. The area under the ROC curve measures how effectively a parameter can discriminate between a genuinely positive and a true negative. It indicates how well the classifier performs at various degrees of relevance. ROC is not advised when the negative class is smaller but more significant. If the positive class is more remarkable in an unbalanced dataset, the Precision and Recall will primarily represent the ability to predict the positive class.

- True Positive Rate (TPR), Sensitivity, or $S_{accuracy}$ based on Equation 3.12
- False Positive Rate(FPR), $(1 - Specificity)$, or $R_{accuracy}$ based on Equation 3.13

The TPR vs FPR axes at various classification thresholds are plotted on a curve and constitute the ROC space representing the x-axis and y-axis, respectively. This graphic demonstrates the relative trade-offs between real positive benefits and false favourable costs; as the classification threshold is lowered, more items are classified as positive, increasing

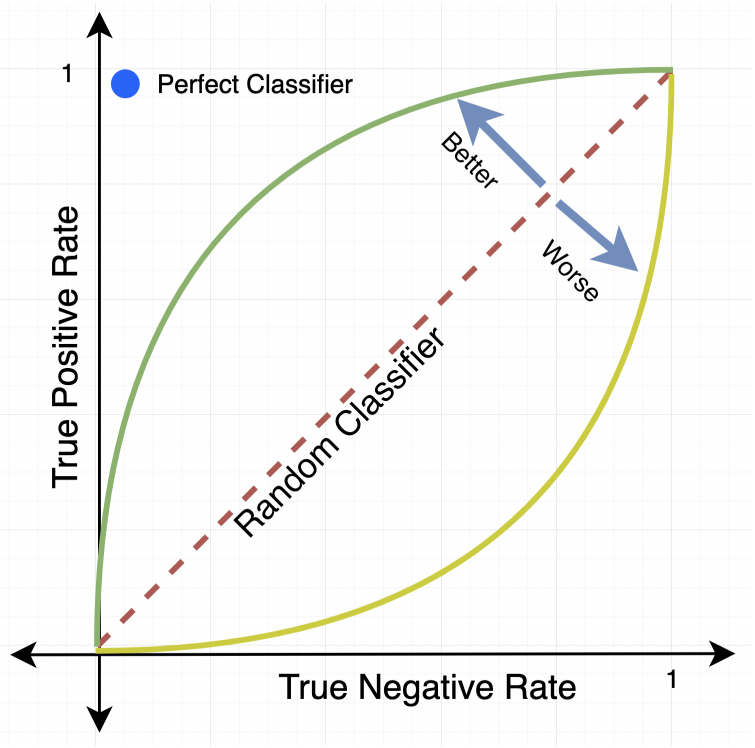


Fig. 3.17 ROC curve.

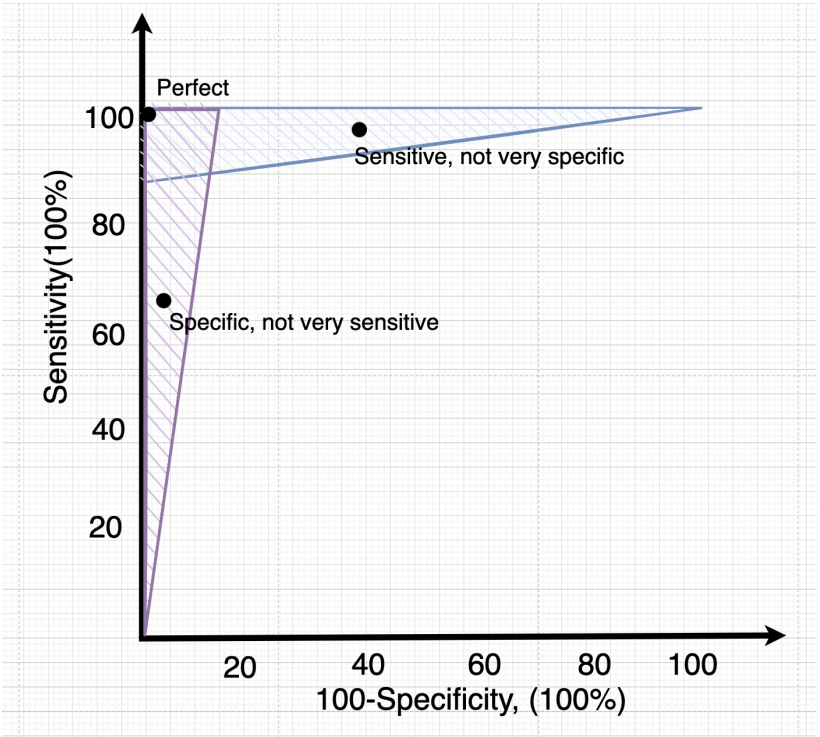


Fig. 3.18 ROC curve interpretation.

both False Positives and True Positives. A prediction of the classifier is represented as a point in the ROC space. A perfect classification would result in a point in the upper left corner of the ROC space, or coordinate (0, 1), with 100% Sensitivity (no false negatives) and 100% Specificity (no false positives). A classifier that expects all instances to be harmful is represented by the point (0, 0), whereas a classifier that predicts all cases to be positive is represented by the point (1, 1). The ROC space is divided by a "Random Classifier" diagonal. Generally, points above the diagonal indicate good classification results, whereas points below the line indicate bad ones. We might assess a logistic regression model several times with different classification criteria to compute the points on a ROC curve -that serves as a basis to determine the improvement in classification problems. The more closes to perfect classification, the better method to be used.

Other Performance Metrics

In machine learning and statistical inference, including Biostatistics, model selection is frequently done using statistical performance metrics as listed in Appendix Table D.2 and D.1. The model that performs the best in terms of a chosen performance criterion among those trained with various sets of hyper-parameters and parameters is appropriately chosen. For utilising the new model to classify untested data, it should increase its overall predictive power by evaluating its performance using several criteria. When a machine learning model is applied to new data, failing to properly evaluate it using a variety of performance measures and relying simply on accuracy might result in inaccurate predictions. With supervised models, always consider the concept of "cross-validation" as predicted values should ideally originate from unseen training sets to avoid overestimating the prediction performance [104].

Here is the list of Biostatistics Performance Metrics used in classification aside from the list given in the above section, some of the technical description is in the Appendix D.0.1 and D.0.2:

1. Error Rate ($Error_{rate}$)
2. Balanced Accuracy (BA)
3. F-score (F_1)
4. Geometric Mean (G_{mean})
5. Matthew's Correlation Coefficient (MCC)
6. Fowles-Mallows Index (FM)
7. Informedness (BM)
8. Positive Likelihood Ratio (LPR)
9. Negative Likelihood Ratio (LNR)
10. Diagnostic Odds Ratio (DOR)

11. False Positive Rate (FPR)
12. False Negative Rate (FNR)
13. False Omission Rate (FOR)
14. Prevalence (Prev)
15. Prevalence Threshold (PrevT)
16. Critical Success Index (CSI / TS)
17. Markedness (MK)

There are many classification performance metrics, and selecting the few important metrics is enough based on the optimisation wanted to interpret. **But for this scenario, analysing the robustness and vulnerability of the given process, it needs to consider all the classification metrics gathered in the literature and interpret it meticulously.** So, it can point down all the strengths and weaknesses of the said process. The function sub-program to compute the classification performance metrics in Python as shown in Appendix A.1.

3.6 Bootstrapping in Artificial Neural Networks (ANNs)

Bootstrapping statistics is a form of hypothesis testing that involves resampling a single data set to create a multitude of simulated samples with repetition. These samples are used for hypothesis testing, standard errors, and confidence interval calculations [105]. Compared to the conventional method, it enables us to get a more precise sample from a smaller data collection. The benefits of bootstrapping are that it is a convenient way to avoid the cost of repeating the experiment to get other groups of sampled data, and it is an easy way to derive estimates of accuracy, standard errors and confidence intervals.

After the training of the ANN, the samples are randomly selected in the dataset to test with n^{th} repetition, for this case, 2000 repetitions, and it is regardless of if the sample is repeatedly selected and it is large enough to justify using the resampling technique to generalise the PCA-SRP in ANN Model. Each group of samples passed through trained ANN and tested the dataset, which computed the accuracy or mean square error (MSE). When all the accuracies or MSEs are accumulated, it will visualise through a graph, which commonly resulting a normal distribution curve as shown in Figure 3.19.

It highlights the attributes of the Central Limit Theorem in describing the spread of deviation and normal distribution of averaging statistical estimators like Mean Square Error (MSE) and model accuracy of ANN Bootstrapping as shown in Figure 3.20.

The model accuracy of the ANN Bootstrapping is the best avenue to interpret and discuss since the whole discussion is the improvement of the incorporated process (PCA-SRP) in

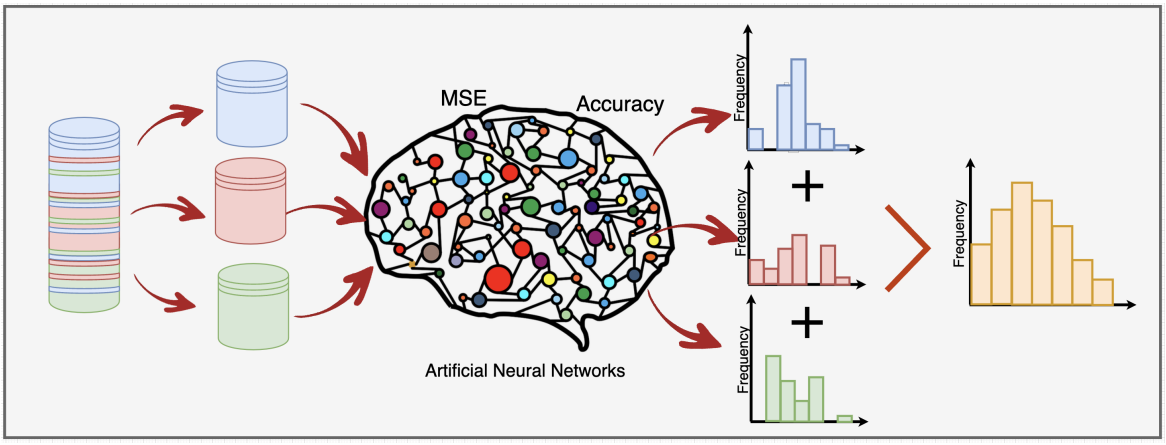


Fig. 3.19 ANN Bootstrapping

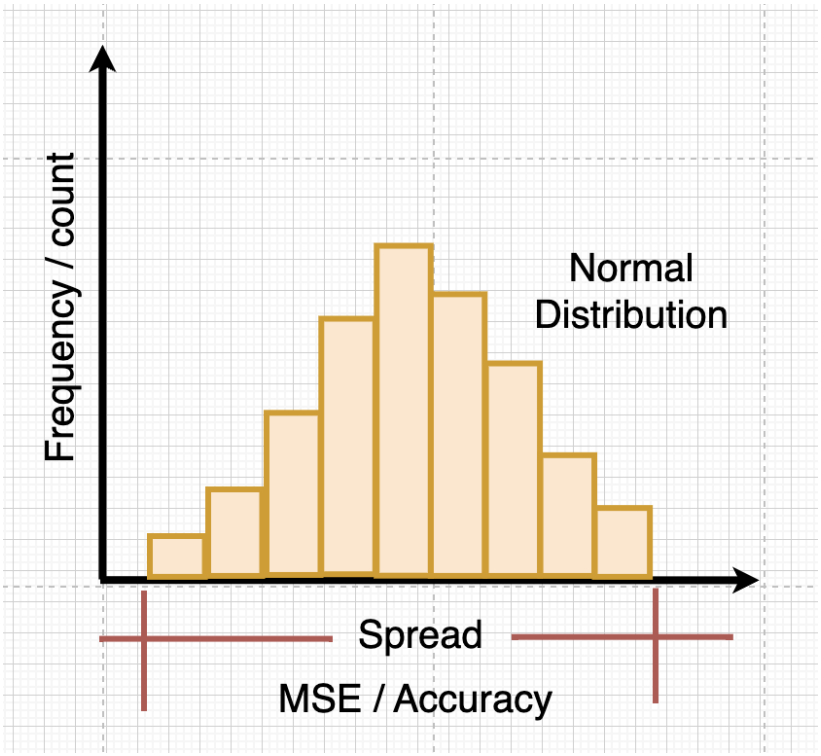


Fig. 3.20 Bootstrapping - Histogram

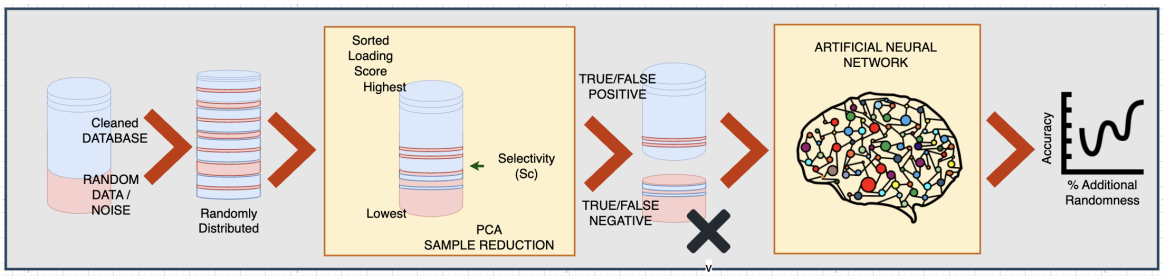


Fig. 3.21 Injected Additional Random Samples

the basic Feed Forward ANN Model rather than Mean Square Error (MSE). Comparing the central limit theorem that is normally distributed of its model accuracies from incorporated PCA-SRP to ANN Model through interpretation of its skewness, shifting, spreads, and amplitude are essential to this discussion. The Bootstrapping Python subprogram is in Appendix A.2, A.2, and A.2 for training of Artificial Neural Network, Bootstrapping using model accuracy or Mean Square Error (MSE), and visualization comparison between PCA-SRP with ANN and ANN only, respectively.

3.7 Effect of Model Accuracy by Injected Additional Random Samples

This validation testing aims to determine the behaviour of the Model Accuracy while increasing the randomness in the dataset with different Selectivity (Sc). The randomness is classified within the range on normal distribution of the dataset. It will show the how PCA-SRP response shown in the Figure 3.21.

3.8 Conclusions and Contributions

This dissertation presents several key points and contributions to the field of data cleansing and machine learning, particularly through the introduction and application of the Principal Component Analysis - Sample Reduction Process (PCA-SRP). The significant conclusion and contributions of this work are summarized below:

1. The introduction of the Sample Reduction Process using Principal Component Analysis (PCA) as an effective data cleansing technique. This process leverages the PCA framework to identify and eliminate less predictive samples, enhancing the overall quality of the dataset used for training machine learning models.

2. Demonstrating the potential of eigenvectors (v) from PCA as loading scores to quantify the predictive value of each sample. This approach provides a quantitative basis for sample selection, which is crucial for improving model performance.

3. The integration of PCA-SRP into basic Artificial Neural Network (ANN) models for classification problems. This technique, implemented prior to the training process, complements existing literature and research by demonstrating its effectiveness in simplifying and improving the training of more complex ANN models.

4. Introducing the concept of Selectivity (Sc) as a limiting threshold to determine which samples to retain or remove. This concept utilizes the Sensitivity vs Specificity testing procedure, providing a systematic way to enhance data quality.

5. Comprehensive testing of the proposed system using various classification performance metrics. This testing process helps in interpreting the robustness and limitations of the PCA-SRP technique, ensuring its reliability in practical applications.

6. Incorporating the basic ANN model into the resampling technique of bootstrapping for 2000 repetitions, using model accuracies to evaluate the effectiveness of the proposed data cleaning technique. This method assesses the impact of PCA-SRP by comparing the shifting skewness and spreads of the model's performance.

7. Figure 3.4 illustrates the process of the mathematical model applied to two kinds of datasets: general data and signal data. It also shows the evaluation and testing of performance metrics, bootstrapping, and the effect of injecting additional linear (from other datasets) and non-linear (random) datasets.

In summary, the implementation of PCA-SRP in data cleansing significantly enhances the predictive quality of training datasets for ANN models. This work not only provides a robust framework for data cleansing but also sets the stage for further research in optimizing data preprocessing techniques for various machine learning applications. The flexibility and effectiveness of PCA-SRP in improving model accuracy and robustness highlight its potential for broader applications in different fields of science and technology.

Chapter 4

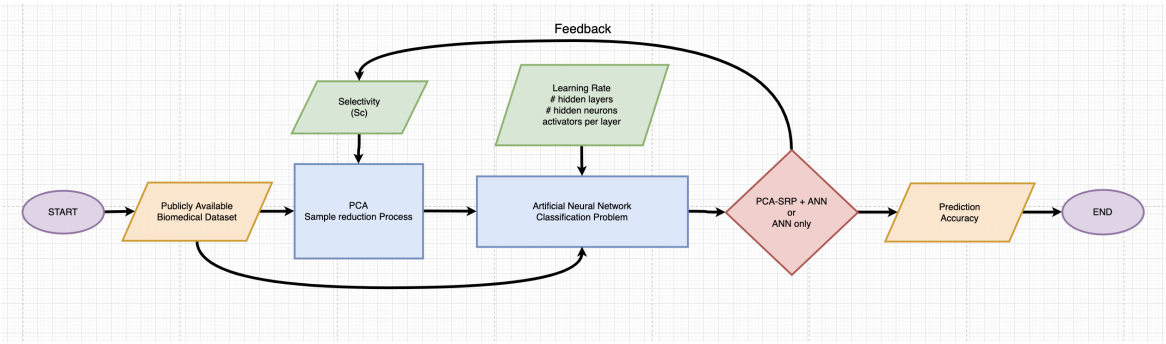
Principal Component Analysis - Sample Reduction Process (PCA-SRP) in Physiological Datasets

4.0.1 PCA-SRP Implementation in ANN

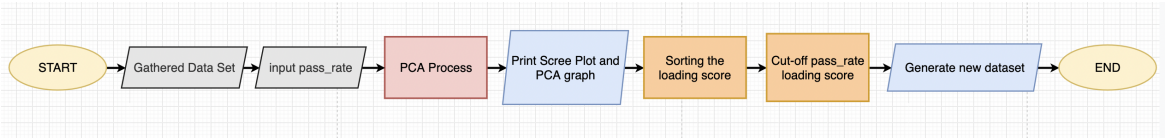
The proposed framework addresses the limitations of the ANN in processing multidimensional datasets through the use of PCA with the sample reduction process (PCA-SRP) [84]. PCA is used to analyse the multidimensional data and if a significant relationship exists between the features of a dataset. It is arranged from the most significant samples to the least to be visualised and put the multidimensional data into perspective. It focuses on implementing dimension reduction by removing data from the multidimensional dataset feature columns that do not have high inter-feature covariance; however, the proposed framework using the loading scores can dissect all the samples or rows in the multidimensional dataset. Figure 4.1a provides a graphical description of the proposed PCA-SRP-based ANN solution.

The PCA-SRP Python subprogram as shown in Appendix A - **Subprogram - PCA - SRP** generates a new set of multidimensional data with fewer samples based on the screen plot to identify the most significant samples. It is used in the Python ANN program as an input to ensure correct classification.

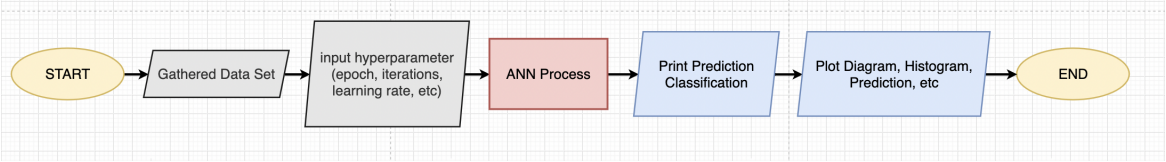
4.1 PCA-SRP and ANN Methodology



(a) Conceptual flowchart of the proposed approach.



(b) Detailed breakdown of the steps in PCA.



(c) Detailed breakdown of the steps in ANN.

Fig. 4.1 Procedural Conceptual Framework

Figures 4.1b and 4.1c show the algorithm used in Python to implement and analyse the concept. The implementation starts by extracting the data in .csv format and injecting it into the PCA-SRP process. Principal components are extracted through the abovementioned threshold based on the input dataset, resulting in a new set of multidimensional datasets used in ANN.

The proposed PCA-SRP utilised in the basic Feed Forward ANN model consists of two hidden layers consisting of 32 neurons and 16 neurons, respectively. The learning rate was set at 0.1 and trained for 100 epochs. The input parameters in PCA-SRP-based ANN are shown in Table 4.1 and justified in Chapter 3.2. Then, the comparative performance of the Accuracy of the PCA-SRP-based ANN against a model trained using ANN alone on different datasets.

The ANN model is prescribed to be a basic Feed Forward ANN with 32 and 16 neurons, respectively, **using ReLU and SoftMax activators**, RELU is conventionally used as an activation function for the hidden layers in a deep neural network, and Softmax as output

Table 4.1 PCA-SRP and ANN parameters.

PCA- SRP and ANN Parameters	**Values**
Most significant sample size	$> Sc \times (PCA_{SRP}(M)_{max})$
Learning rate	0.1%
Epochs	100
Number of ANN neurons	2 hidden layers (32 and 16, respectively)
Activation functions used	ReLU in hidden layers SoftMax in final layer

Table 4.2 Datasets used and their metadata.

Datasets	**Number of Dimensions**	**Sample Size**	**Target Classification**
Heart Disease	14	301	2
Gender Voice Recognition	21	3167	2
Breast Cancer	31	568	2
Cancer Patients	24	1098	3

activation [85]. A **testing size of 20%, training size of 80%**, it is described that the $p \approx 0.8$ is empirically the best division into the training and the testing test, due to the decreasing $p(1 \dot{-} p)$ error measurement at $p \leq 0.8$ [86]. Moreover, a **learning rate of 10%**, for this reason, the loss starts decreasing significantly between LR 0.001 and 0.1 as shown in the Figure 4.2 [87] . One hundred (100) epochs are chosen to visualise the gradual changes and identify the behaviour of the performance metrics. Furthermore, $Sc = 0.98$ or PC_1 of the highest loading score will be used in PCA analysis.

4.2 Multidimensional Physiological Datasets

The multidimensional, open-access, and publicly available physiological datasets for classification problems are tested under two layers of the basic Feed Forward Artificial Neural Networks (ANNs) Model as described in Table 4.1. Noise and random data are added based on a standard distribution added to the original dataset before training to simulate noisy data samples. Figure 4.3 shows how a given multidimensional dataset is pre-processed and used in the proposed PCA-SRP approach. Each multidimensional dataset or corpora has selected

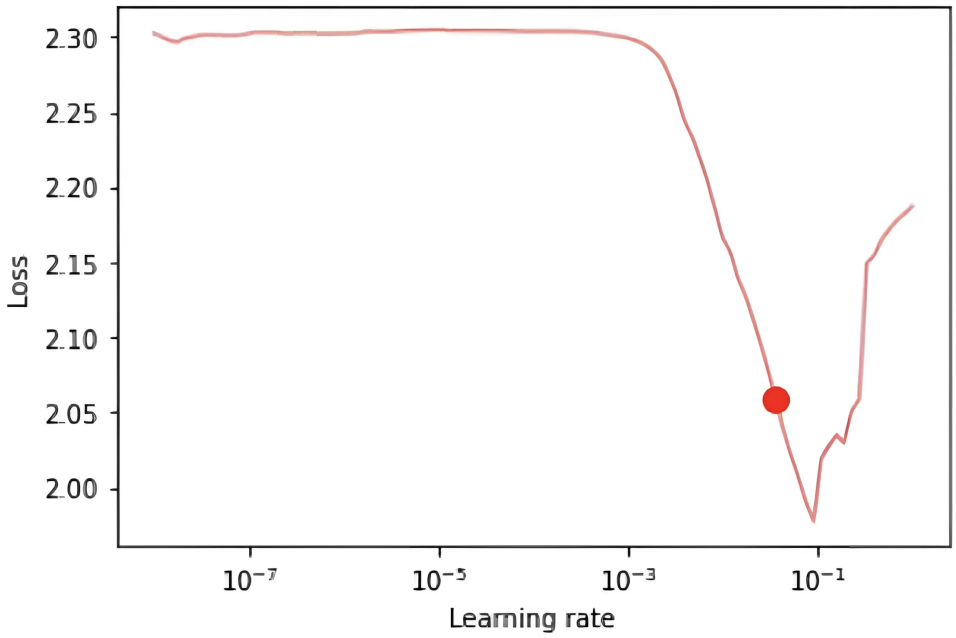


Fig. 4.2 Typical Loss vs Learning Rate (LR) Behaviour.

the variation of its dimensions, primarily sample sizes, and the number of classifications shown in Table 4.2, with a reasonable amount of noise and random samples.

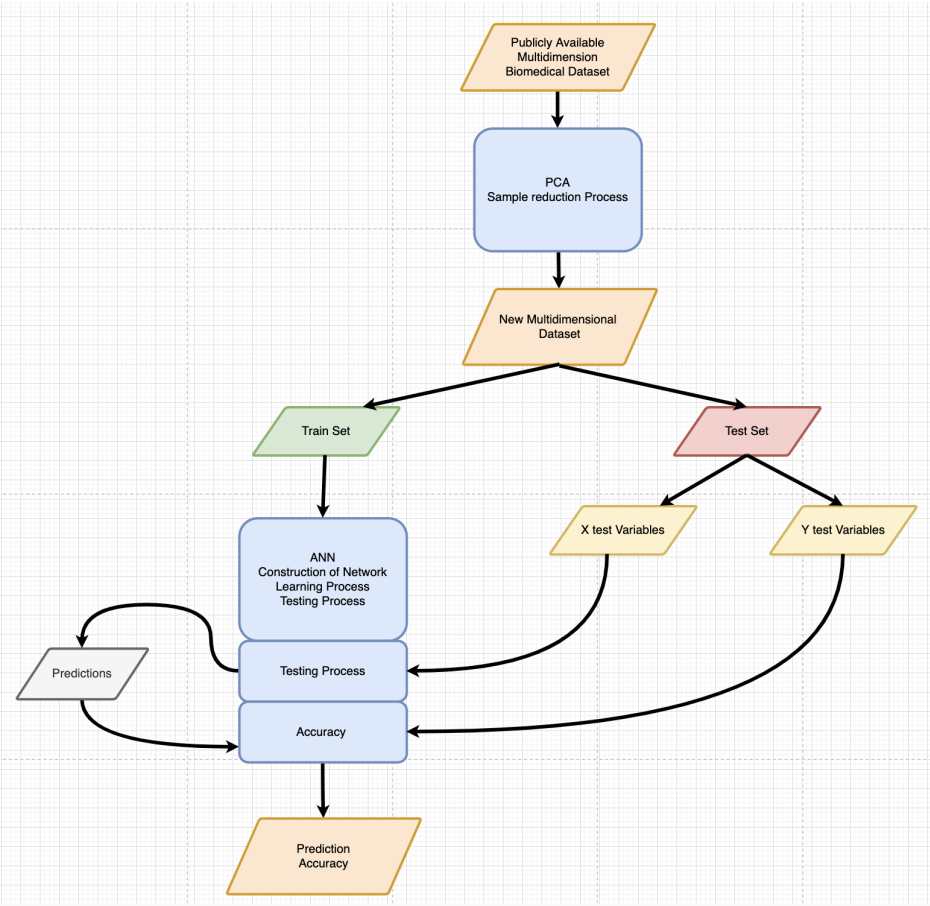


Fig. 4.3 Pre-processing an application of multidimensional datasets in the proposed approach.

The different multidimensional datasets are shown in Table 4.2 and are acquired from different scientific and medical laboratories [22–24, 106, 25]. Specifically, it is tested through the procedure by different samples and dimensions, showing that the Python implementation works with various dataset sizes. Moreover, this particular dataset is a benchmark of many research, particularly in predicting certain diseases and recognising some attributes given the different features; it is also subject to ample classification of random function approximation through many complicated ANN models; this concept will supplement a data cleansing technique for the pre-processing procedure before the training of complicated ANN Models by incorporating this new concept in their study as discussed in Chapter 3.2.

Visualisation is one of the avenues to comprehend the dataset; through it and proper evaluation of the performance metrics will identify the working functional attributes. So, this chapter mains to understand the given datasets - its scatteredness and correlation with other samples by quantifying its sample scores and the behaviour ANN Model by incorporating the PCA-SRP concept in Model Accuracies and its resilience in injected additional randomness.

Furthermore, its response to Receiver Operating Characteristic Curve - Area under the Curve (ROC-AUC), which will be tested under the Figure **??** to evaluate the Sensitivity and Specificity of the datasets through PCA-SRP.

4.2.1 Heart Disease Dataset

There are 76 features in this open access database, but all available studies mention using a subset of 14. The Cleveland dataset, in specific, is the only one that ML scientists have used to date. The field "target" relates to the patient's presence of heart disease.

Experimenting with the Cleveland patients dataset, which is concentrated on simply attempting to distinguish presence (values 1) from absence (values 0) to find any other trends in heart data to predict certain cardiovascular events or find any clear indications of heart health [22]. The following parameters are evaluated for each patient based on the following:

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholesterol in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise-induced angina
- old peak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) coloured by fluoroscopy
- that: 3 = normal; 6 = fixed defect; 7 = reversible defect

Figures 4.4 and Table 4.3 show the result of the scree plot based on the values indicating the PCA value distributions of the features in the dataset and the *Sc*-threshold cut-off limit, respectively. Based on the values, 78.48% of the top samples were selected in the training of the ANN model for 98% Selectivity.

Table 4.3 Sorted loading scores of heart disease dataset.

Sample ID	Loading Scores
278	0.057987
226	0.057981
2	0.057976

45	0.057968
42	0.057961
...	...
147	0.003164
144	0.001178
138	0.000718
139	0.000605
327	0.000470

4.2.2 Gender Voice Recognition Dataset

The gender voice recognition dataset is based on acoustic properties of the voice and speech to identify a voice as male or female. The dataset comprises 3167 recorded voice samples collected from male and female speakers. The voice samples are pre-processed by acoustic analysis in R using the Seewave and TuneR packages, with an analysed frequency range of 0–280 Hz (human vocal range) [23].

Each voice's acoustic features are evaluated based on the following:

- mean frequency in kHz (meanfreq)
- standard deviation of frequency (sd)
- median frequency in kHz (median)
- first quantile in kHz (Q25)
- third quantile in kHz (Q75)
- nterquantile range in kHz (IQR)
- skewness (skew)
- kurtosis (kurt)
- spectral entropy (sp.ent)
- spectral flatness (sfm)
- mode frequency (mode)
- frequency centroid (centroid)
- average of fundamental frequency across acoustic signal (meanfun)
- minimum fundamental frequency across acoustic signal (minfun)
- maximum fundamental frequency across acoustic signal (maxfun)
- average of dominant frequency across acoustic signal (meandom)
- minimum of dominant frequency across acoustic signal (mindom)
- maximum of dominant frequency across acoustic signal (maxdom)

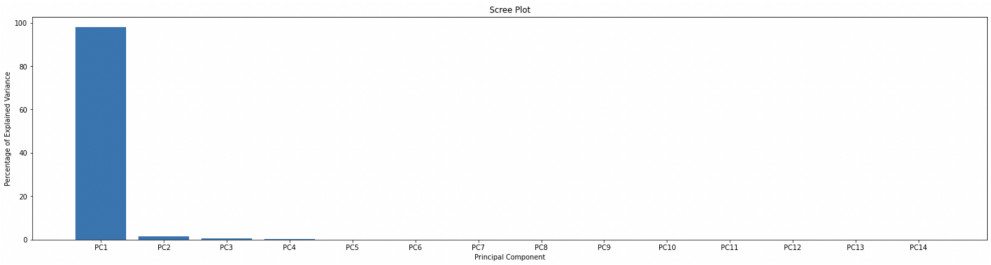


Fig. 4.4 Scree plot of the heart disease dataset.

- range of dominant frequency across acoustic signal (dfrange)
- modulation index. Computed as the gathered absolute change among adjacent measurements of fundamental frequencies divided by the frequency range (modindx)
- label: male or female

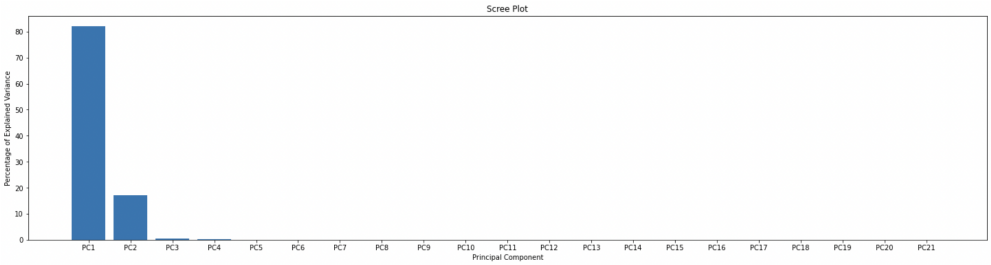


Fig. 4.5 Scree plot of gender voice recognition dataset.

Figures 4.5 and Table 4.4 show the threshold cut-off limit and the result of the scree plot showing the PCA distributions of the features in the dataset, respectively. In training the ANN model, the proposed method selected 23.82 % of the top samples from the dataset.

Table 4.4 Sorted loading scores of gender voice recognition dataset.

Sample ID	Loading Scores
2462	0.019580
3121	0.019578
2729	0.019575
2022	0.019574
2711	0.019571
...	...
1601	0.000244
1626	0.000243
3259	0.000181

1641	0.000168
3267	0.000068

4.2.3 Breast Cancer Classification Dataset

The computed dataset consists of a series of digitised images of a mass breast fine needle aspirate (FNA). They describe features of the cell nuclei present in the 3-dimensional image as described in [24].

The following data on the UCI Machine Learning Repository:

- radius
- texture (standard deviation)
- perimeter
- area
- smoothness
- compactness
- concavity
- concave points
- symmetry
- fractal dimension

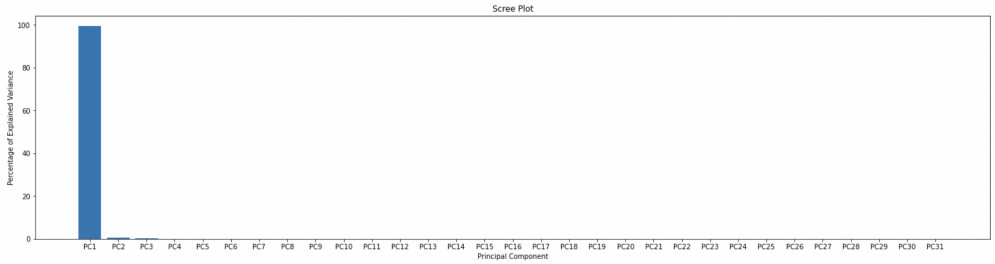


Fig. 4.6 Scree plot of breast cancer classification.

Figure 4.6 and Table 4.5 show the threshold cut-off limit and the result of the scree plot showing the PCA distributions of the features in the dataset, respectively. The selection of the top eigenvalues in the samples obtains 93.67% of the original dataset.

Table 4.5 Sorted loading scores of breast cancer classification dataset.

Sample ID	Loading Scores
466	0.041997
545	0.041996

224	0.041996
438	0.041995
93	0.041995
...	...
580	0.001718
578	0.001383
594	0.000928
576	0.000828
595	0.000477

4.2.4 Lung Cancer Patients Dataset

The data comprise information about hundreds of cancer patients and their lifestyles. Based on the cancer patients dataset consists of three classes (low, medium and high severity) based on the cancer patients dataset [25].

Each lung cancer patient's features are evaluated in accordance of:

- age
- gender
- air pollution
- alcohol intake
- dust allergy
- occupational hazards
- genetic risk
- chronic lung disease
- balanced diet
- obesity
- smoking
- passive smoking
- chest pain
- coughing of blood
- fatigue
- weight loss
- shortness of breath
- wheezing
- swallowing difficulty

- clubbing of fingernails
- frequent cold
- dry cough
- snoring
- severity of lung cancer disease

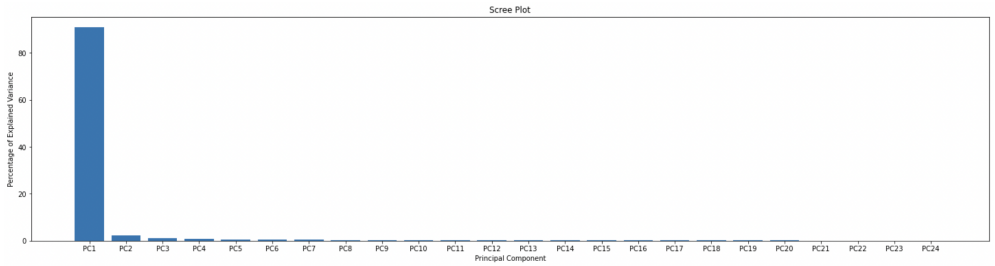


Fig. 4.7 Scree plot of cancer patients dataset.

Figures 4.7 and Table 4.6 provide the threshold cut-off limit. And the graphical description of the PCA distributions amongst the features in the dataset, respectively. The selection of high eigenvalues features resulted in 45.41 % of the samples in the original dataset used for the model's training.

Table 4.6 Sorted loading scores of cancer patients dataset.

Sample ID	Loading Scores
900	0.031442
776	0.031442
332	0.031442
443	0.031442
110	0.031442
...	...
1082	0.006573
1061	0.006205
1043	0.004993
816	0.001869
98	0.000278

Table 4.7 Datasets sample status after PCA—sample reduction process.

Dataset	Samples Used	%Passrate
Heart Disease	259/330	78.48
Gender Voice Recognition	786/3300	23.82
Breast Cancer	562/600	93.67
Cancer Patients	499/1099	45.41

4.3 PCA - SRP Results in Physiological Datasets

Upon acquiring the given datasets, it is injected through the noise and random samples. By definition, noise is the unwanted small form of energy or samples; on the other hand, random samples are the unconscious and unspecified values within the range of expected values and "seemingly" good data.

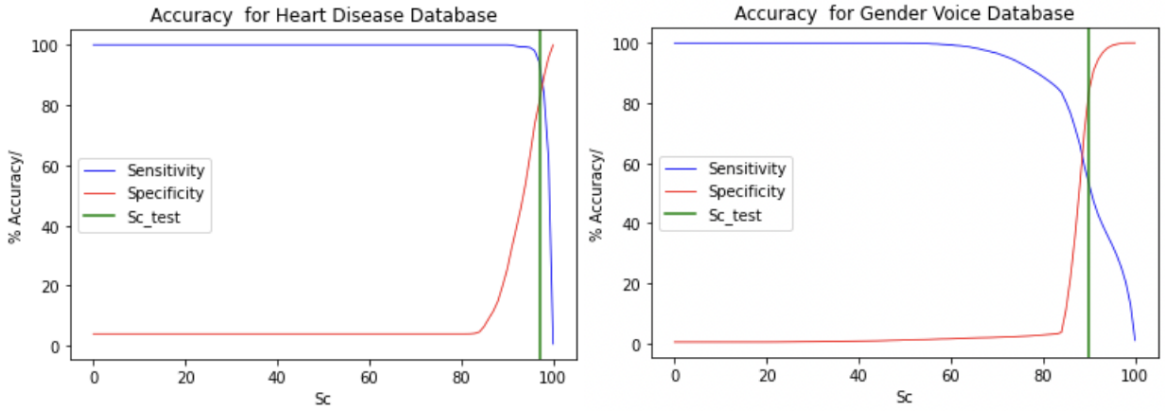
The datasets with noise and random samples are processed using PCA-SRP with 98% Selectivity (*Sc*), shown in Table 4.7, then it is subjected to following tests:

- PCA-SRP + ANN accuracy testing to compare the validation model accuracy with and without the PCA-SRP in an ANN.
- Sensitivity vs specificity testing is a diagnostic test to find the approximate *Sc* range values (Sc_{test}).
- Receiver operating characteristic (ROC) curve testing using methodology PCA-SRP in different physiological datasets in terms of organisation and classification of samples. Moreover, ROC curves also provide a practical evaluation of machine learning techniques.
- Accuracy vs additional random samples testing is a diagnostic test responding to the sudden increase in noise and random sample spikes.

4.3.1 Sensitivity vs Specificity Testing

Sensitivity measures how many true positives remain in the *S* set, and it is described as a sudden dip as the *Sc* increases. While **Specificity** measures how many true negatives remain in the removed *R* set and increases while *Sc* increases. The evaluation is performed in added **10% randomness**, so it will classify the good samples into random ones in identifying the True Positive,(TP) True Negative(TN), False Positive(FP), and False Negative (FN).

The Figure 4.8 presented is the diagnostic testing of Sensitivity and Specificity Curve Line with $Sc_{rec} = PC_1$, which aims the identification of *Sc* and draw a generalisation out of the given physiological datasets.



(a) Heart disease dataset.

(b) Gender voice recognition dataset.

(c) Breast cancer classification dataset.

(d) Cancer patients dataset.

Fig. 4.8 Sensitivity vs specificity diagram.

Table 4.8 Sensitivity and Specificity Result.

Dataset	Sc_{rec}	**Sensitivity**	**Specificity**	Sc_{test}	**Sensitivity**	**Specificity**
Heart Disease	96	98.04	73.68	97	94.15	81.04
Gender Voice Recognition	80	88.75	22.87	90	53.08	84.103
Breast Cancer	96	99.47	57.64	99	94.51	81.84
Cancer Patients	88	93.83	25.86	95	81.8	39.1

When $Sc_{test} \geq Sc_{rec}$, wherein Sc_{test} will have more R samples will introduce in the system. Furthermore, if $Sc_{rec} \geq Sc_{test}$, as the result in Table 4.8 - there is no conclusive evidence that Sc_{test} is better than Sc_{rec}.

Note: Giving priority on the Sensitivity is more desirable because it is more S samples is part of the system than Specificity that removing more S samples in the system.

A high Sc value loses information and true positives in the S set but increases the Specificity of the dataset; however, a low value of Sc adds noise and random samples that yield fewer true negatives and increase Sensitivity. Careful adjustment of Sc will vouch for a good result as a cleaning agent in the system.

In ANN applications need high predictive samples [40], Equation 3.15 is still valid ($Sc_{rec} = PC_1$).

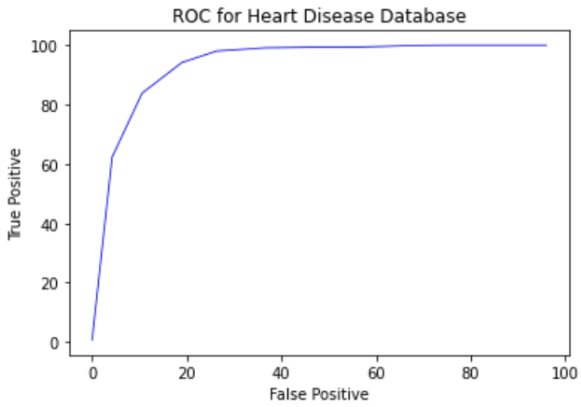
4.3.2 Receiver Operating Characteristic (ROC) Curve Testing

As observed in Figure 4.9, receiver operating characteristic (ROC) shows that the datasets have an organisation of the random samples. The larger the area under the curve (AUC), the better the classifier methodology for the True Positive Rate (Sensitivity) vs True Negative Rate (1— Specificity) diagram, as seen in Figure 3.18. Ideally, the objective is the perfect classifier; nevertheless, above the random classifier line would allow us to conclude that the methodology is acceptable.

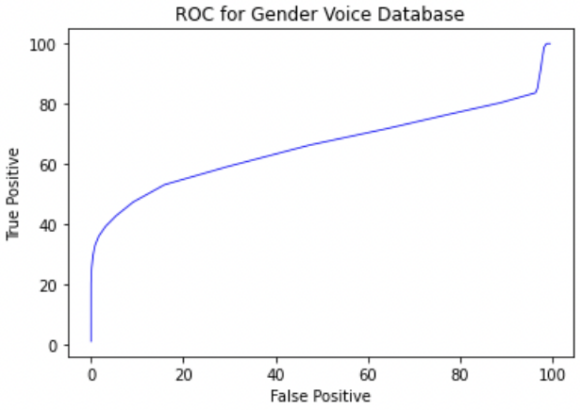
Given all the datasets, the cancer patients dataset has fewer AUC but still acceptable results.

4.3.3 PCA-SRP + ANN Comparison Accuracy Testing

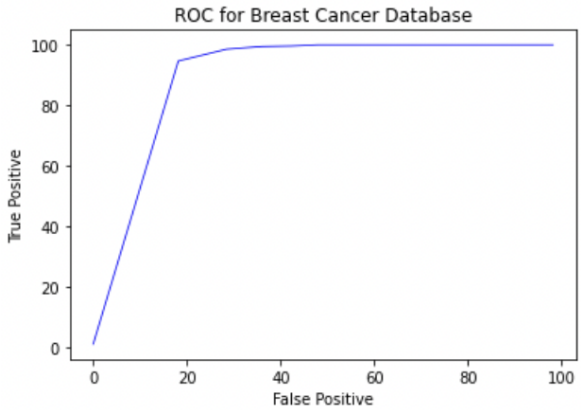
We compared the validity model accuracy using the PCA-SRP in ANN classification problem as suggested by the results presented in Figure **??** and determined its effect upon being subject to noise and random samples. Tables 4.10 and 4.11 display the validation accuracy using ANN +



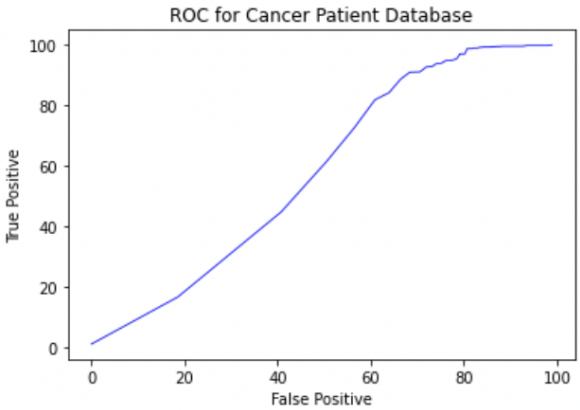
(a) Heart disease dataset.



(b) Gender voice recognition dataset.



(c) Breast cancer classification dataset.



(d) Cancer patients dataset.

Fig. 4.9 Receiver operating characteristic (ROC) curve.

Table 4.9 Area under the Dataset's Curve (AUC).

Dataset	**AUC**
Heart Disease	89.9
Gender Voice Recognition	84.87
Breast Cancer	90.05
Cancer Patients	84.139

PCA-SRP and ANN only, showing a significant increase as shown in Figure 4.10 for both noise and random samples.

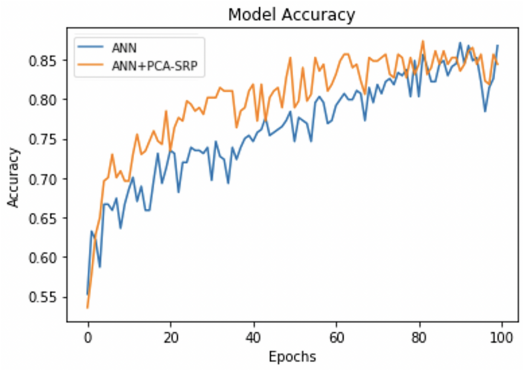
Table 4.10 Dataset noise sample accuracy.

Dataset	**ANN Only**	**ANN with SRP**	**% Increase**
Heart Disease	76.19	79.89	3.7
Gender Voice Recognition	90.6	93.29	2.69
Breast Cancer	90.49	92.36	1.86
Cancer Patients	79.82	83.27	3.45

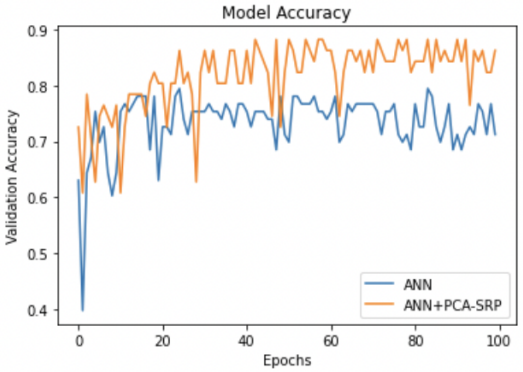
The difference between dataset accuracy subjected to noise and random samples is the definition of the accuracy curve line between ANN + PCA-SRP to ANN only, even though both significantly increase accuracy. The accuracy of datasets under the influence of random samples swings and deviates more and less defined in comparison with accuracy with noise samples.

Table 4.11 Dataset random sample accuracy.

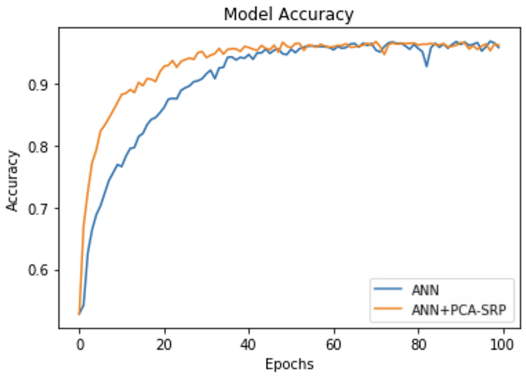
Dataset	**ANN Only**	**ANN with SRP**	**% Increase**
Heart Disease	73.69	81.76	8.07
Gender Voice Recognition	85.72	91.2	5.42
Breast Cancer	83.3	86.46	3.12
Cancer Patients	80.90	88.81	7.91



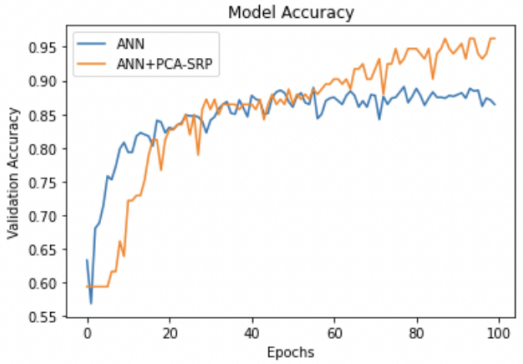
(a) Heart disease dataset with noise samples.



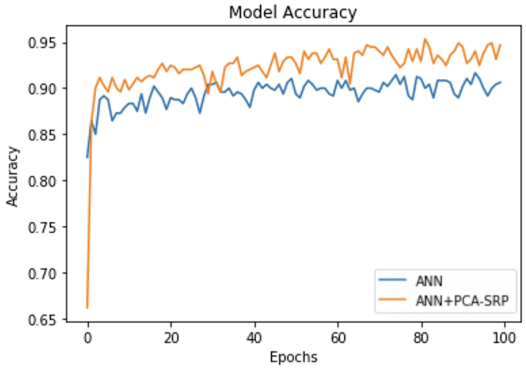
(b) Heart disease dataset with random samples.



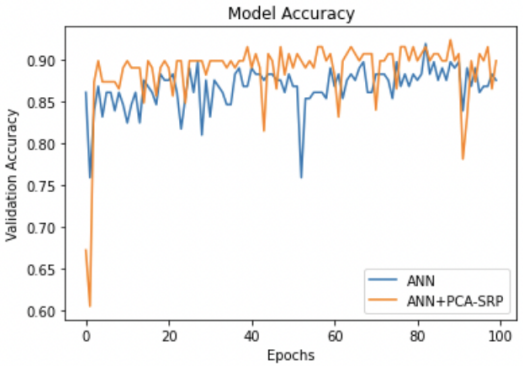
(c) Gender voice recognition dataset with noise samples.



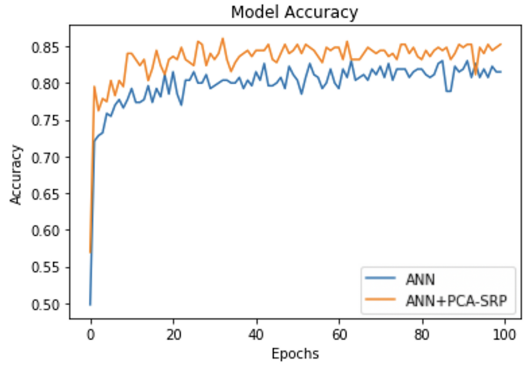
(d) Gender voice recognition dataset with random samples.



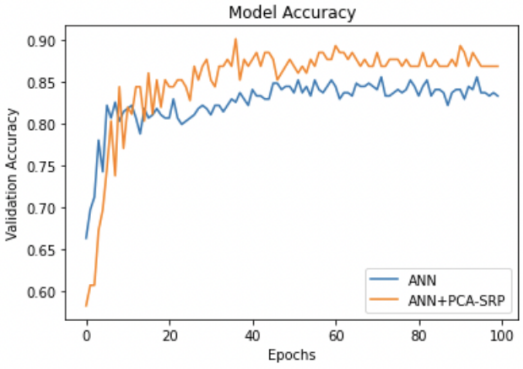
(e) Breast cancer classification dataset with noise samples.



(f) Breast cancer classification dataset in random samples.



(g) Cancer patients dataset with noise samples.



(h) Cancer patients dataset with random samples.

4.3.4 Model Accuracy vs Injected Additional Random Samples Testing

Since the cancer patients dataset shows the least-effective classifier, as shown in Figure 4.9, it was tested for its response to the injection of the sudden spike of additional random samples up to 100%.

This test is show the responsiveness of the PCA-SRP in increasing randomness of the data and its effect in model accuracy.Figure 4.11 presents the validation accuracy of both ANN + PCA(SRP) and ANN only to additional noises up to 150% with Selectivity (Sc) of 88% and 98%, respectively. It has also shown a reasonable increase in accuracy using 98% Selectivity (*Sc*) instead of 88%.

The high-valued data have been preserved and maintained their accuracy until a specific additional noise point. Nevertheless, the predictive power remained intact until that point, even though some data were lost.

The $ANN + PCA_{SRP}$ maintains the highest performance of cancer patient classifications with high *Sc* values. The methodology allows for a significant advantage by gradually slackening the decrease in classification problem accuracy over the sudden increase in noise in the system.

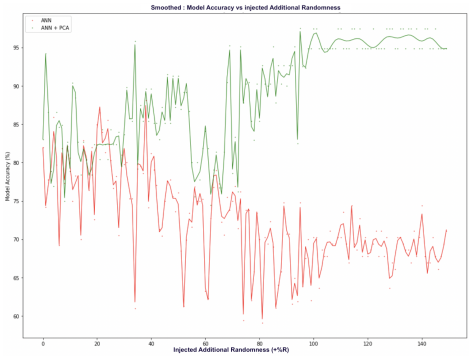
The ANN classification problem requires strong training sets, which requires a lot of high *Sc* while disregarding the low ones; based on the observation in both Figures 4.11 Sc_{ANN} is better as described in Equation 3.15

4.4 Contributions and Conclusions

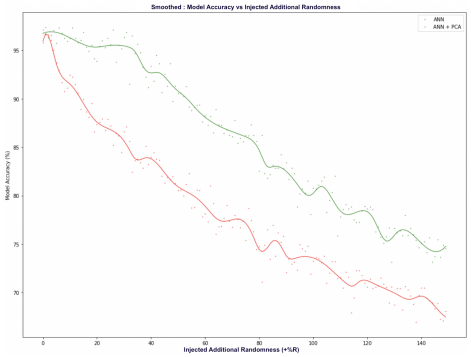
The material presented in this paper shows a significant improvement in the accuracy of an ANN in classification problems with the aid of the Principal Component Analysis—Sample Reduction Process (PCA-SRP). The ANN cast-off 10% of the learning rate, two (2) layers with 32 and 16 hidden neurons, respectively, ReLU activators in hidden layers and SoftMax in output activators with 100 epochs or iterations, as shown in Table 4.1 based on the PCA-SRP and ANN Python implementation program implemented on the gathered multidimensional datasets, namely heart disease, gender voice recognition, breast cancer classification and cancer patients dataset provided in Table 4.2. These datasets were then used to test the PCA-SRP + ANN accuracy comparison testing, Sensitivity vs Specificity testing, Receiver operating characteristic (ROC) curve testing and accuracy vs additional random samples testing; the results show significant improvements.

To aid the PCA-SRP, removing the dirty and imprecise dataset is achieved based on the results shown in Table 4.7, which allowed us to reduce the number of samples in the process and allowed for a significant increase in accuracy, as shown in Tables 4.10 and 4.11. Further,

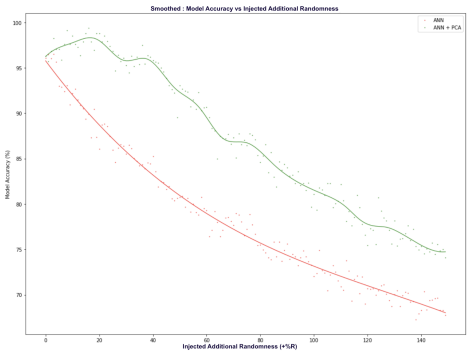
(a) Heart Disease dataset (Sc = 88%)



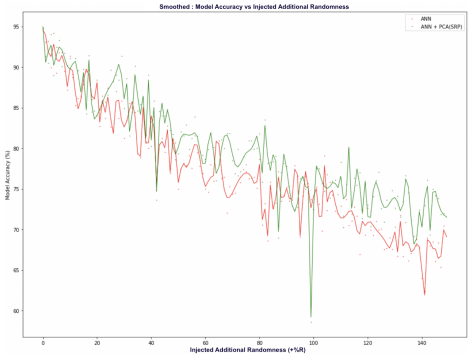
(b) Heart Disease dataset (Sc = 98%)



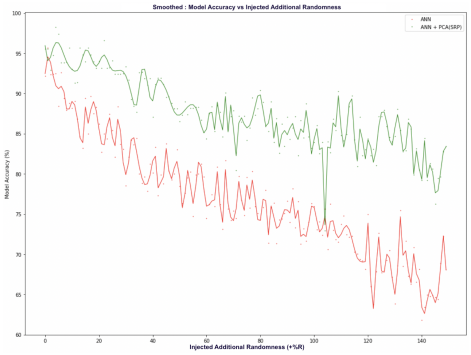
(c) Gender Voice dataset (Sc = 88%)



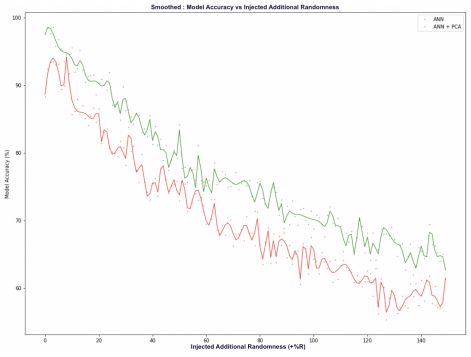
(d) Gender Voice dataset (Sc = 90%)



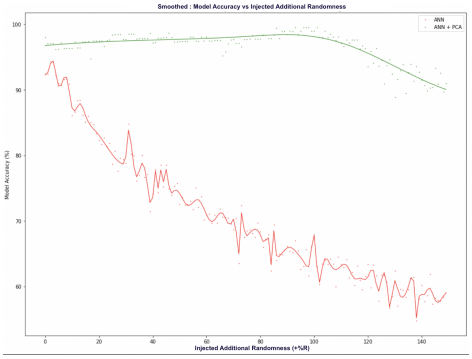
(e) Breast Cancer dataset (Sc = 88%)



(f) Breast Cancer dataset (Sc = 98%)



(g) Cancer Patient dataset (Sc = 88%)



(h) Cancer Patient dataset (Sc = 98%)

Fig. 4.11 ANN accuracies in additional (+)%Randomness Response

we also determined the recommended *Sc* range values for standard cleaning and the ANN classification problem.

Here is the following contributions of this said chapter:

1. Scree plot visualisation is essential in identifying the most highly explained variance samples.

2. The intersection between Sensitivity and Specificity lines is approximately balanced Selectivity *Sc*, which equals PC_1 of the PCA of explained variance. Moreover, it is the point wherein the balanced Sensitivity and Specificity have maximum acquired good samples and minimum acquired bad samples.

3. Using PCA-SRP in incorporating the filtered training set into the ANN model will help to increase the Model accuracy of the given datasets.

4. Also, the visualization through PCA-SRP in the event of injecting more randomness in the ANN Model, which the the response of the deteriorating model accuracies was slowing gradually.

Future research will further investigate the performance of massive physiological datasets and determine how to load them into PCA-SRP cleansing agents; one of the suggestions is loading through batch processing. Furthermore, an investigation into various field applications to explore incorporating the investigated cleaning techniques, such as real-time biomedical automation, image-based medical diagnosis classification and human thought processes [107–113] could be a desirable research avenue.

The proposed methodology can be applied to a wearable EEG or similar device in order to extract brainwave signal that seems non-deterministic, stochastic, nonstationary, noisy, non-linear, and non-unified reference base signal as well. in a given brain wave activity that limits and remove the unwanted signal in the brain and increase the accuracy of Neural Network classification model.

Chapter 5

Introduction and Previous Work on EEG Signal Acquisition and Analysis

Electroencephalography (EEG) is a non-invasive data-collection technology that monitors the brain's electrical activity along the scalp. Because EEG records the voltage variations caused by an ionic current within the brain's neurons, it may be used to assess some of the brain's intrinsic properties [114]. Monitoring the electroencephalographic signals of animals (rabbits' and monkeys' brains, in 1890) was one of the first explorations into brain-related processes [115]. German scientist and psychiatrist Hans Berger (1873–1941) captured the first human EEG signal in 1924. Since then, EEG has often been used for clinical purposes, and through time, the EEG has evolved into a helpful tool for identifying brain disorders (epileptic seizures, for instance) [116] [117]. There are several important questions concerning EEG, but afraid of the answer [118]. It is more complicated because the human brain is a complex non-linear system showing convoluted emergent properties, including consciousness, which the EEG signal describes as having 5 Ns in nature:

- Noisy,
- Non-deterministic and stochastic,
- Non-linear,
- Non-stationary, and
- Non-unified reference base signal.

Given the EEG undesired characteristics, the physiological variations between human brains' EEG readings may make it hard to identify which signal attributes and properties correspond to its different mental activities.

5.1 Brain - Computer Interface (BCI)

A thorough study of the brain's electrical and physiological properties and acquiring its parts' behaviour are avenues to understanding its peripherals, such as the limbs' motor action, mental imagery, and other mental activities. Nevertheless, they could be more adaptable and precise outside a controlled laboratory setting.

The development of new applications bridging brains with machines, known as Brain-Computer Interface (BCI) technology, was facilitated by recent advancements in neuroscience and engineering. The first BCI devices were created in the 1960s, most notably the implanted chip that could both transmit and receive electrical impulses from the brain [119]. This allowed the patient to walk around freely while being stimulated by radio waves in the brain. A few years later, studied the implementation of a straightforward, non-invasive BCI based on "visually evoked potentials" using scalp-recorded brain signals in people [120].

These studies paved the way for the creation of non-invasive BCI paradigms using neuroimaging methods like electroencephalography (EEG), magnetoencephalography (MEG), functional magnetic resonance imaging (fMRI), and functional near-infrared spectroscopy (fNIRS) [121]. Indeed, BCI enables controlling external devices with the brain, such as a computer, a robot, or an exoskeleton by converting the recorded neural activity into digital commands via mathematical and AI methods [122]

Conventional medical and research-grade EEG devices have been effectively utilised for various brain state estimations. There are differences between consumer and medical systems in terms of the number of electrodes, the complexity of processing, and the ability to remove noise. More design convenience for "real world" occupational application may be seen in inexpensive EEG headsets, like :

- Neurosky
- Mindwave
- InterAxon
- Muse
- Emotiv Epoc
- OpenBCI.

Open-source software, however, and professional development may eventually increase these systems' capacities. This adaptability is advantageous to developing countries, small businesses, and hobbyist users, but the choice of the best models and algorithms will ultimately be highly context-specific.

BCI frameworks may often be divided into several classes . The three classification schemes—dependability, recording methodology, and mode of operation—are shown in Figure 5.1 [123] .

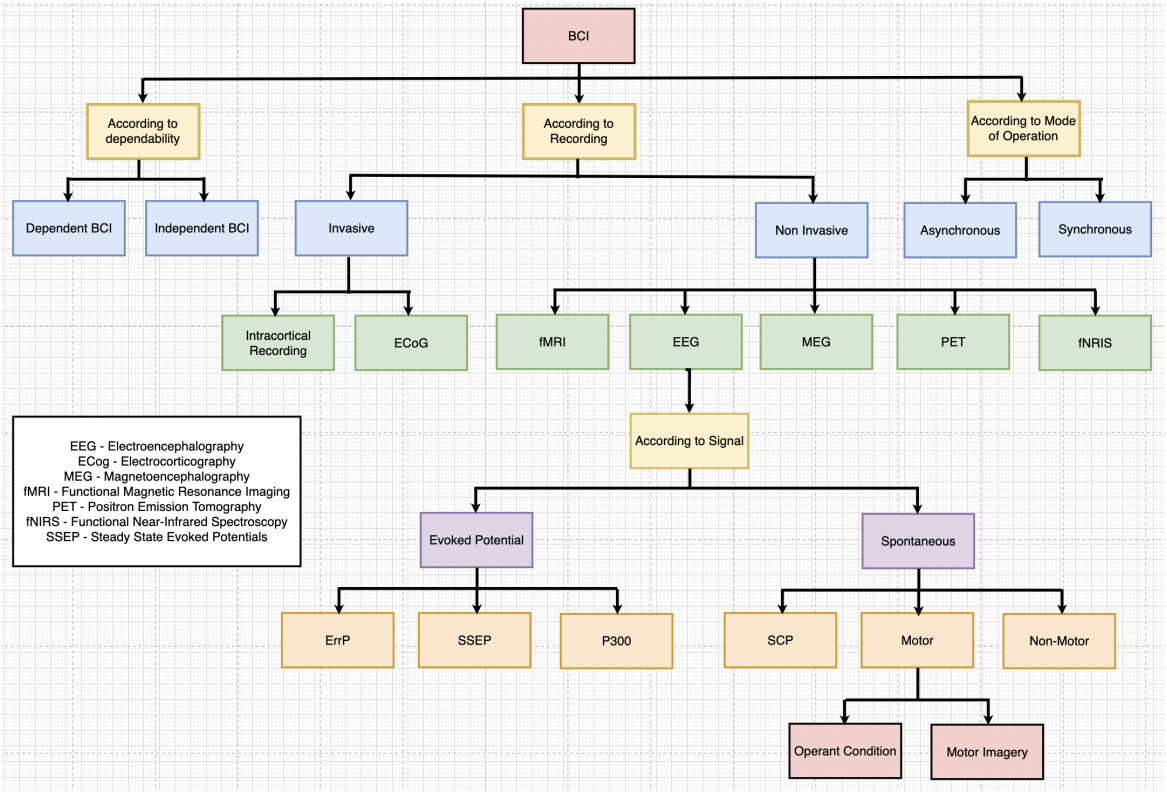


Fig. 5.1 Branches of BCI.

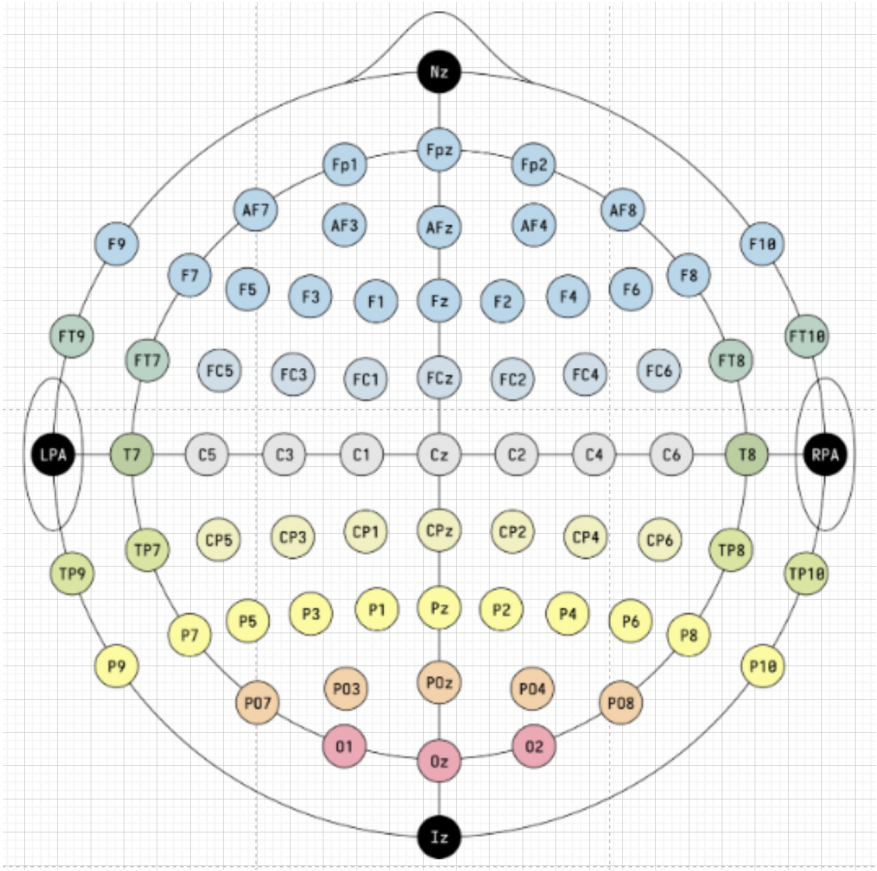


Fig. 5.2 10-20 System.

5.1.1 10-20 System

The figure above shows one of the EEG wearable sensors e.g. EMOTIV Epoch+, with 14 channels. It is a 10–20 system, also known as the International 10–20 system. It is a way of describing and applying the position of scalp electrodes in an EEG test for voluntary lab research.

The "10" and "20" correspond that the actual distances between neighbouring electrodes are 10% and 20% of the entire front-back or right-left distance (nasion to inion) of the skull, respectively. as shown in Figure 5.1.1 [124]. This standard approach was developed to ensure that the results of a subject's study (clinical or research) could be obtained, duplicated, and properly analysed and compared with other studies. The method is based on the idea that an electrode's location is related to the area of the brain beneath it, particularly the cerebral cortex.

The letters designated to each electrode implantation position to denote which lobe or region of the brain it is reading from are pre-frontal (Fp), frontal (F), temporal (T), parietal

(P), occipital (O), and central (C). There are (Z) sites. A "z" (zero) site is an electrode that is positioned on the midline plane of the skull (Fpz, Fz, Cz, Oz), and it is typically used as a reference or measuring point. It is also widely used as "grounds" or "references." On the right side of the head are even-numbered electrodes (2, 4, 6, 8), whereas on the left side of the brain are odd-numbered electrodes (1, 3, 5, 7).

Table 5.1 Brain's Anatomy - Lobes and its Function

Lobes	Functions
Frontal lobe	Planning, Problem Solving, Motivation, Judgement, Decision Making, Impulse Control, Social Behavior, Personality, Memory, Learning, Reward, Attention, Movement
Parietal lobe	Touch, Pain, Temperature, Pressure, Vibration, Analyzing, Recognizing, Memory of Somatic Sensations, Coordination of Visual, Auditory, and Somatosensory Stimuli, Spacial and Body Awareness
Occipital lobe	Seeing Objects/Stimuli, Analyzing, Recognizing and Memory of Visual Stimuli, Shapes, Colors, Sizes
Temporal lobe	Hearing Sounds, Pitch, Frequency, Analyzing, Recognizing Memory of Auditory Stimuli

5.1.2 EEG Frequency Bands

Obtaining raw data in the time domain from the specified electrodes of EEG sensors and translating it into the frequency domain with corresponding Power Spectrum Distribution (PSD) signal energy. The brainwaves are divided into four sub-frequency bands, namely, as follows.

Table 5.2 EEG Frequency Bands and its corresponding Activities

Band	Frequency	Activity
Delta (δ)	1- 4 Hz	Deep Sleep
Theta (θ)	4 - 8 Hz	Drowsiness, Light Sleep
Alpha (α)	8 -13 Hz	Relaxed
Beta (β)	13 - 30 Hz	Active thinking, Alertness, Hyperactivity

These frequency bands are carried along the pink noise, $1/freq$, as shown in Figure 5.4.

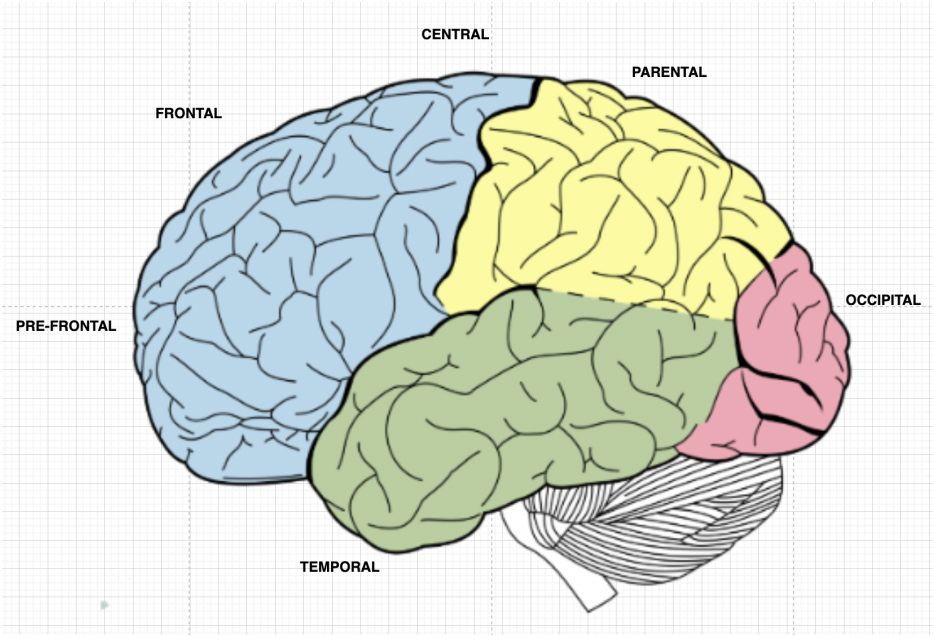


Fig. 5.3 Brain Lobes Anatomy.

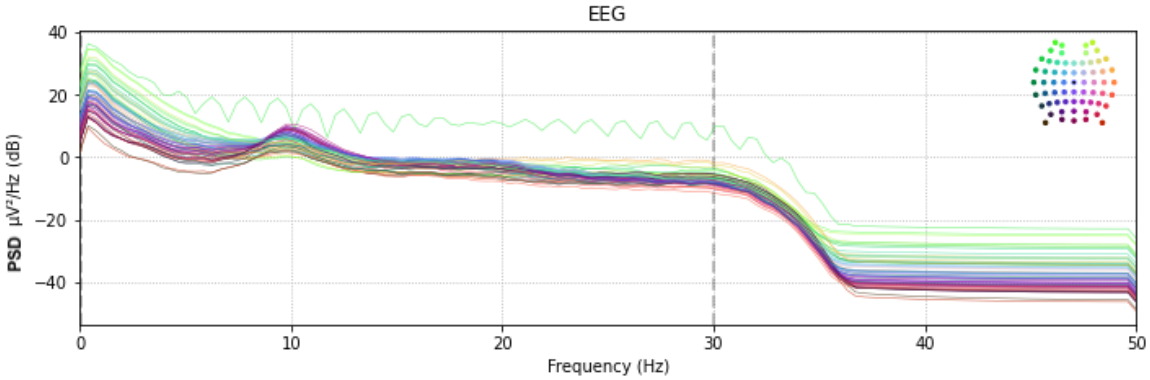


Fig. 5.4 EEG Signal - Spectral Waveform

5.1.3 EEG Signal Analysis

Electroencephalography (EEG) analysis with Computer technology and mathematical signal analysis processes are used in electroencephalography (EEG) analysis to extract information from EEG data. To enhance brain-computer interface (BCI) technologies, help researchers better understand the brain, and aid doctors in making diagnoses and treatment decisions. EEG analysis techniques may be broadly categorised in a different methodology. The approach can be characterised as parametric and non-parametric. Most EEG analysis methods have traditionally been classified into one of four categories: time domain, frequency domain, time-frequency domain, or non-linear methods. Several more current methods exist as follows:

1. **Preprocessing.**

   The EEG signal is weak and needs to be amplified to be brought to a range acceptable for pre-processing.

   - **Artifact Removal**

     Unwanted signals from both physiological and non-physiological artefacts were frequently detected in EEG readings. Physical artefacts include eye movement artefacts (also known as Electrooculography artefacts) and muscle movement artefacts (also known as Electromyography (EMG) artefacts) (EOG). Moreover, additional artefacts are produced by heart activity and are called electrocardiograms (ECG). Non-physiological artefacts are caused by interference from nearby wires and electrical devices. Conventional filters like high pass and low pass filters eliminate the EEG data's low frequency and high frequency. Moreover, various cutting-edge filters have been utilised that aim to increase the signal-to-noise ratio and enhance the quality of EEG information in the signal, including:

     – Finite Impulse Response (FIR)
     – Infinite Impulse Response (IIR)
     – Principal Component Analysis (PCA) – Dimension Reduction
     – Independent Component Analysis (ICA)
     – Empirical Mode Decomposition

   - **Time Epoch - Segmentation** t_s

     The EEG signal is a non-stationary signal by nature; hence its values might change from one location to another. EEG signal segmentation may be used to find a sequence. Moreover, segmentation is crucial for the application of feature

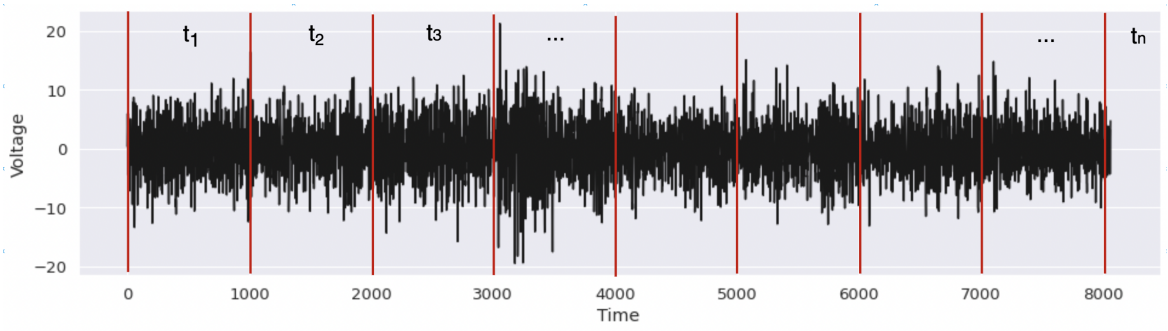


Fig. 5.5 Time Epoch or Sweeping

extraction and classification. EEG segmentation might range from one second to many minutes depending on the intended use, sometimes called epoch or time sweeping.

Time Segmentation t_s is a general term for tuning the scope in desired time to examine the signals, so it is dividing the signal in a series of times and converting it into samples as shown in Figure 5.5.

It is the pre-processing part of the EEG signal before it employs in the Power Spectrum Distribution (PSD). The sub-function Python implementation below is the example of transforming the continuous data into wavelet data of 1 second time sweeping.

Appendix A.1 shows the Python sub-program for generating the time sweeping or epoch from the EEG time raw data.

2. **Features Extraction.**

Different methods to extract EEG features vary from simple features like mean or standard deviation to complicated features in the time or frequency domain or non-linear features; a wide range of EEG data may be retrieved.

- Time domain characteristics including Zero crossing (ZC), Mean absolute value (MAV), Slope sign change (SSC), and Waveform Length (WL).
- Wavelet Transformation
- Hilbert Huang Transformation
- Auto-Regressive (AR) parameters,
- **Power Spectral Density (PSD) using Welch-Simpson Estimation** will discussed in the next section.

3. **Classification.**

   Several algorithms are employed to categorise various mental states in EEG readings. Various applications call for various changes and algorithms. Depending on the desired application, the classification methods can be either supervised or unsupervised as follows:

   - Fuzzy Logic
   - Logistic Regression
   - Decision Tree
   - Random Forest
   - Support Vector Machine (SVM)
   - K – Nearest Neighbor
   - Naives Bayes
   - Genetic Algorithm
   - Particle Swarm Optimisation (PSO)
   - Artificial Immune System (AIS)
   - Artificial Neural Networks (ANNs)

5.2 Power Spectrum Distribution (PSD)

Power Spectral Distribution (PSD) measurement describes the power per unit area per unit wavelength and the measurement of a signal's power content vs frequency. Typically, a PSD is used to describe random broadband signals in spectral resolution to digitise the signal and normalises the PSD's amplitude. The Power Spectral Density is a characteristic of the EEG signal often used. Its feature is computed chiefly using the Fourier Transform (FT) of the time-domain signal, which displays the signal's spectral density distribution in the frequency domain. However, it is possible to compute the PSD from the time domain series directly.

A signal consisting of the same subcarriers has a constant power spectral density (PSD) over its bandwidth and the total signal power shown in Equation 5.1 [125]. The average power P of a signal $a(f)$ over time is therefore given by the following time average, where the bandwidth BW is centred about some arbitrary frequency $f = fc$.

$$P = \lim_{BW \to \infty} \frac{1}{BW} \int_{f_c - \frac{BW}{2}}^{f_c + \frac{BW}{2}} a(f)^2 \, df \tag{5.1}$$

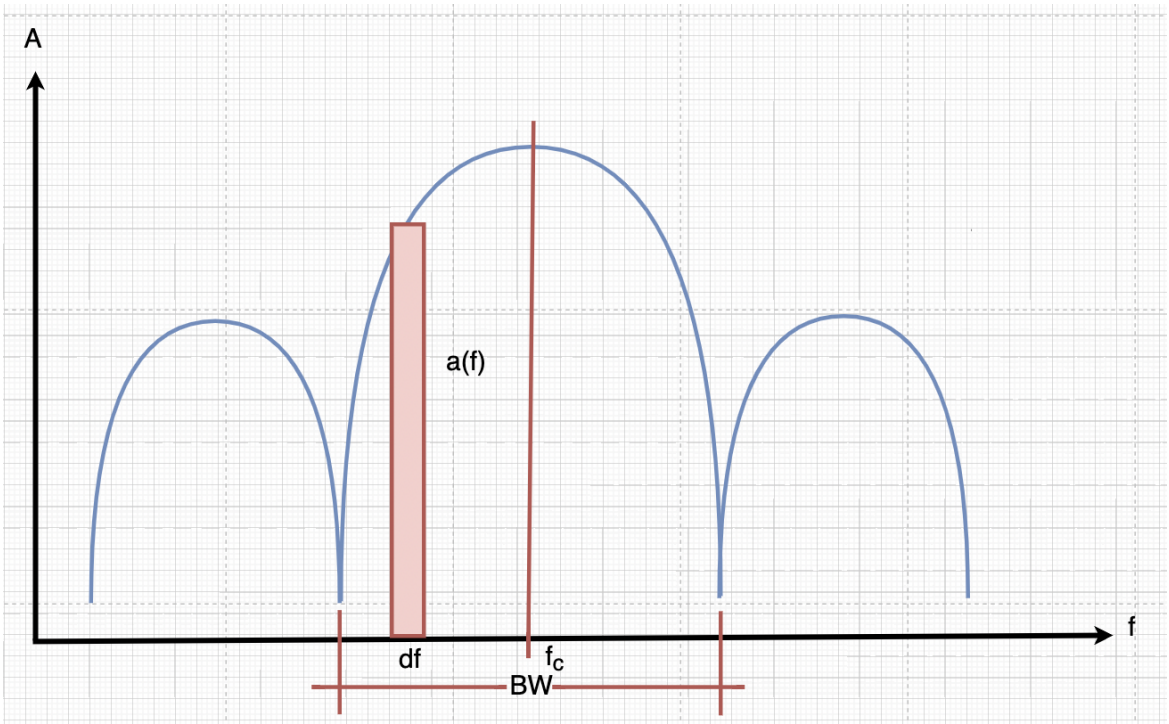


Fig. 5.6 Power Spectrum Distribution (PSD).

Using the transformation, the extract and filter the typical EEG frequency signal band before it is processed in PSD through Fourier Transformation:

- Delta (Δ) $(0.5-4Hz)$
- Theta (θ) $(4-8Hz)$
- Alpha (α) $(8-12Hz)$
- Beta (β) $(12-30Hz)$

This function at Appendix A.1 provides the band PSD of each band for every 1 second and converts it into a data frame in . a CSV file for easy manipulation and data storage.

5.2.1 Short-Time Fourier Transform (STFT) Analysis - Amplitude vs Frequency

The Fourier Transform in the data is linear and stationary or strictly periodic, using uniform trigonometric functions sine and cosine. However, unsteady data or non-uniform and non-linear data, or deformed signals generally require additional harmonics. Moreover, the energy spreads across spurious harmonics. Also, it has a non-physical representation of data.

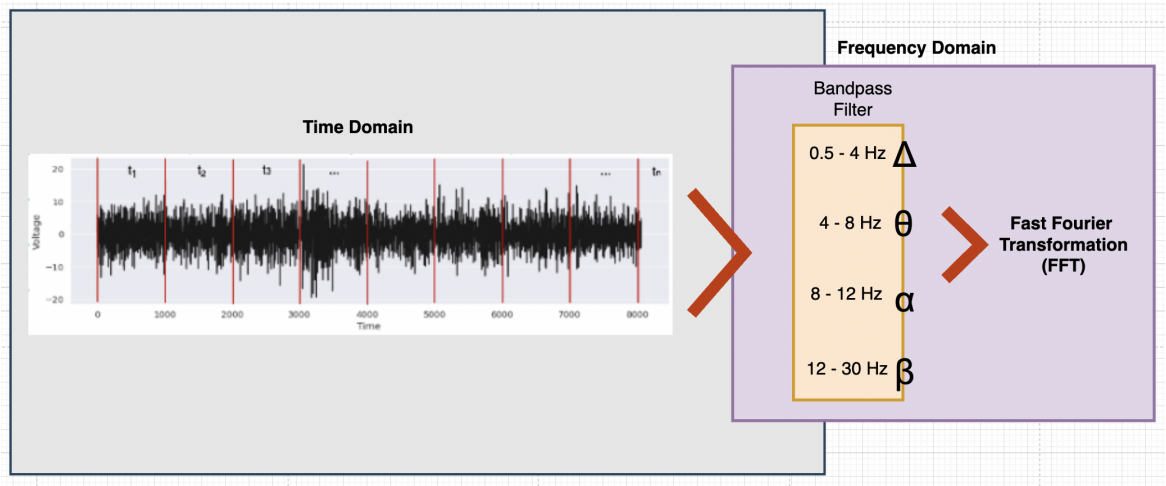


Fig. 5.7 Extraction and Filtering of EEG Signal Bands

It is about a signal added to another signal to generate seemingly complicated signals. Unfortunately, the complicated calculations to decompose the periodic signal x_t in terms of an infinite sum of sines and cosines [126]. Fourier series using the equation:

$$x_t = \frac{1}{2} \cdot a_0 + \sum_{k=1}^{\infty} (a_n cos(\omega kt) + b_n sin(\omega kt)) \tag{5.2}$$

The Fourier Transform is about circular paths (not 1-D sinusoids), and Euler's formula shown in Equation 5.3 shown in Figure 5.8

$$e^{j\omega kt} = cos(\omega kt) + sin(\omega kt) \tag{5.3}$$

will become

$$X_k = \sum_{t=0}^{T-1} x_n \cdot e^{-j\omega kt} \tag{5.4}$$

From Discrete Fourier Transform, the Continuous Fourier Transform is used to transform a signal from the time domain to frequency domain F_k where $\omega = 2\pi f$

$$F_k = \int_{-\infty}^{+\infty} x_n \cdot e^{-2kj\pi ft} dt \tag{5.5}$$

The sub-function shown in Appendix A.1 is the Python implementation of plotting the signal above Figure 5.9 for visualising it in amplitude concerning the frequency domain, that library spectrogram has a Fourier Transformation Function which automatically generates its amplitude corresponding to its frequency.

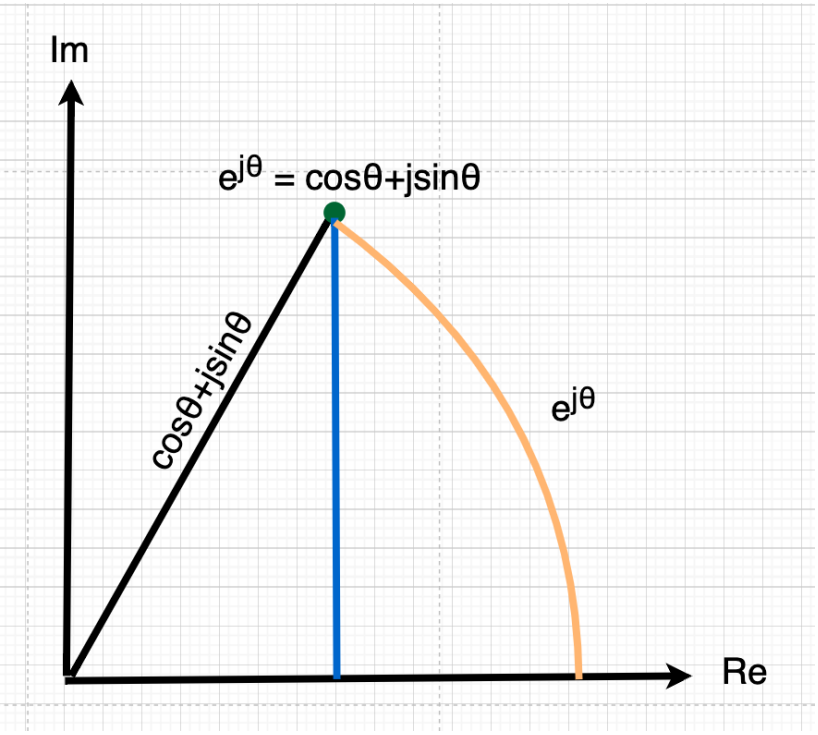


Fig. 5.8 Euler's Diagram

5.2.2 Simpson's Estimation Method and Welch's PSD Calculation

Simpson's Estimation Method

Simpson's rule is a method for approximating definite integrals of functions. It is precise for linear and quadratic functions and typically (but not always) more accurate than approximations made using Riemann sums or the trapezium rule. According to Simpson's Rule, the error in estimating a four-times-differentiable function's integral is proportional to the function's fourth derivative at some point in the interval.

Suppose that $f(x)$ is defined on the range $[a,b]$. The definite integral of $f(x)$ on the interval is therefore approximated by Simpson's rule on the interval by the following formula.

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right). \tag{5.6}$$

When applying Simpson's rule, the interval is often divided into N equal-sized subintervals, where N is an even integer, and the integral over each pair of adjacent subintervals is approximated using the abovementioned estimate.

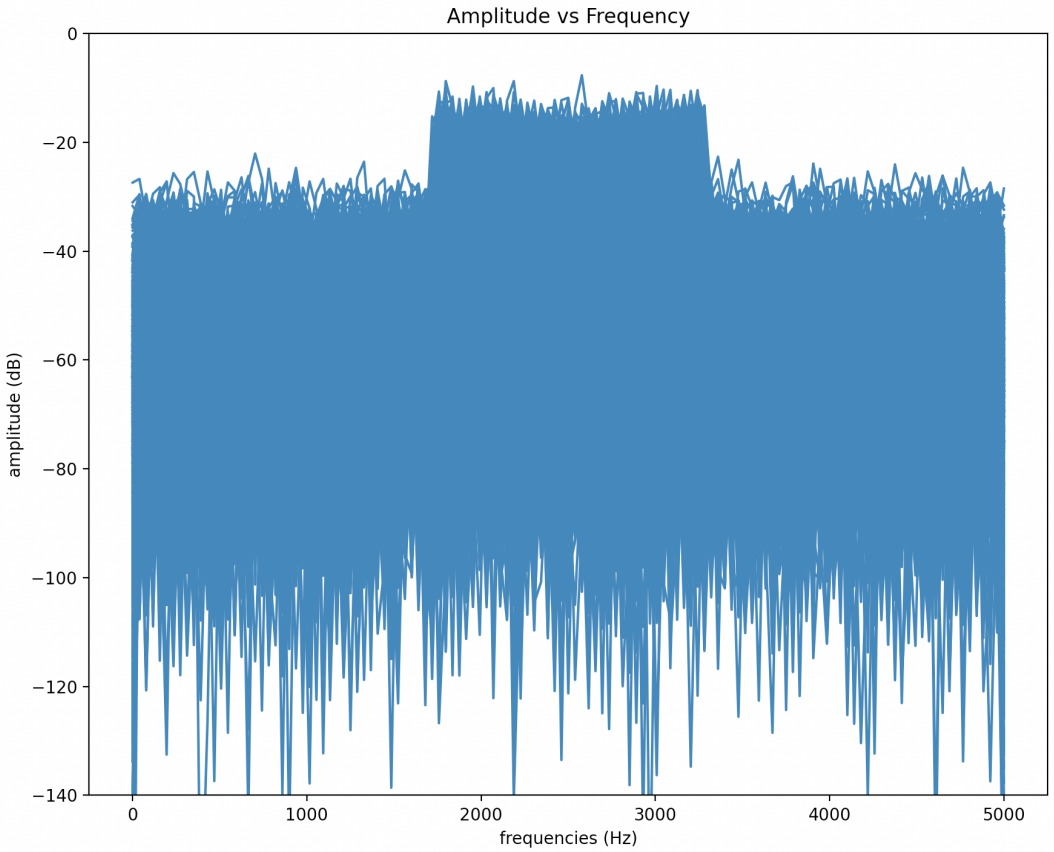


Fig. 5.9 Amplitude vs Frequency Domain of a signal.

Welch's PSD Calculation

Welch's method—known as the periodogram method—estimates power spectra. Breaking the temporal signal into sequential blocks, creating a periodogram for each block, and then averaging.

Using the signal x, denote the mth windowed, zero-padded frame by

$$x_m(n) \triangleq w(n)x(n+mR), \quad n = 0,1,\ldots,M-1, m = 0,1,\ldots,K-1 \tag{5.7}$$

K represents the number of accessible frames, and R is the window hop size. The periodogram for the mth block is thus provided by

$$P_{x_m,M}(\omega_k) = \frac{1}{M}\left|\mathrm{FFT}_{N,k}(x_m)\right|^2 \triangleq \frac{1}{M}\left|\sum_{n=0}^{N-1} x_m(n)e^{-j2\pi nk/N}\right|^2 \tag{5.8}$$

The same way it was previously and the Welch estimate of the power spectral density is provided by

$$\hat{S}_x^W(\omega_k) \triangleq \frac{1}{K}\sum_{m=0}^{K-1} P_{x_m,M}(\omega_k) \tag{5.9}$$

In other words, it is simply a time-averaged periodogram. The periodograms are created from non-overlapping sequential data blocks where $w(n)$ is the rectangular window. Other window kinds generally have their analysis frames overlapped.

The Appendix A.2 shows the Python subprogram for calculating the Simpson Methond and Welch PSD Approximation.

5.3 EEG Applications, Tools, and Libraries

In recent years, many libraries and tools have been developed to advance the study and research of EEG brainwave signals. In parallel, it has been significantly applied with different implementations, from clinical evaluation to medical analysis and brain-computer interface (BCI) applications, as shown in the Figure 5.10. These libraries are used to visualise, analyse, and evaluate EEG brainwave signals.

Some EEG Applications, Tools, and Libraries (not exhaustive list)

Typical analysis tools and libraries commonly used in EEG Brainwave Analysis Visualization and Evaluations:

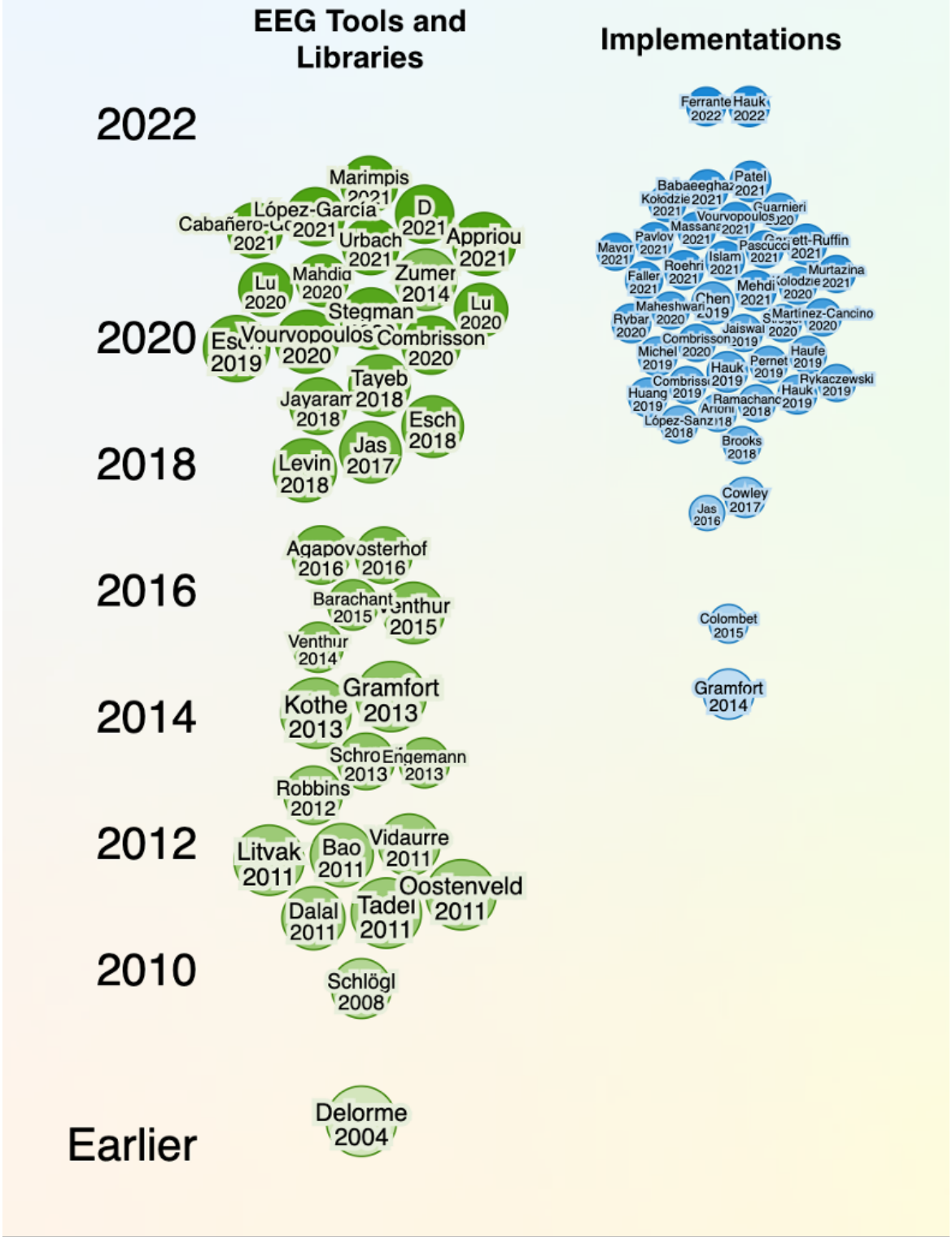


Fig. 5.10 EEG Applications, Tools, and Libraries and its Timeline

- EEGLab [127] 2004
- BioSig [128] 2008
- Brainstorm [129] 2011
- FieldTrip [130] 2011
- SPM8 [131] 2011
- PyEEG [132] 2011
- NUTMEG [133] 2011
- MNE [134, 135] 2013
- BCILab [136] 2013
- ProNTo [137] 2013
- Wyrm [138] 2014
- PyRiemann [139] 2015
- CoSMoMVPA [140] 2016
- BEAPP [141] 2018
- MOABB [142] 2018
- Gumpy [143] 2018
- VisBrain [144] 2019
- NeuroRA [145] 2020
- TensorPac [146] 2020
- FitGrid [147] 2021
- Dyconnmap [148] 2021
- BioPyC [149] 2021
- EEGLib [150] 2021
- NeuXus [151] 2022
- MVPALAB [152] 2022

EEG Brainwave Visualization, Analysis, and Evaluation Techniques

- Statistical Analysis
  - Covariance
  - Correlation
  - Regression
  - ANOVA
  - Normality
  - Sphericity
  - T-test
  - F-test

  - Multivariate
- Time Series Analysis
- Frequency Analysis
- Power Spectral Density (PSD) Analysis
- Event-Related Potentials (ERP)
- Topological Mapping (2D and 3D)
- Event-Related Potentials (resting and activity)
- Functional Connectivity Analysis
- Synchronisation Analysis
- Confusion Matrix Analysis
- Receiver Operating Characteristic Curve Analysis
- Area Under the Curve (AUC)
- Classification Accuracy
- Loss Estimation Analysis
- Principal Component Analysis (PCA) Dimension Reduction
- Independent Component Analysis (ICA)
- Riemannian Potato
- SPM reconstruction
- Cosine Similarity
- Space Sensor Analysis
- Region of Interest Analysis (ROI)
- Hilbert-Huang transformation
- Autoregressive Transformation
- Wavelet Transformation

5.4 EEG -BCI Current Challenges

Much BCI research has been done recently to develop potential assistive technologies. The market has already seen the release of a few commercial BCI appliances. A noteworthy undertaking, the BNCI Horizon 2020 project [153], has put up a BCI future agenda. However, some crucial issues and challenges exist in every component of the BCI paradigm, which has been addressed in the BCI community [154].

Some of the BCI issues have been addressed:

- **Data fusion**, in particular, how the data from multiple electrodes are merged to lessen the data dimensionality and improve the classification results.

- **Issues with EEG headsets**, the quality of EEG data for BCI application mainly depends on the EEG headset and whether the electrode types (wet and dry) offer identical signal properties.
- **Eletrodes standard**, the number of electrodes (1, 3, 4, 8, 14, 16, 20, 24, 32, or 64) and the differences between these headsets are substantial and are incompatible. Thus, it is essential to standardise the minimal number of electrodes for various EEG modalities.
- **Lack of ideal Data Analysis Methods**, some focused on detection, removal of artefacts, or accuracy enhancement, and some feature extraction technique is only suitable to specific mental activity, which is not feasible for specific applications.
- **Commercialisation of EEG-Based BCI Technology**. A trend in Lab-Based BCI Technology leads to a BCI device due to the production of low-quality brain signals. One of the major issues with BCIs is that practically all BCI tests have been done in a controlled lab, independent of the realistic surroundings of the intended users.

Moreover, adding to the dilemma, the anomaly of the EEG signals is the ununified base signal for every specific task, which differs from device to device and subject to subject.

Another issue arises from the research perspective, communicated with the Emotiv spokesperson shown in the conversation in Figure 5.11. The integrity of EEG data passed through Emotiv's cloud system can not be manipulated and does not introduce random signals. At the same time, the limited access to good quality data to investigate and work on it due to restrictions put by the EEG devices provider to protect its data centre. So it will be hard for the researchers to isolate and evaluate the authenticity of the process. **Another option is to find an EEG database large enough and with many subjects involved to test the proposed technique.**

Since the irregularity of EEG headsets, data fusion with different electric biases, and commercialisation of EEG leads to low-quality signals, mathematical intervention and improvement, in general, approached data analysis methodology might help the problem, especially in the part of the removal of artefacts and data cleansing. Given the technical BCI current challenges, the proposed data cleansing technique (PCA-SRP) will be an avenue to solve some integral parts of the challenges [117].

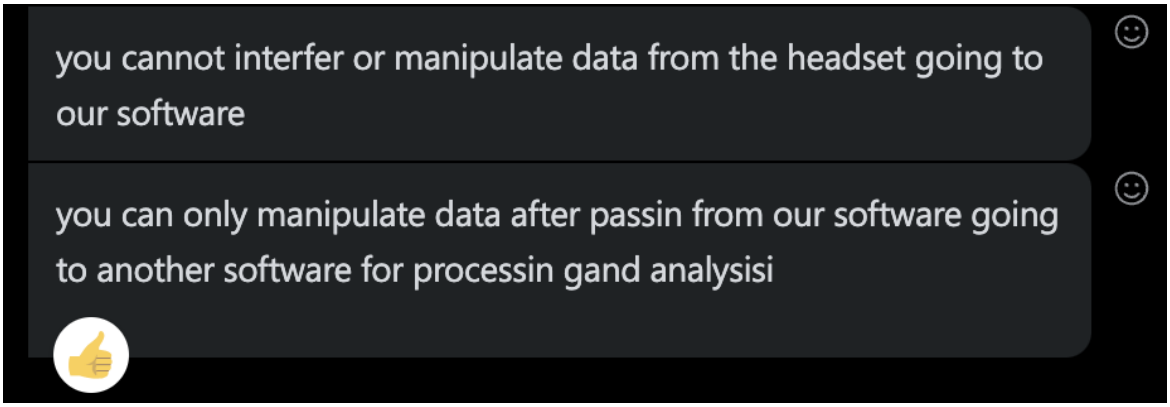


Fig. 5.11 Emotiv spokesperson conversation.

5.5 Summary

This chapter involves the knowledge of Electroencephalography (EEG) Technology, its biological anatomy of the brain and its function, EEG frequency bands, commonly used EEG signal analysis, feature extraction, tools, software application, visualisation techniques, and current EEG-BCI challenges; it provides a basic understanding that supports to comprehend the concepts that will introduce in the following few chapters of this study regarding EEG-BCI technology.

Furthermore, this chapter also supports the determination of EEG datasets, their EEG inconsistencies and anomalies through the explained EEG analysis, feature extraction, and visualisation techniques in this chapter, and the evaluation to be used.

Chapter 6

PCA-SRP Implementation on Physionet's EEG Motor Movement (MM) Dataset

This chapter summarises the experiment protocols and methods for acquiring and pre-processing EEG signals. It visually and mathematically evaluates the effect of the proposed Principal Component Analysis - Sample Reduction Process through the following testing visually and mathematically by reducing the least predictive samples based on the criteria of PCA-SRP in training the simple Artificial Neural Network (ANN).

6.0.1 EEG Motor Movement (MM) Dataset

The researchers used a separate EEG sensor headset to produce the PhysioNet's Motor Movement (MM) - Left and Right (L/R) Hands and Feet Movement EEG signals [16], which included RAW EEG data with various labels and data standards stated in their descriptions. The dataset used in this study was made available to the public, and the characteristics of these databases are detailed in depth. Therefore, these details may depend on the following experimental analysis in the ensuing chapters.

Experimental Protocol

The EEG L/R Motor Movement (MM) Dataset was provided by the developers of the BCI2000 instrumentation device (http://www.bci2000.org). The headset has 43 electrode positioning systems and 64 wet sensors with a sampling rate of 160 Hz (10-10 System). Figure 6.1 shows a placement plan for the electrodes as discussed in Chapter 5.1.1 (10 – 20 System).

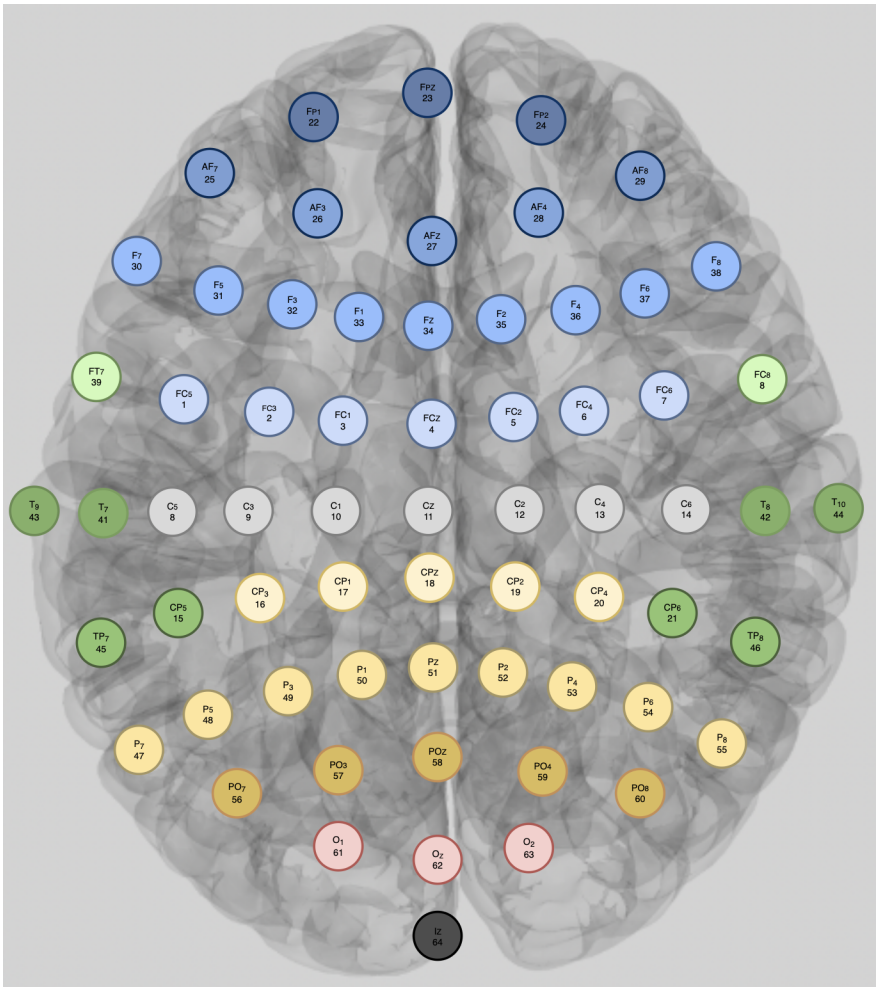


Fig. 6.1 EEG Standard 64 Channel Electrodes Positioning Notation.

The Montage of EEGs was recorded using 64 electrodes per the international 10-10 system standard (apart from electrodes Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10). The numbers behind each electrode name represent the order in which they occur in the data, considering that signals in the records are numbered from 0 to 63, whereas the numbers in the graphic range from 1 to 64.

Furthermore, more than 1500 one- and two-minute EEG recordings comprise this collection belonging to 109 subjects. Each participant conducted 14 experimental runs, comprising three two-minute runs for each of the four activities described below:

1. A target appears on the screen's left or right side. The subject opens and closes the corresponding fist until the target disappears. Then the subject relaxes.

2. A target appears on the screen's left or right side. The subject imagines opening and closing the corresponding fist until the target disappears. Then the subject relaxes.

3. A target appears on either the screen's top or bottom. The subject opens and closes fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

4. A target appears on either the screen's top or bottom. The subject imagines opening and closing either fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

In summary, the experimental runs were:

- **Runs 1:** Baseline, eyes open
- **Runs 2:** Baseline, eyes closed
- **Runs 3, 7, and 11:** Task 1- Motor Action (open and close left or right fist)
- **Runs 4, 8, and 12:** Task 2- Imagery Action (imagine opening and closing left or right fist)
- **Runs 5, 9, and 13:** Task 3 – Motor Action (open and close both fists or both feet)
- **Runs 6, 10, and 14:** Task 4 – Imagery Action (imagine opening and closing both fists or both feet)

Each annotation includes one of three codes (T0, T1, or T2):

- **T0** corresponds to the rest
- **T1** corresponds to the onset of motion (real or imagined) of the left fist (in runs 3, 4, 7, 8, 11, and 12) and both fists (in runs 5, 6, 9, 10, 13, and 14)
- **T2** corresponds to the onset of motion (real or imagined) of the right fist (in runs 3, 4, 7, 8, 11, and 12) and both feet (in runs 5, 6, 9, 10, 13, and 14)

These comments are stored as 0, 1, or 2 in the "target" state variable in the BCI2000-format versions of these files [155]. Furthermore, the dataset is configured by combining the same channel of the subject (1-109) as shown in Figure 6.2.

This EEG dataset is classified as non-invasive according to recording, spontaneous according to EEG signal, and a motor imagery dataset as described in Figure 5.1. Due to its benchmark EEG dataset as stated in Chapter 1.1, large enough as discussed in Chapter5.4, and the ample research study [156–158] about complicated ANN classification models using

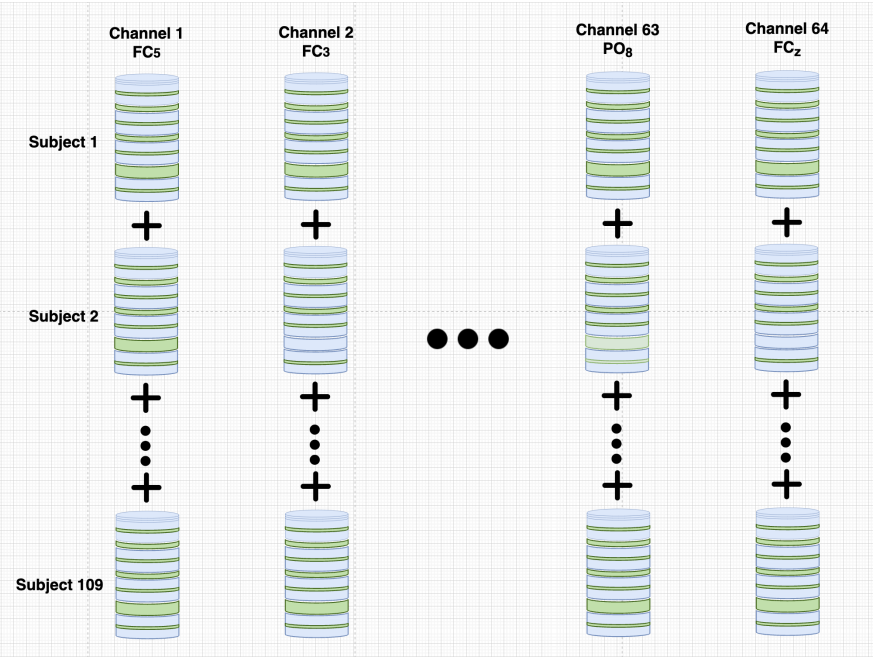


Fig. 6.2 MM/I Experimental Dataset.

this dataset as a supplement approach in improving those prior research studies from the past literature as stated in Chapter 3.2.

Furthermore, the said EEG dataset is chosen for the unbiased and the actual authenticity of how the data is actually gathered clinically by the medical expert.

6.1 Experimental Framework

The experiment is set up in a way that involves the Physionet's L/R Motor Movement EEG signal. Conventionally, the EEG signals are sampled through epochs and analysed in the time-frequency domain, which is also part of the procedure. As shown in Figure 6.2, each subject's EEG data has been concatenated to analyse according to EEG channels. As described in Chapter 4, EEG signals have been characterised as **noisy**, **non-deterministic**, **non-linear**, **non-stationary**, and **non-unified reference base signal**, which is understandably incomprehensible data - the EEG signals.

Figure 6.3 shows the Experimental Framework that meticulously evaluates the time domain - EEG signal acquired in the datasets - (L/R MM) is converted to the frequency domain by **Fast Fourier Transformation**, and the **Power Spectral Density** calculated through **Simpson's and Welch Approximation Calculation** in EEG band signal through a bandpass filter (Δ, Θ, α, and β) as described in Chapter 5.2.

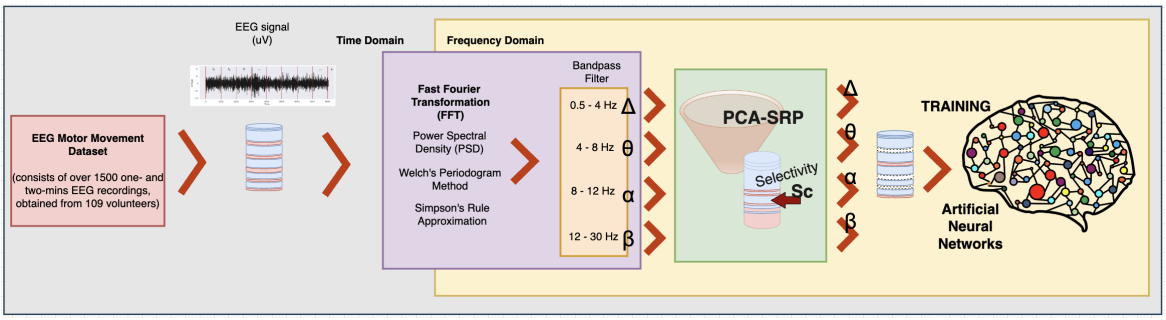


Fig. 6.3 Experimental Structured Framework

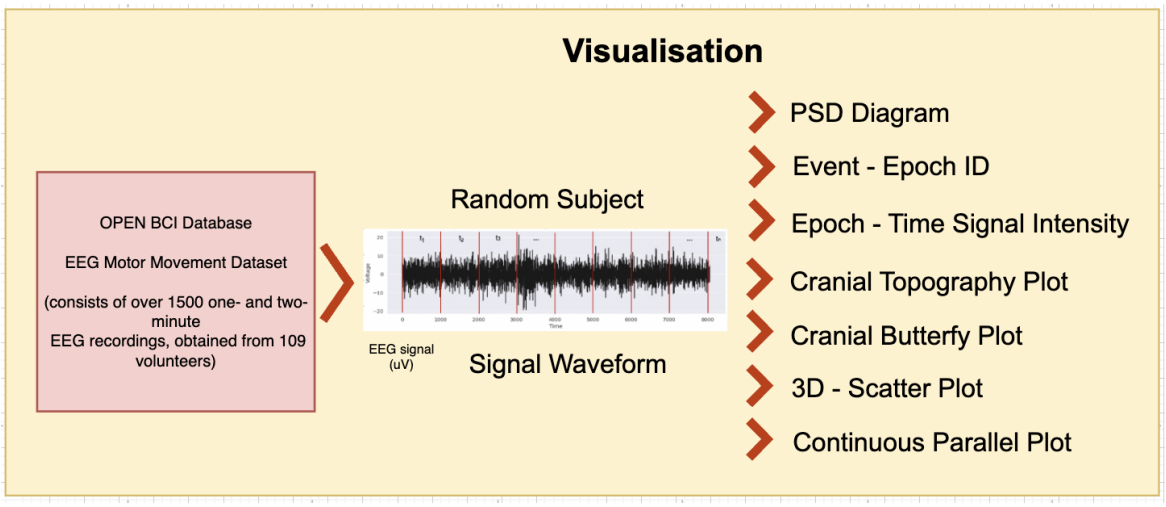


Fig. 6.4 MM - EEG Signal Visualisation

These techniques of EEG feature extraction (frequency domain) and approximation in EEG motor and imagery dataset is implemented due to its being standard extraction and analysis [159] and concluded as described as best feature extraction for MI-based application, in real time [160]. Although another feature extraction techniques have been introduced but still under further research and study.

Furthermore, the implementation of Principal Component Analysis - Sample Reduction Process (PCA-SRP) mathematically in the dataset as discussed in Chapter 3 with given Selectivity ($Sc = PC_1$) based in the Equation 3.4. The filtered EEG dataset trains the basic Feed Forward ANN Model, specified in Table 4.1.

6.1.1 EEG Datasets Comparative Visualization

Understanding the EEG datasets needs a visualise it from different perspectives to comprehend the behaviour of its attributes and response to different changes. As well as to figure out the characteristic of each sample.

In the following graphs, MNE [161, 135] EEG Python processes describe and summarise the EEG L/R Motor Movement (MM) Paradigm Dataset. The combined 109 subjects – EEG channels datasets have been assessed with different visualisation plots as shown in Figure 6.4 as follows:

1. **Signal Waveform Diagram** that shows the Amplitude vs continuous time visualisation diagram of the EEG signal waveform of the particular electrodes concerning its EEG channel/s; it identifies the signal amplitude peaks and can compare it to other channels as shown in Figure 7.3. The basic visualisation of a signal-time domain; identifies which of the electrodes have a different biased, reference signal, or anomaly in calibration. It might also identify if the signal is a good EEG signal, uses a different sensor, or is from a different subject.

   The Appendix A.2 shows the Python Implementation for plotting the EEG raw data in the time domain.

2. **Power Spectrum Density (PSD) Diagram**, which shows the power in dB for the inclusion of EEG bands (0 – 50Hz) of each channel as shown in Figure 7.4. This visualisation is viewed in the frequency perspective or frequency domain; it will identify if the signal is an EEG signal and which frequency band is responsible for specific mental actions.

   The Appendix A.2 depicts the Python Implementation for plotting the EEG raw data in the frequency domain.

3. The **Event - Epoch ID Diagram** shows and visualises the time and epoch wherein the activity has been made as shown in Figure 7.5. Using this visualisation separates the time limits of each sample and the corresponding events that occur and is also responsible for dividing the signals into different samples or epochs.

   Python Implementation for visualising the EEG events shows in Appendix A.2.

4. **Epoch/Channel – Time Signal Intensity Plot** – to visualise the signal intensity in respect of time as zero time is the actual time of motor movement event (Left / Right Hand and Feet movement). This visualisation is responsible for identifying the specific part of the brain and epoch for certain actions induced electrically.

The Python Implementation for Visualising the Epoch-Time Signal Intensity Plot shows in the Appendix A.2.

5. **Cranial Topography Plot** – to visualise the signal intensity Localisation of the brain's cognitive activity at a specific point in time. Moreover, **Cranial Butterfly Plot** is a plot of superimposed sensor time series, the negative and positive ongoing traces in an interesting period via a highlighted parameter. Cross-reference the electrical intensity to some areas of the brain, which can identify the epoch and subject differently from the rest of the samples.

   For visualising the Cranial Topography and Butterfly Plot in Python Implementation at the Appendix A.2

6. **3D Scatter Plot** to visualize in 3-dimension, the two EEG datasets which capture even more variance. This visualisation is the scatteredness of the sample according to its target events or the identification of simulated controlled randomness in the PC space.

   Here is Python Implementation for visualising plot the 3-D Scatter Plot and Cranial Localization Sensor Visualization shown in Appendix A.2 and A.2, respectively.

7. **Parallel Coordinates Plot** to determine the overall path of each sample in features of two EEG datasets either or different mental actions or good. This visualisation is responsible for knowing the dataset s response of different samples - target events or identifying the simulated controlled randomness to its featured dimensions.

   The Appendix A.2 depicts the Python Implementation for visualising plot the Parallel Coordinates Plot.

8. **Scatter Matrix Plot** to generalise the relation of the samples between the two EEG datasets in their featured - EEG signal bands. It is a grid (or matrix) of scatter plots used to visualise bivariate relationships between combinations of featured dimensions such as Delta (Δ), Theta (Θ), Alpha (α), and Beta (β). Each scatter plot in the matrix visualises the relationship between a pair of dimensions, allowing many relationships to be explored in one chart.

   ThePython Implementation for visualising plot the Scatter Matrix Plot shown in the Appendix A.2.

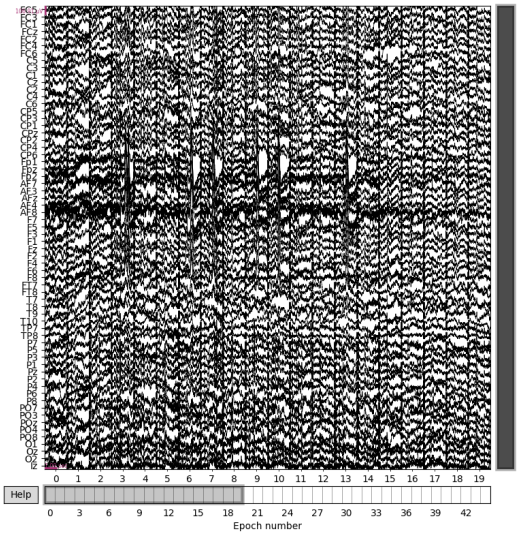


Fig. 6.5 Subject 40 - Left-Right Side Motor Movement: Signal Channel Plot

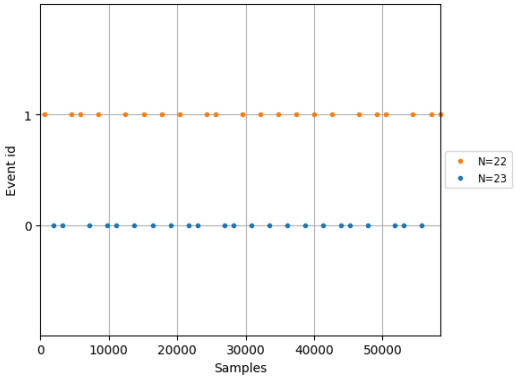


Fig. 6.6 Subject 40 - Left-Right Side Motor Movement: Event-ID Plot

6.2 Discussion and Result

This section involves the human brain's electrical activity of moving the left or right either hands and feet, the Principal Component Analysis - Sample Reduction Process (PCA-SRP) 's visual result, and its effect on basic Artificial Neural Networks (ANN) Model through bootstrapping analysis.

6.2.1 Brain's Electro-physiological Behaviour

The epoch is one of the suitable parameters to consider for holding a single event of mental action. **subject 40 is chosen example due to normal occurrences of the subject's brain activity electric response in doing the motor movement action.**

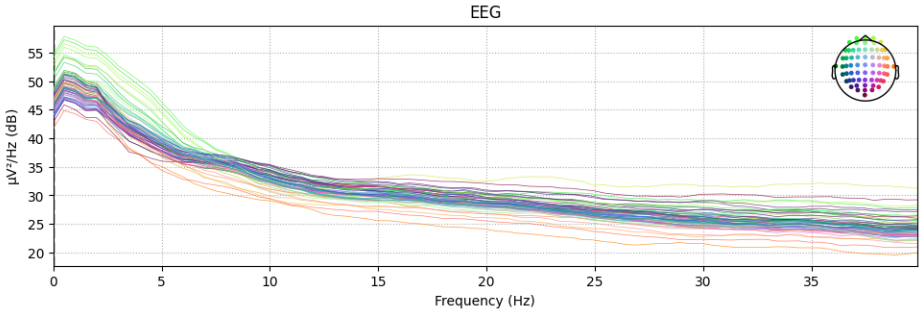
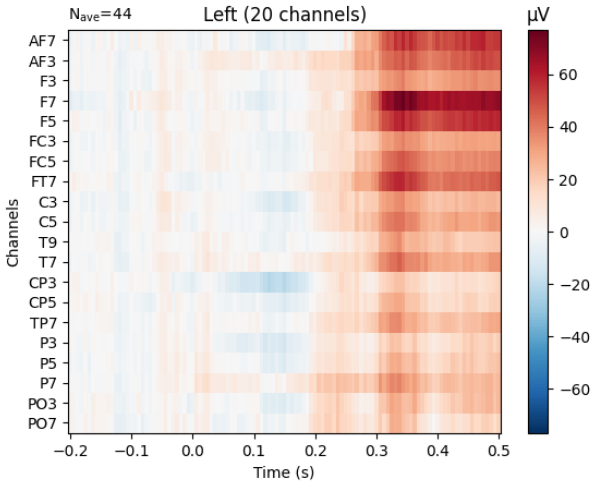


Fig. 6.7 Subject 40 - Left-Right Side Motor Movement: PSD Plot

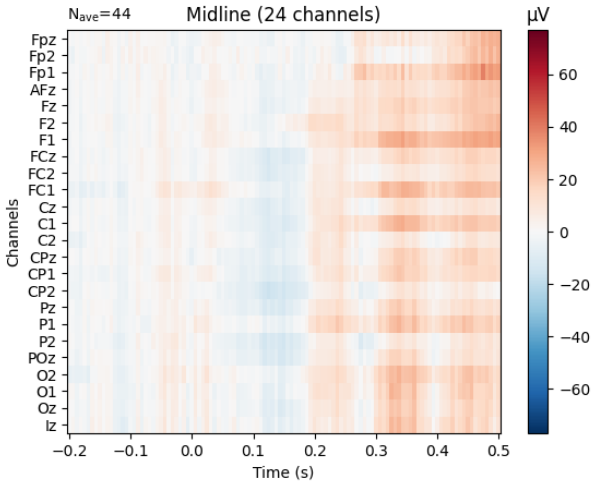
This Motor Movement EEG of subject 40, shown in Figure 6.5 - describes how the time signal is divided per epoch or samples. Moreover, the Figure 6.6, there are approximately 370 seconds of EEG recording for 64 channels at 160 Hz sampling frequency, which gives off 45 epochs entirely or approximately 8.2 seconds or 1315 samples each epoch and the respective sample event ID: zero (0) signifies the left-side motor movement, and one (1) the right-side motor movement of the hand and foot.

Figure 6.7 of subject 40 shows the typical response of Power Spectral Densities (PSD) through a pink noise ($\frac{1}{freq}$) frequency domain response of all the channels electrodes.

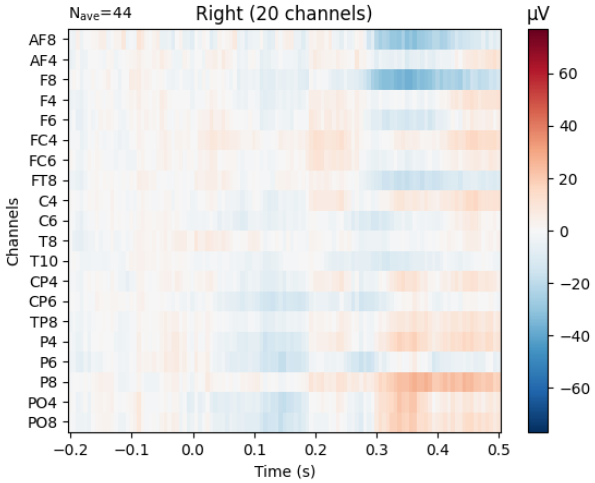
As observed in Figure 6.8, it shows the left side body motor movement of hands or feet mental activity in the following specific Region of Interest (ROI) of the brain - Left, Mid-line, and Right Side Hemisphere. An upward deflection is surface negative, and a downward deflection is surface positive signal intensities., which the cranial serves as the reference neutral. It is also indicated that the Frontal Lobes are responsible for motor activities, as discussed in Table 5.1 - Brain's Anatomy - Lobes and its Functions.



(a) Left-side Brain Channels

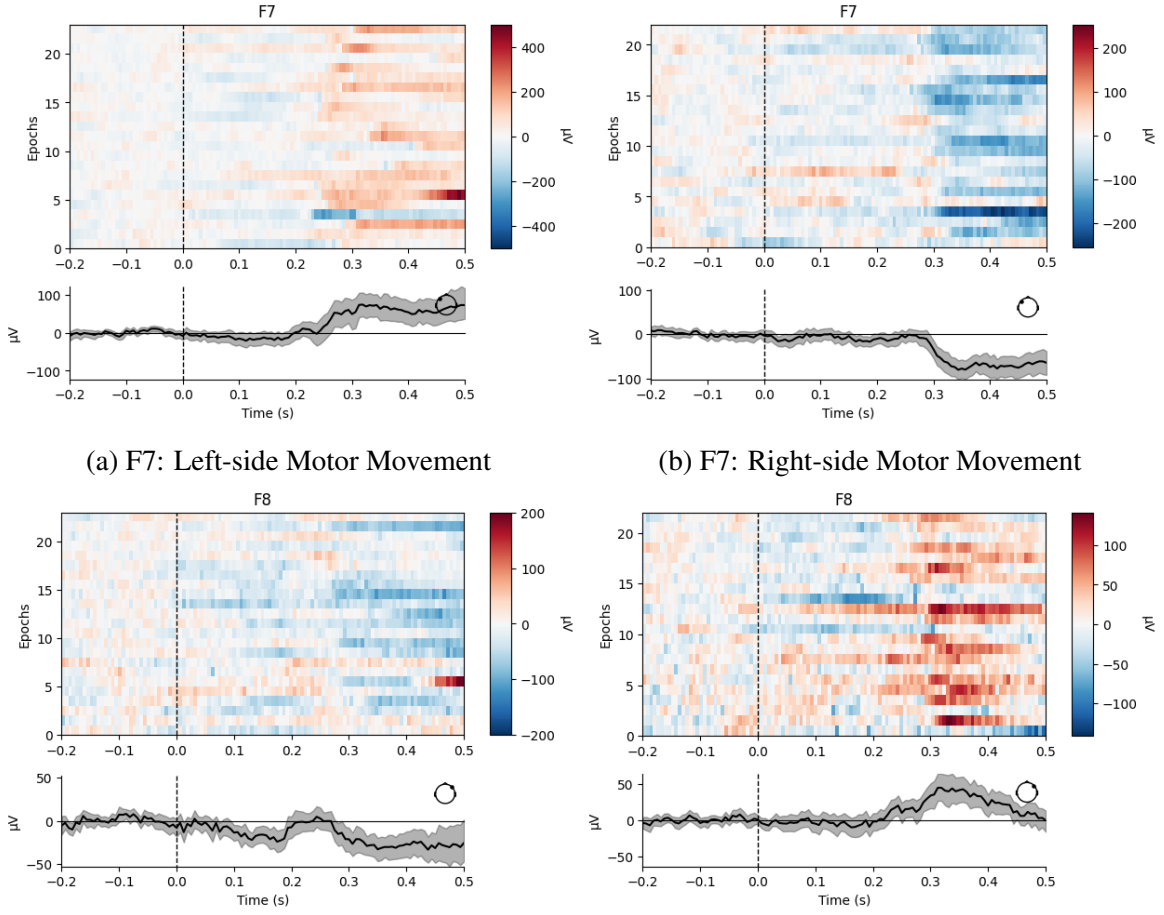


(b) Mid-side Brain Channels



(c) Right-side Brain Channels

Fig. 6.8 Subject 40: Region of Interest (ROI)- Signal Intensity Plot



(a) F7: Left-side Motor Movement

(b) F7: Right-side Motor Movement

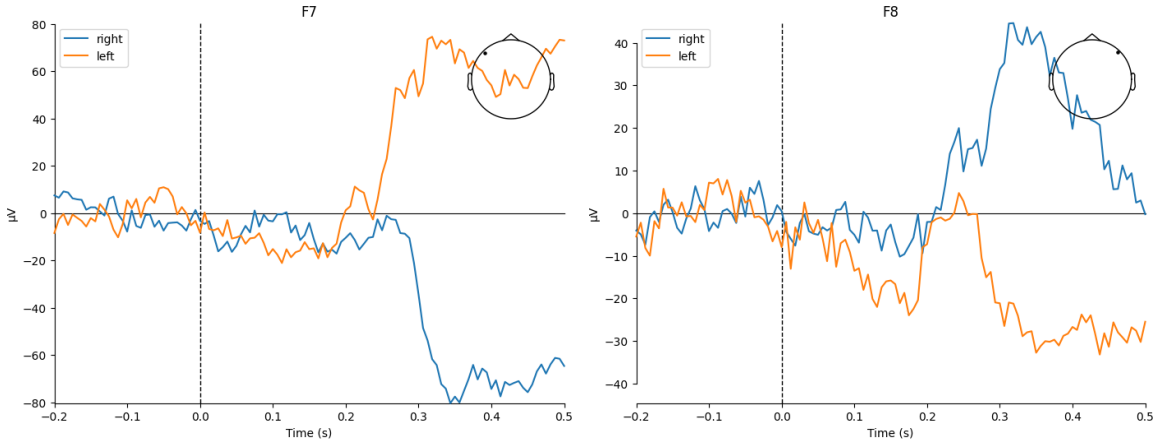
(c) F8: Left-side Motor Movement

(d) F8: Right-side Motor Movement

Fig. 6.9 Subject 40- Left-Right: Epoch-Time Signal Intensity at Channel F7 (29) and F8 (37)

Figure 6.9 shows the intensity level in both F7 and F8 channels - Frontal Lobes. When the left hand and foot move, F7 gives positive signal intensities, and the F8 channel shows negative intensities and vice versa in the movement of the right hand and foot.

Furthermore, a 0.0 ms timeline signifies the actual start of the movement occurs, but it shows some delays after 300 ms, approximately before the brain recognises it. Figure 6.10 is the average intensities of left and right hand and feet motor movement at F7 and F8 EEG channels, that there is a shift of polarity (negative to positive or vice versa) when changing to the movement from left to right or vice versa for channel F7 and F8.



(a) F7 Channel - Average Signal Intensity (b) F8 Channel - Average Signal Intensity

Fig. 6.10 Subject 40 - Left-Right: Average Signal Intensity Plot

Figure 6.11 shows that subject 40 is more positive deflection for most of the channels but is more positive at channel 29 (F7). However, channel 37 (F8) shows more negative. It depicts that these two specific regions are more responsible for motor movement among the other EEG electrodes for subject 40.

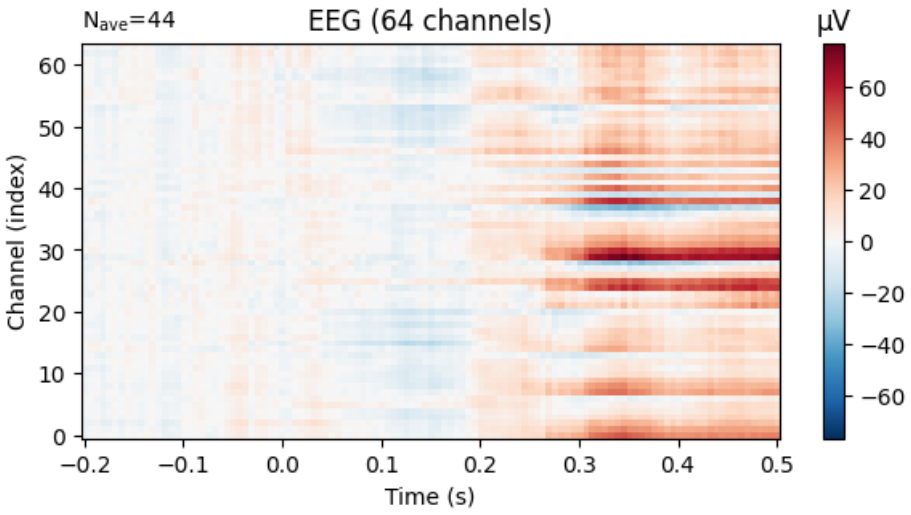
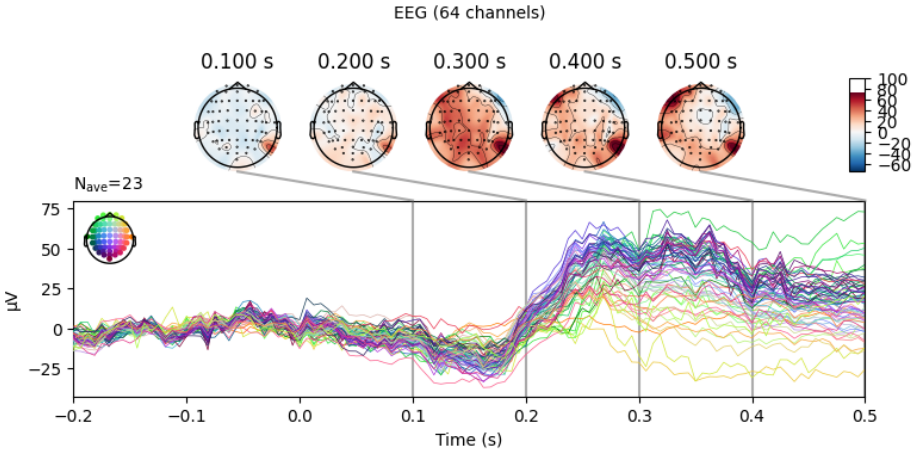
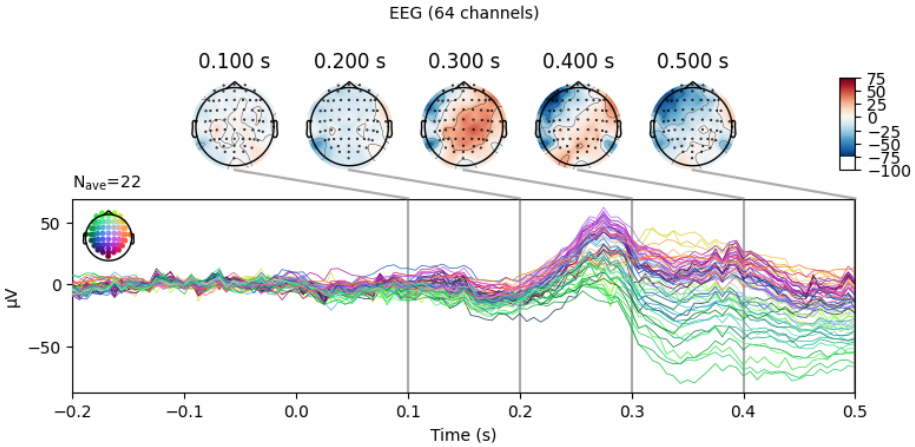


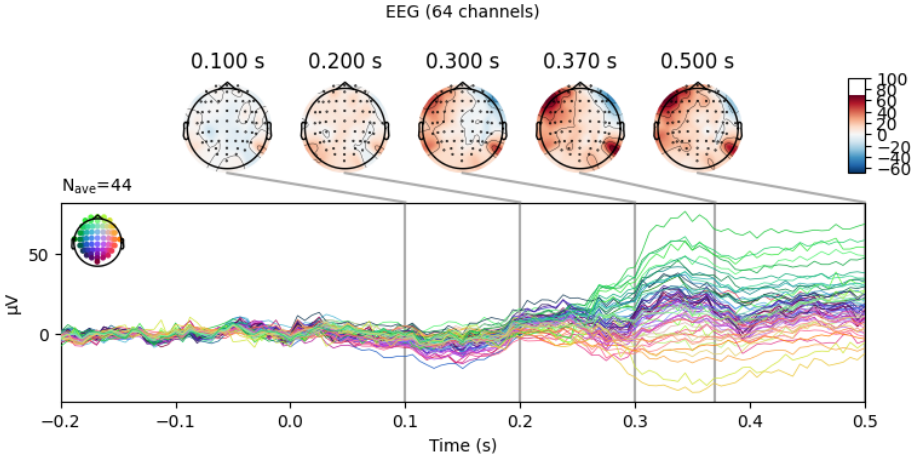
Fig. 6.11 Subject 40 - Left-Right: Channel Signal Intensity Plot



(a) Left-side Motor Movement



(b) Right-side Motor Movement



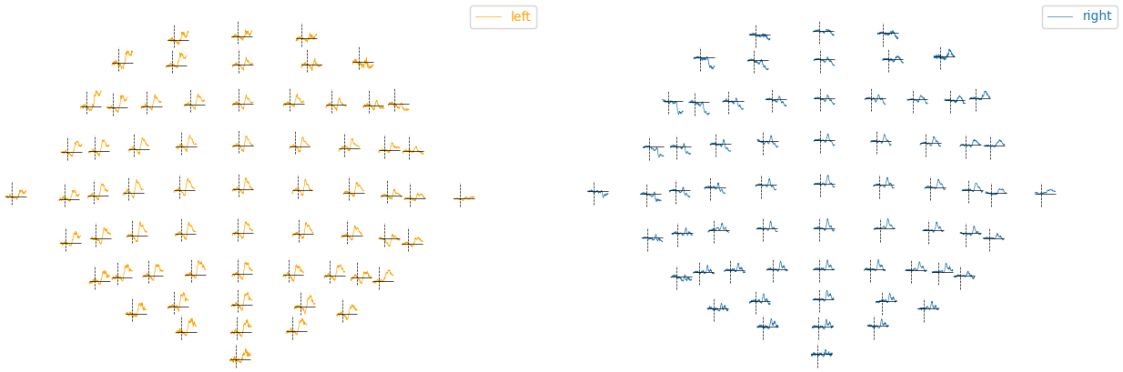
(c) Left - Right (difference) Motor Movement

Fig. 6.12 Subject 40: Cranial Topography and Butterfly Signal Intensity Plot

As shown in Figure 6.12, there is a burst of signal intensities positively and negatively approximately 300 ms after the action has been done by moving the subject's hand and feet left or right. It shows how the brain perceives and acknowledges the motor action of the individual. Wherein the left-side motor movement indicates more red bursts (positive deflection) in the left side of the brain and few regions of blue burst (negative deflection) in the right side of the brain as shown in Figure 6.12a while in the Figure 6.12b, it is an entirely different response in the right-side motor movement of hands, feet, or both

Figure6.12c, it is shown that the signal intensity difference between left to right side motor action appears more intensities (positively and negatively) in the Frontal Lobe of the brain, especially noticeable in F7 and F8, which is entirely accurate in the majority of the subjects. Moreover, central and occipital gave almost zero difference between the two motor movements, which signifies no reaction to any motor movement of hands or feet. Furthermore, TP7-10 and P7-10 show some activity but are not related to L/R motor action due start of the appearance before the motor action has been made.

Figure 6.13 shows every EEG channel's different signal time responses. As shown in Figure 6.13a, the left-side motor movement shows negative deflection at the right frontal region of the brain, and the rest of the region depicts positive deflection signal intensities. However, the right-side motor movement shows negative deflection at the left side of the brain while the rest of the region is at positive negative deflection signal intensities shown in Figure 6.13b. It justifies the general rule that the left and right hemispheres of the brain control the 'opposite' side of the body.



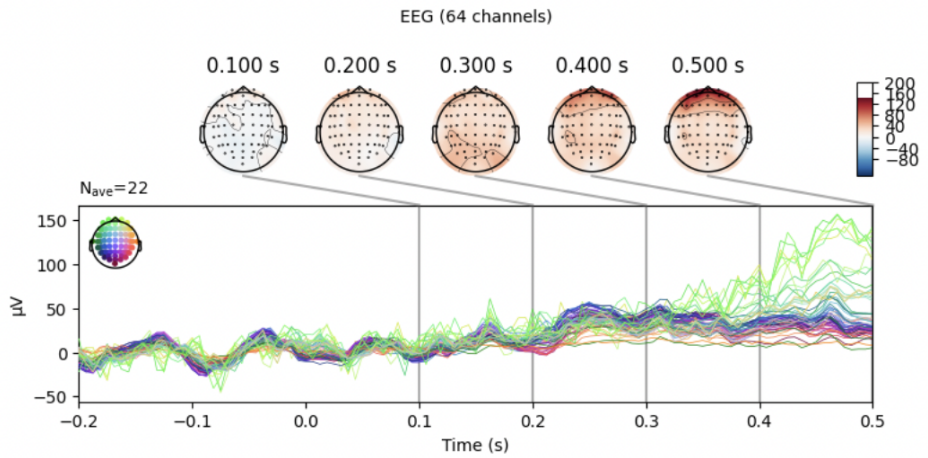
(a) Left-side Movement (b) Right-side Movement

Fig. 6.13 Subject 40: Left - Right: Channels Signal Plot

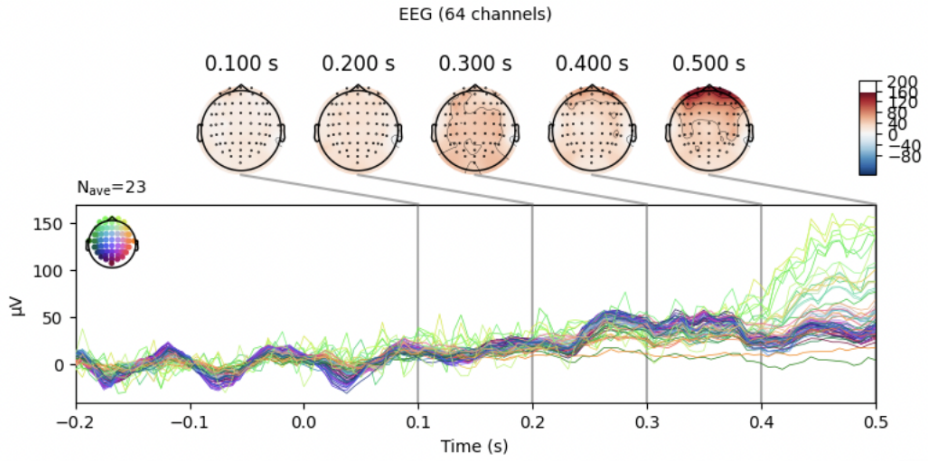
EEG Electro-physiological Anomalies

Among the 109 subjects, there are some constitute an unusual and considered anomaly, like subject 83, which shows very high positive intensities in Frontal Lobes for both L/R Motor Movement as shown in Figure 6.14; the causes are unknown due to lack of supporting data provided. Another unusual EEG signal shown in Figure 6.15 is that parts of the Central Lobe of subject 109 acted with different signal waves than the rest of the channels. That might cause by sensor-biased problems or electrode fixture misalignment.

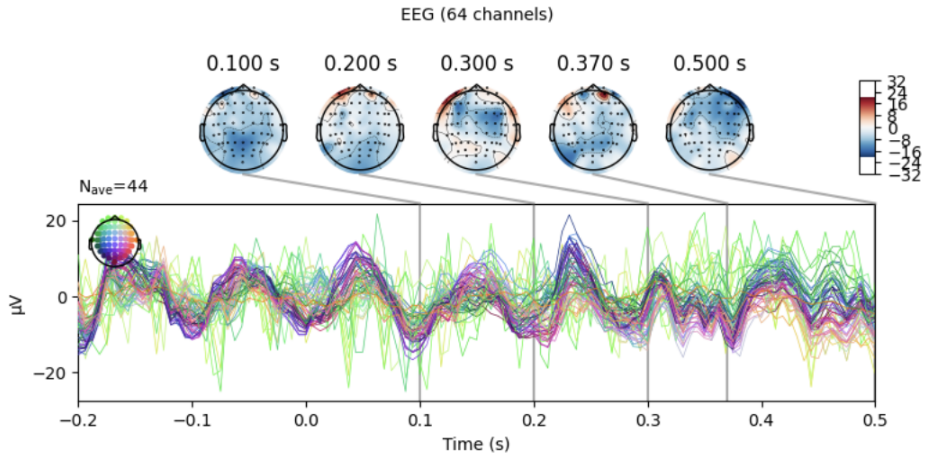
Furthermore, combining the same channel of all 109 subjects, as shown in Figure 6.2, with different reference-biased conditions and the unique wiring of the individual's brain, led to seemingly stochastic data that alter the training structure of ANN's random function approximation.



(a) Left-side Motor Movement

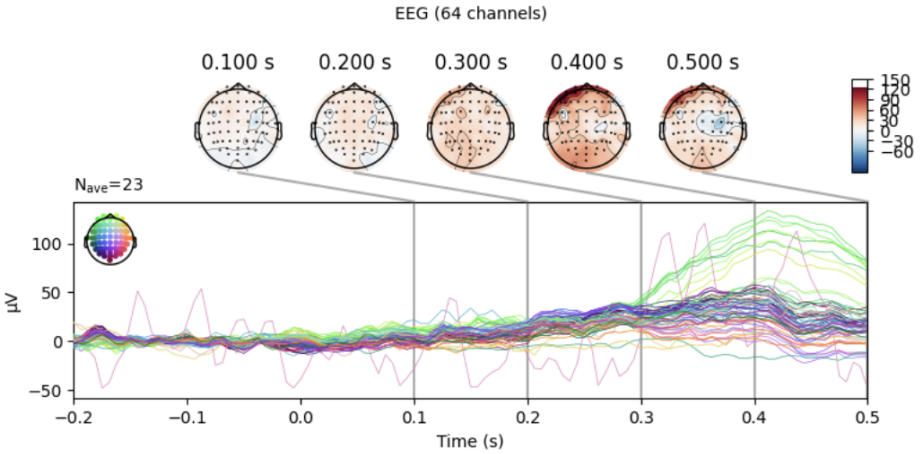


(b) Right-side Motor Movement

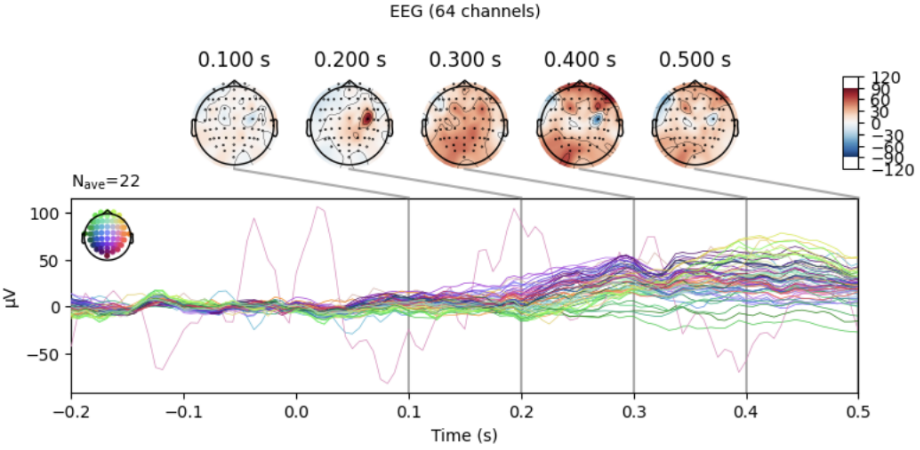


(c) Left - Right (difference) Motor Movement

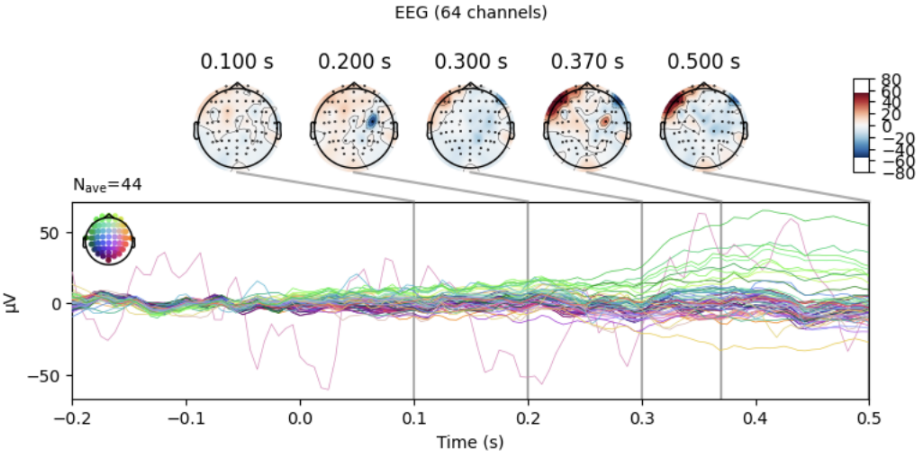
Fig. 6.14 Subject 83: Cranial Topography and Butterfly Signal Intensity Plot



(a) Left-side Motor Movement



(b) Right-side Motor Movement



(c) Left - Right (difference) Hands and Foot

Fig. 6.15 Subject 109: Cranial Topography and Butterfly Signal Intensity Plot

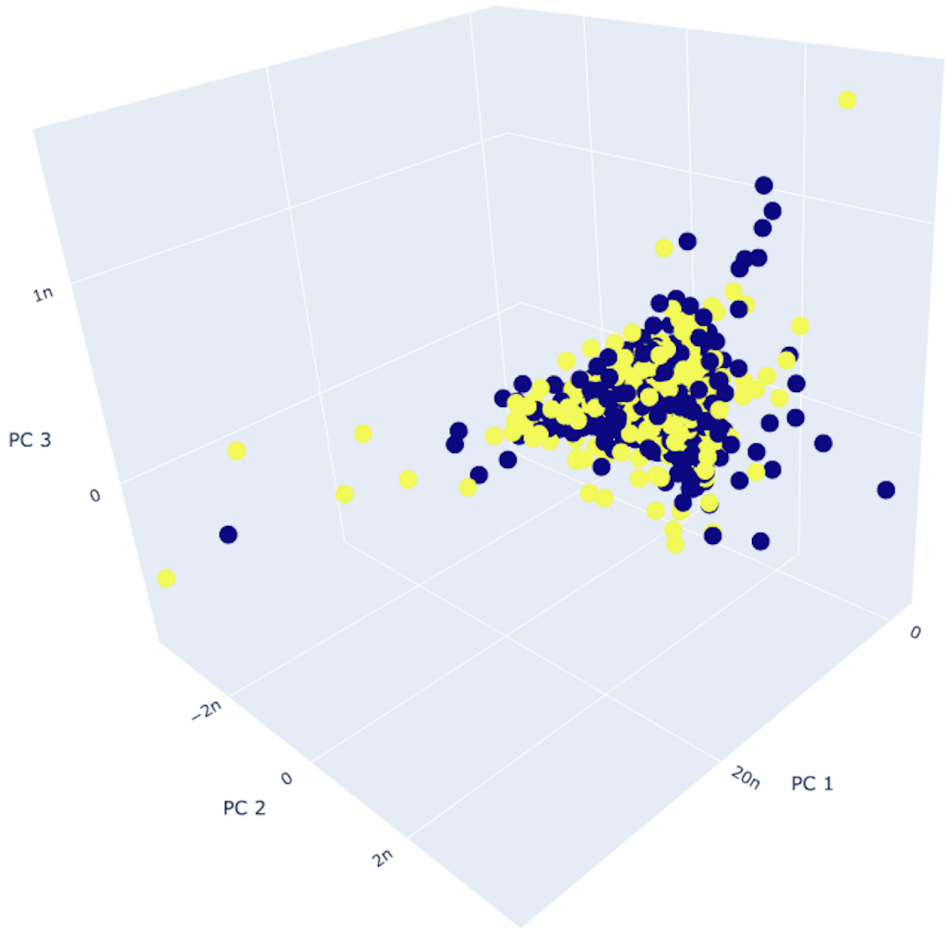


Fig. 6.16 MM Dataset Scatter Plot

6.2.2 PCA - SRP in L/R Motor Movement (MM)

This section describes the effect of PCA in the Motor Movement EEG Dataset, particularly in the L/R motor movement. Figure 6.16 shows the PC space of the dataset; blue represents the left-side motor movement, while yellow represents the right-side movement. Visualizing the PC space of the L/R movement, it is difficult to grasp the general distribution of PC points, due to the low loading scores tends to diverge near to [0,0,0] PC space.

The sample epochs have been converted from 4 dimensions (Δ, Θ, α, and β) to 3 dimensions ($PC1$, $PC2$, $PC3$) to properly grasp visually the dataset. As observed in the L/R MM Dataset Scatter Plot, most of the samples are near to [0,0,0] PC space, which means the majority of the samples are close to each other concerning its variance in a sense, it is difficult to identify a Selectivity (Sc) threshold. The closer the samples to [0,0,0] PC space,

the sample is the less variance it is. However, by the concept of this experiment, the good and bad samples are unidentified, and the previous experiment in Chapter 3 at Equation 3.15 hypothesised that recommended Selectivity is approximately equal to the first Explained Variance Ratio (PC_1).

Even the Figure 6.17 - L/R MM Dataset Continuous Parallel Plot indicates the path of each sample in the EEG Frequency Band and shows some degree of difficulty in concluding the path relation between movement in left-side over ride-side motor movement. Hence, hard to identify the relationship between the seemingly random samples.

As the Principal Component Analysis - Sample Reduction Process (PCA-SRP) took place with Selectivity (Sc) Threshold with $Sc = PC_1$ indicated in each channel in Figure 6.18a, showing that the lowest Sc is in the region of Occipital and Parental Lobes and the highest Sc is in Frontal and Pre-Frontal Region of the brain. Hence, as observed in Figure 6.18b, more samples have been removed in the Parental and Occipital Lobe of the Brain. The least are in the channels of the Pre-Frontal and Frontal lobes out of the 109 subjects, with approximately 45 epochs or events per subject and approximately 4927 samples.

As the implementation of the basic Feed Forward ANN Model is shown as described in Table 4.1 that has two (2) hidden layers (32 and 16 neurons) with four (4) input nodes and one (1) output node. The EEG dataset will be divided in 80/20 % for training the ANN and testing the ANN through Cross Validation Approach shown in Figure 3.12, then calculate the accuracy and loss through the Equation C.61 and C.62, respectively. Furthermore, the comprehensive details are in the Appendix D.3.

Table 6.1 is the summary of the accuracy of using PCA-SRP in ANN in comparison to ANN alone; it shows in Figure 6.19 the accuracy of each EEG channel, which increases by **up to 7%** more. As observed in Figure 6.19, all the model accuracies in all channels in the ANN alone give approximately the same value (45 - 46%). Moreover, it signifies that all the channels' EEG bands behave without PCA-SRP but with slightly different model accuracy magnitudes. Compared to the ANN using PCA-SRP, it removed different samples for each channel.

6.2.3 Artificial Neural Networks (ANNs) Bootstrapping

The ANN bootstrapping histogram in Figure 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, and 6.27 with 2000 repetitions shows a comparison of the accuracy of using Principal Component Analysis -Sample Reduction Process (PCA-SRP) in the L/R Motor Movement and Imagery (MM) EEG dataset, in comparison without said data cleaning process. It also shows and follows the normal distribution graph as discussed in Chapter 3.6.

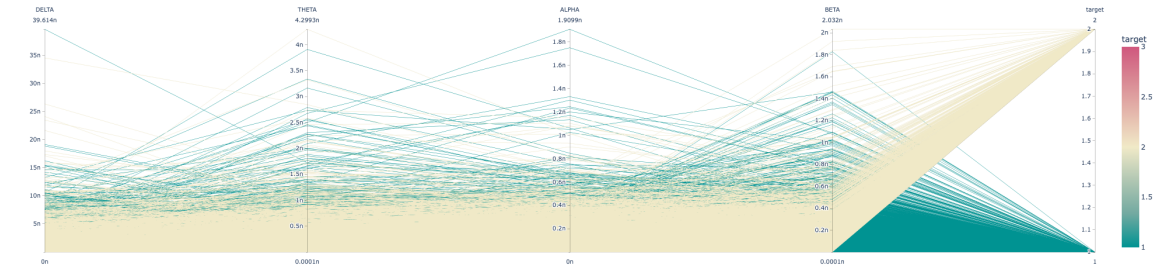
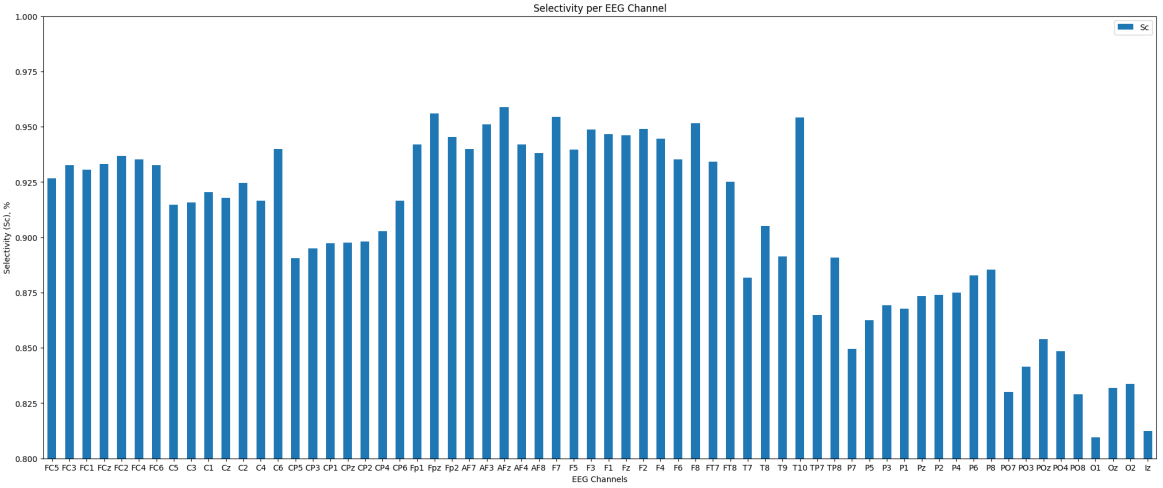
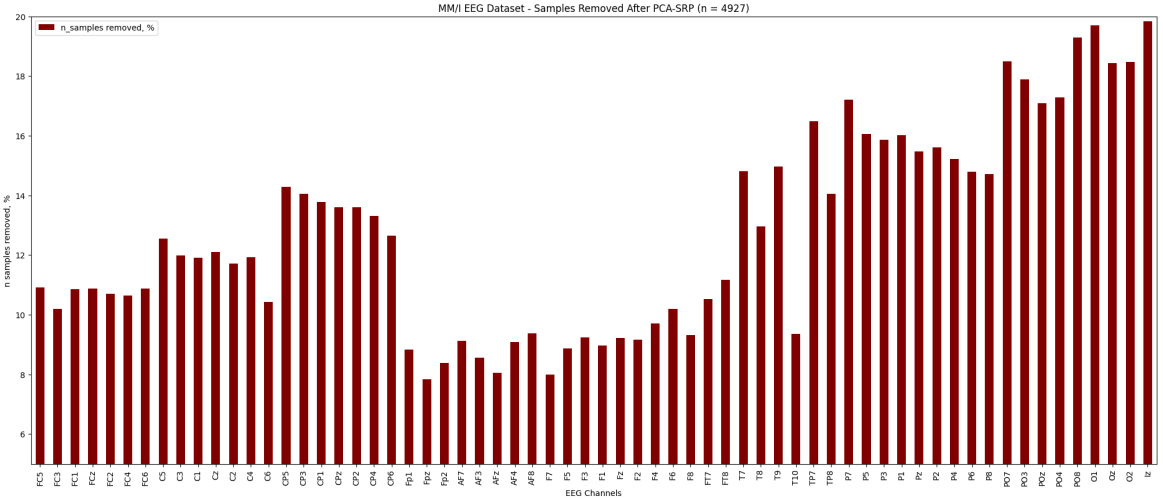


Fig. 6.17 L/R MM Dataset Continuous Parallel Plot



(a) Selectivity (Sc = PC_1)

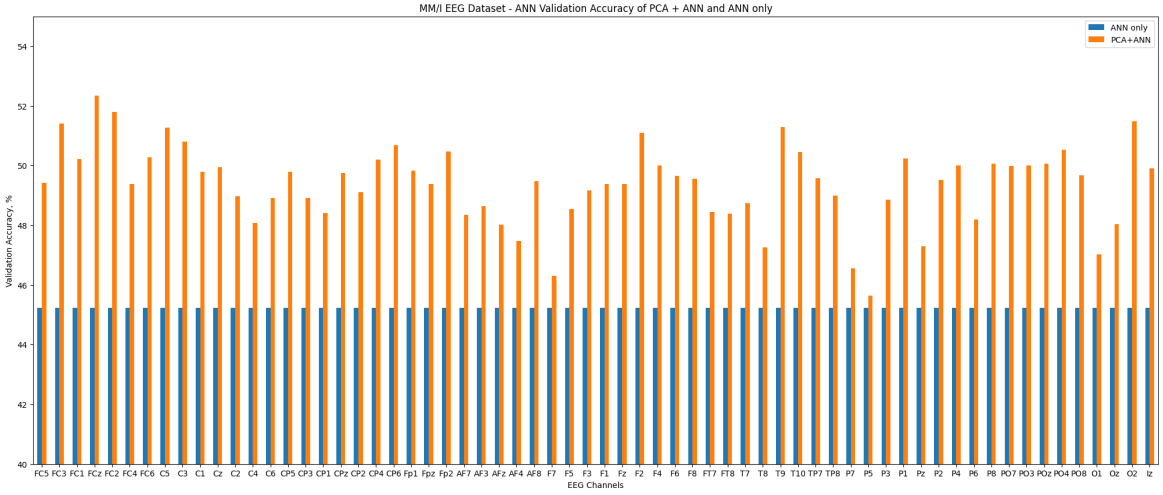


(b) Samples Removed (n = 4927 samples/epochs)

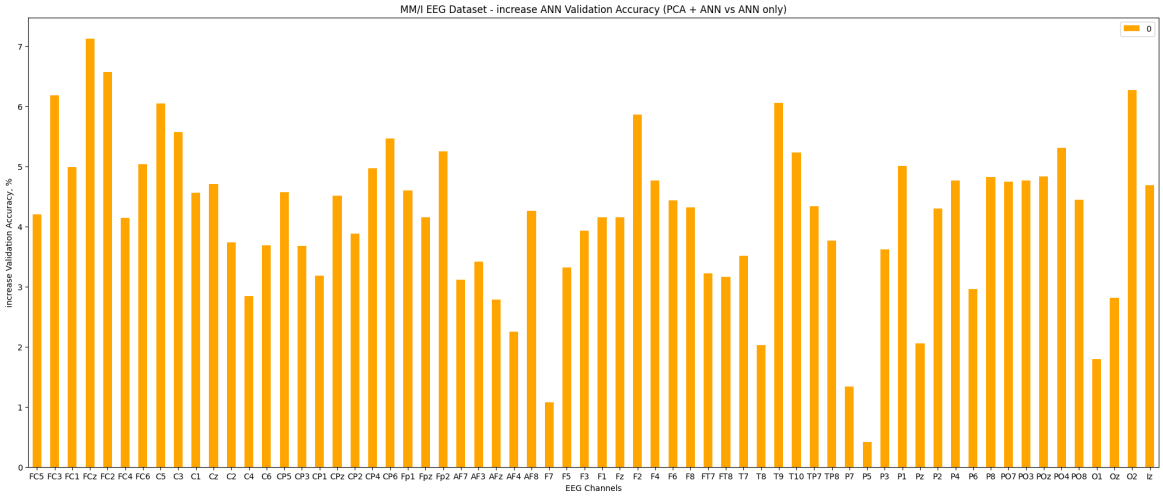
Fig. 6.18 Selectivity (Sc) and Number of samples Removed (%)

Table 6.1 ANN Model Validation Accuracy Increase ($PCA_{SRP} + ANN$ vs ANN only)

EEG Channels	Accuracy Increased, %	EEG Channels	Accuracy Increased, %
FC5	4.197260737	**F1**	4.153579473
FC3	6.17916286	**Fz**	4.152211547
FC1	4.989714563	**F2**	5.860487133
FCz	7.119411051	**F4**	4.766735435
FC2	6.566733539	**F6**	4.431143552
FC4	4.142445326	**F8**	4.319307208
FC6	5.039774299	**FT7**	3.224786758
C5	6.042835116	**FT8**	3.159430444
C3	5.573186278	**T7**	3.509591162
C1	4.563970447	**T8**	2.025477767
Cz	4.70906496	**T9**	6.055517554
C2	3.732252121	**T10**	5.225346655
C4	2.840468049	**TP7**	4.341462255
C6	3.690856695	**TP8**	3.763192892
CP5	4.567917407	**P7**	1.3353616
CP3	3.680548072	**P5**	0.4189103842
CP1	3.185558677	**P3**	3.620776534
CPz	4.513214469	**P1**	5.008283257
CP2	3.879412055	**Pz**	2.058451056
CP4	4.963226676	**P2**	4.295582741
CP6	5.461275101	**P4**	4.766735435
Fp1	4.603221178	**P6**	2.95721209
Fpz	4.156173319	**P8**	4.826191068
Fp2	5.250675708	**PO7**	4.749322653
AF7	3.114949018	**PO3**	4.766735435
AF3	3.418233395	**POz**	4.827937484
AFz	2.779981494	**PO4**	5.307840109
AF4	2.251111269	**PO8**	4.440101981
AF8	4.257216692	**O1**	1.797038347
F7	1.073241234	**Oz**	2.816486359
F5	3.319072723	**O2**	6.264250249
F3	3.928747773	**Iz**	4.685722709



(a) EEG Channels: ANN Model Validation Accuracy(PCA-SRP)+ANN VS ANN only



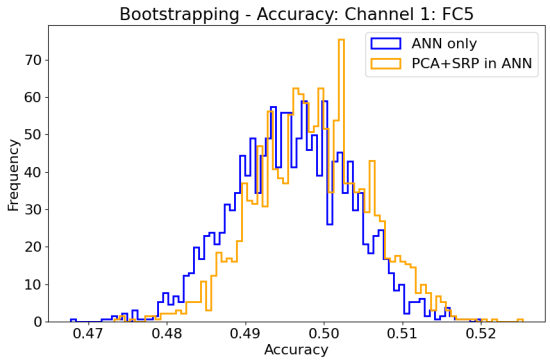
(b) EEG Channels: ANN Model Validation Accuracy Increased

Fig. 6.19 MM Dataset - EEG Channels: ANN Model Validation Accuracy Increased

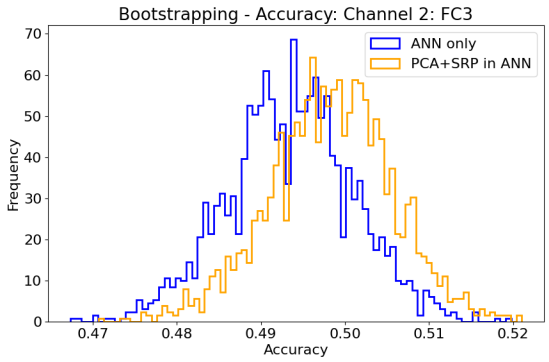
As observed, for example, in Figure 6.20b, the dataset used by PCA-SRP was tilted to the right, which means the majority of the 2000 repetitions, the Model Accuracy is higher than with the incorporation of PCA-SRP. However, Figure 6.27a appears that ANN Bootstrapping results have no apparent skewness either to left or right, which signifies that PCA-SRP has such a minimal impact on the ANN. Moreover, Figure 6.26h manifests a minimal skewness to the left that signifies a negative Model Accuracy with the incorporation of PCA-SRP.

ANN Bootstrapping results show that most of the EEG channels are skewed to the right, which signifies the increase in accuracy using PCA-SRP compared to not using it. Only PO_7 shows skewed to the left or negative ANN Bootstrapping result, while some Central, Parental, Temporal, and Occipital Lobes electrodes (C_4,CP_3,CP_4,CP_5,T_10,TP_7,P_2, P_3,P_4,P_5,P_7, P_8, PO_3, 0_1, and I_z) manifest without any changes in the ANN Bootstrapping result, and the rest of the electrodes appear skewed to the right or positive ANN Bootstrapping Accuracy results.

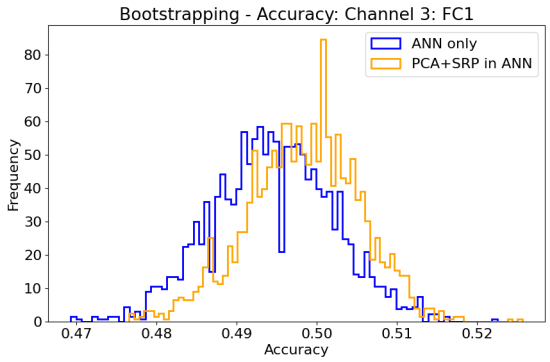
Since the discussion is about the L/R Motor Movement EEG dataset, the region responsible is the Frontal and Pre-frontal Lobes of the brain; it shows significant improvement in ANN bootstrapping Model accuracy results with the utilisation of PCA-SRP.



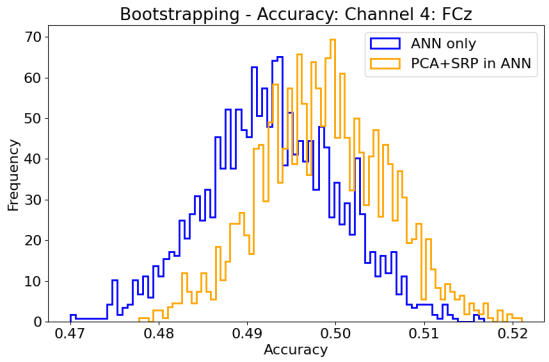
(a) Channel 1 FC5



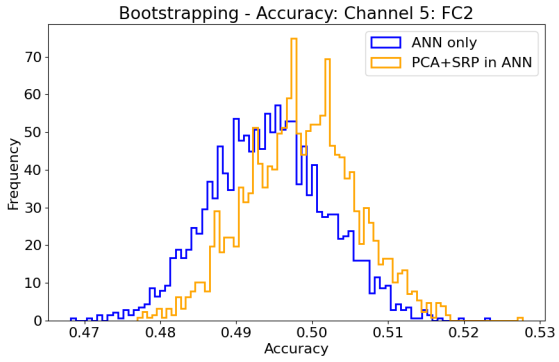
(b) Channel 2 FC3



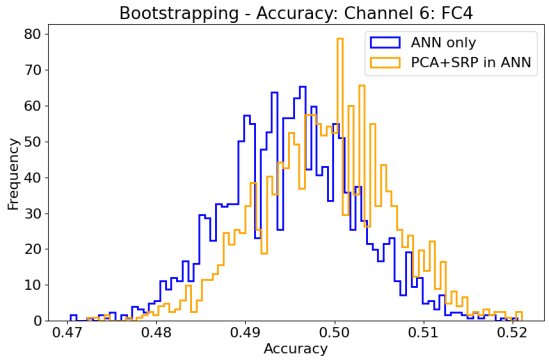
(c) Channel 3 FC1



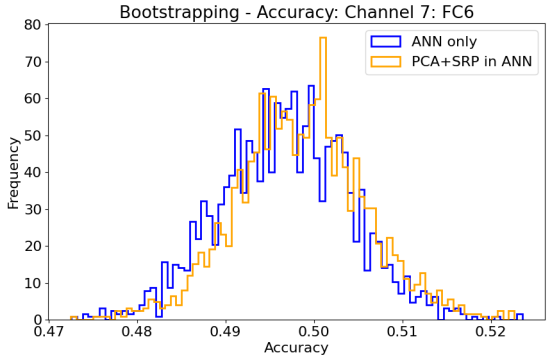
(d) Channel 4 FCz



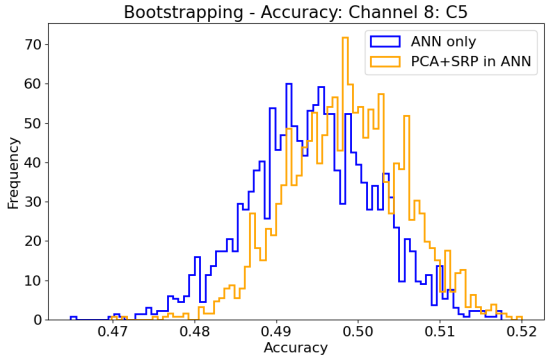
(e) Channel 5 FC2



(f) Channel 6 FC4

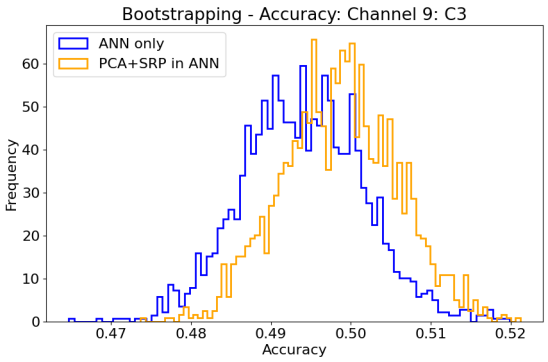


(g) Channel 7 FC6

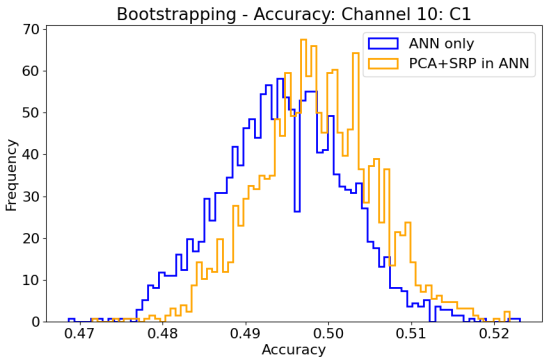


(h) Channel 8 C5

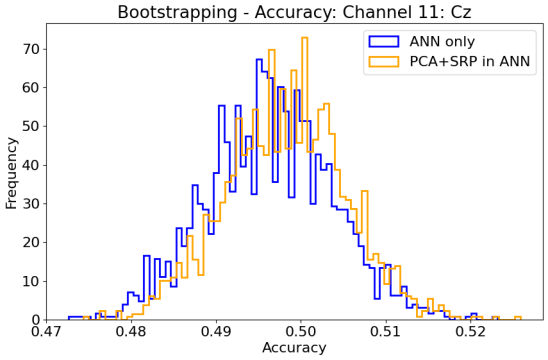
Fig. 6.20 Bootstrapping - Accuracy: Channel 1 to 8



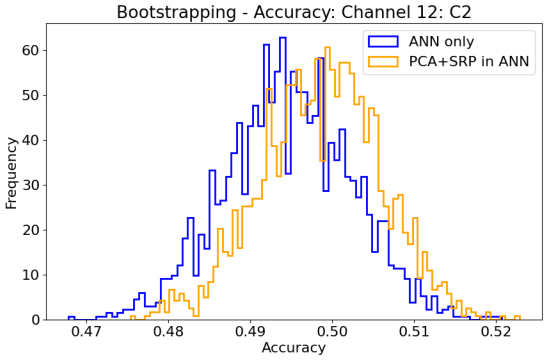
(a) Channel 9 C3



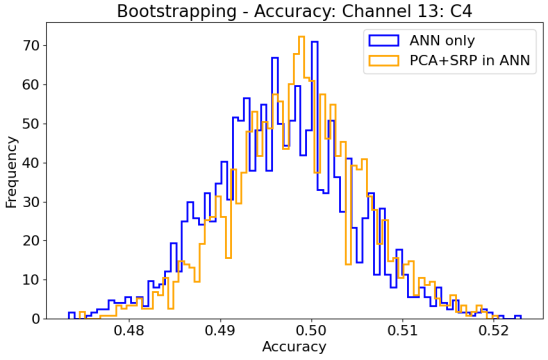
(b) Channel 10 C1



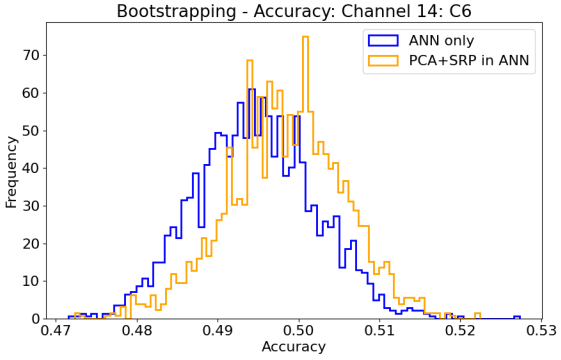
(c) Channel 11 Cz



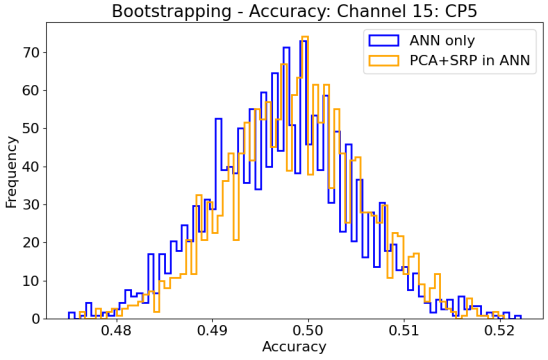
(d) Channel 12 C2



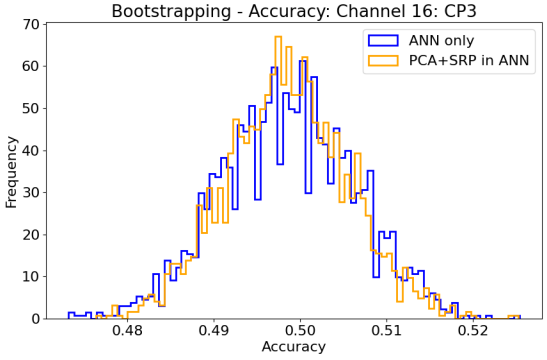
(e) Channel 13 C4



(f) Channel 14 C6

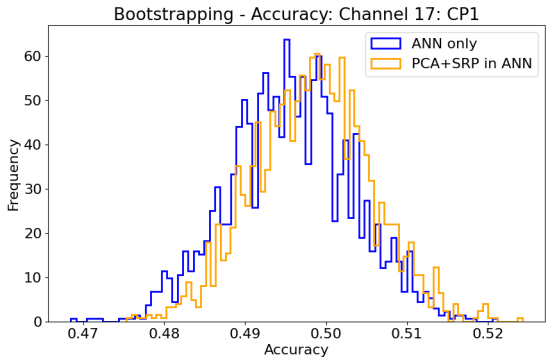


(g) Channel 15 CP5

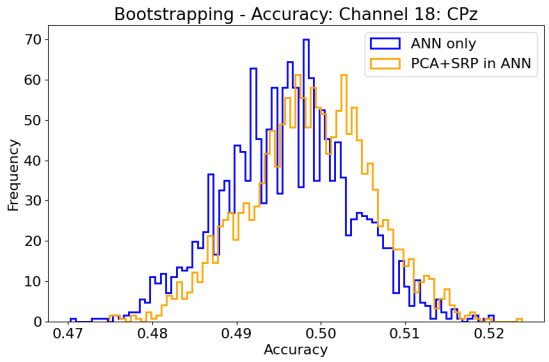


(h) Channel 16 CP3

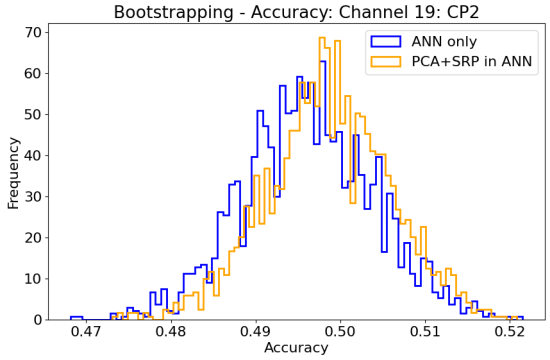
Fig. 6.21 Bootstrapping - Accuracy: Channel 9 to 16



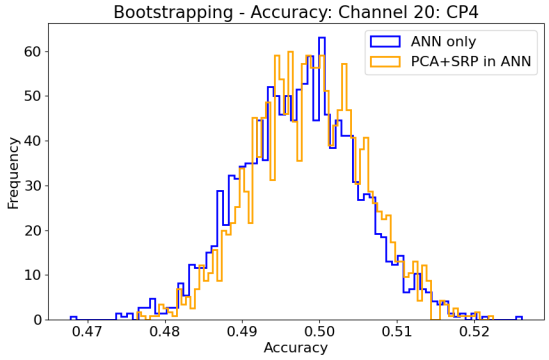
(a) Channel 17 CP1



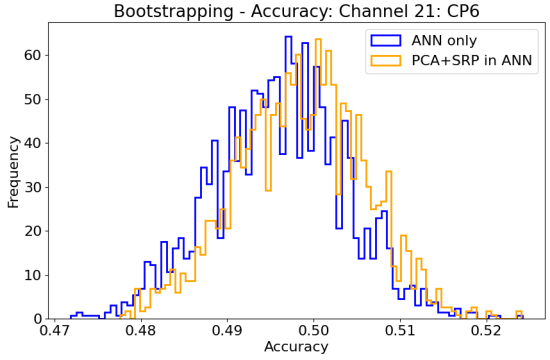
(b) Channel 18 CPz



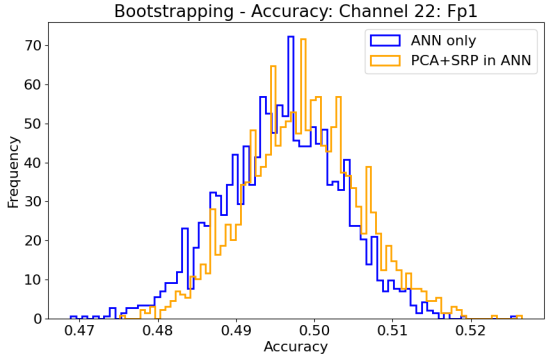
(c) Channel 19 CP2



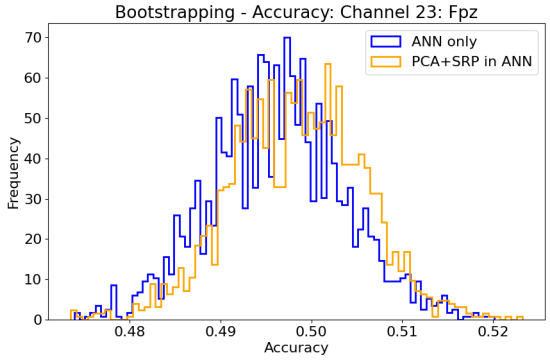
(d) Channel 20 CP4



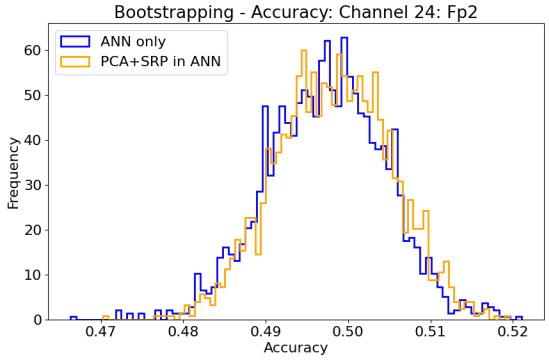
(e) Channel 21 CP6



(f) Channel 22 Fp1

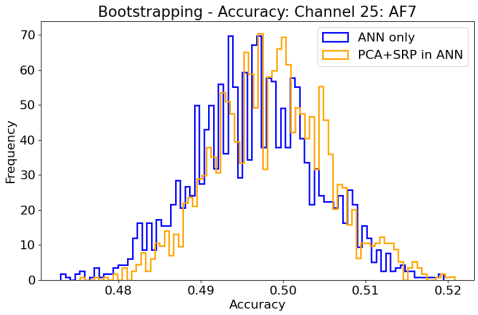


(g) Channel 23 Fpz

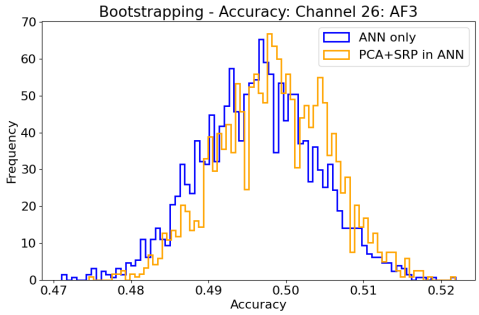


(h) Channel 24 Fp2

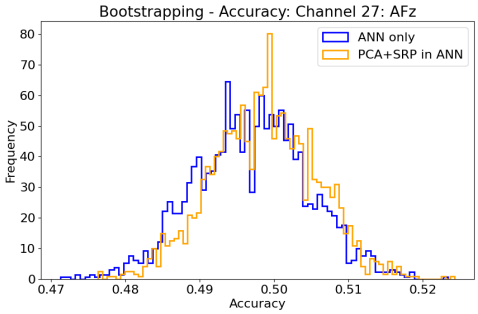
Fig. 6.22 Bootstrapping - Accuracy: Channel 17 to 24



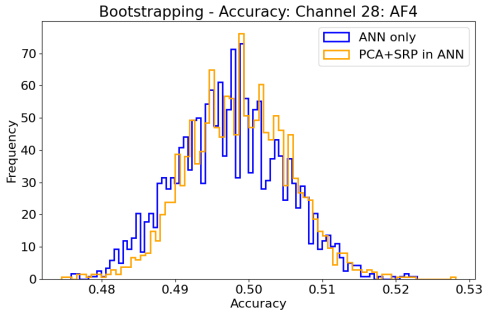
(a) Channel 25 AF7



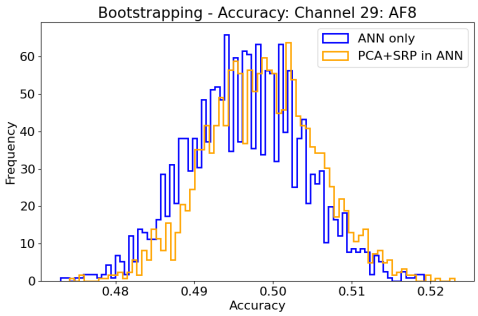
(b) Channel 26 AF3



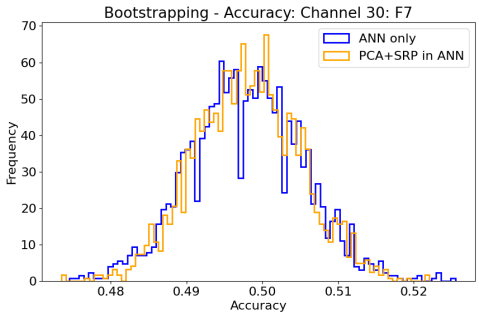
(c) Channel 27 AFz



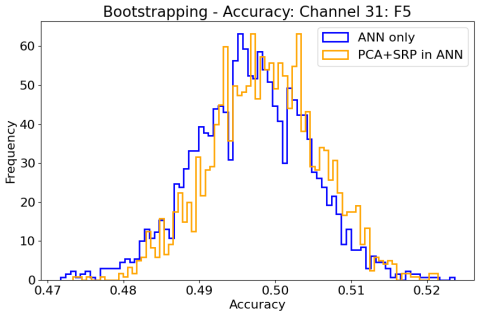
(d) Channel 28 AF4



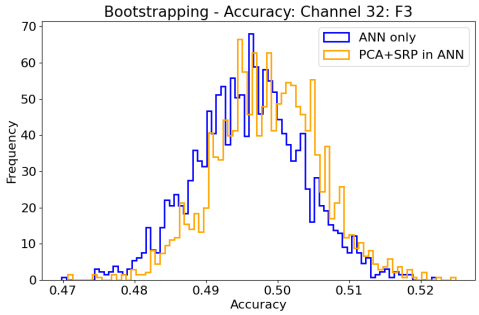
(e) Channel 29 AF8



(f) Channel 30 F7

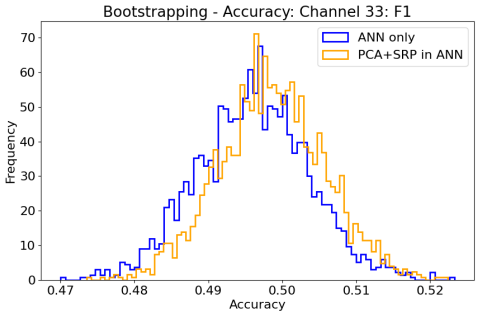


(g) Channel 31 F5

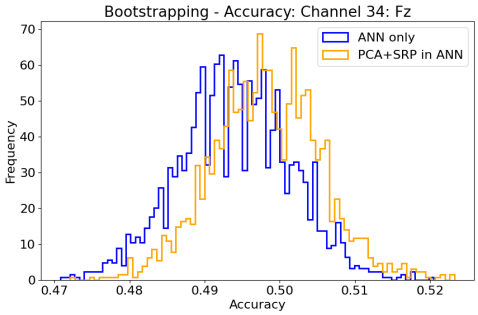


(h) Channel 32 F3

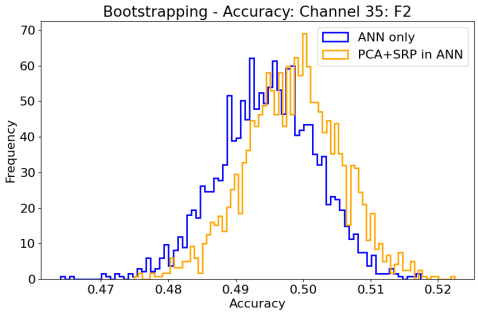
Fig. 6.23 Bootstrapping - Accuracy: Channel 25 to 32



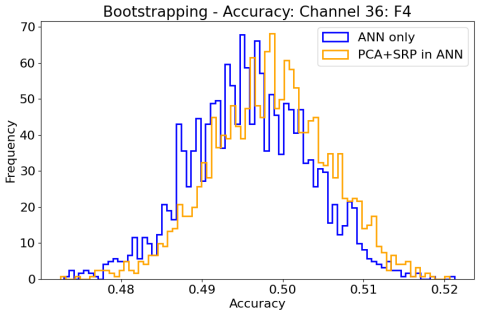
(a) Channel 33 F1



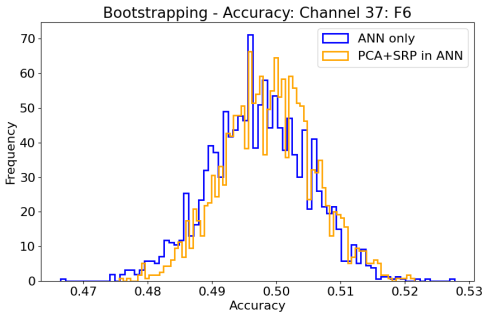
(b) Channel34 Fz



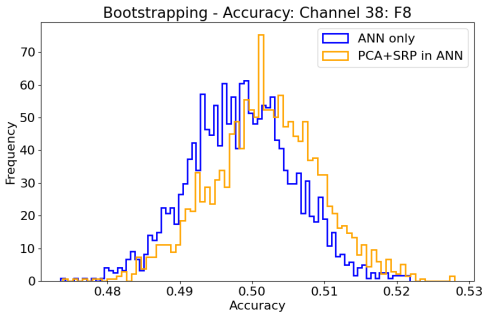
(c) Channel 35 F2



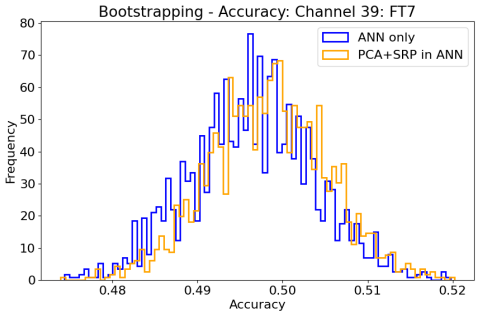
(d) Channel 36 F4



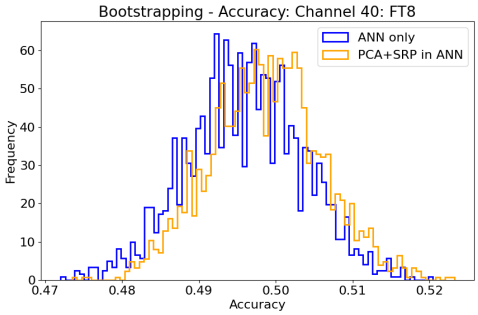
(e) Channel 37 F6



(f) Channel 38 F8

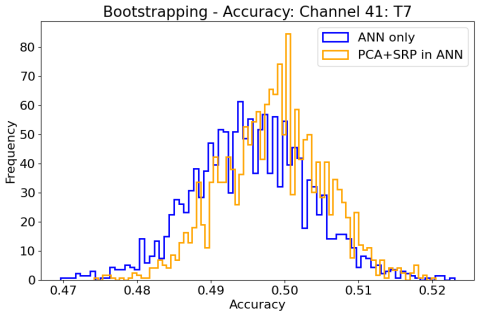


(g) Channel 39 FT7

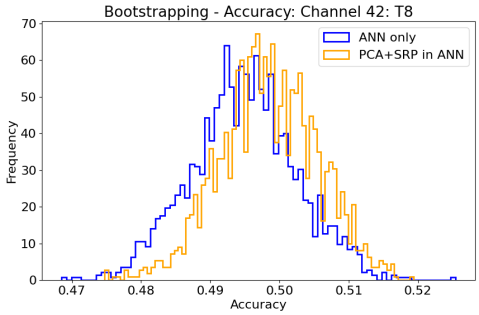


(h) Channel 40 FT8

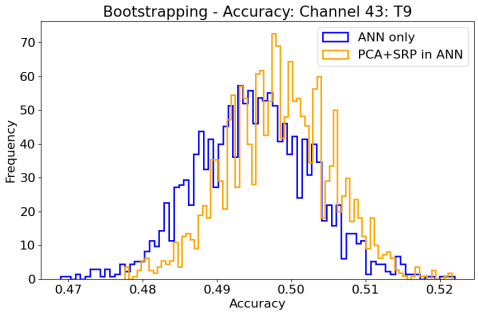
Fig. 6.24 Bootstrapping - Accuracy: Channel 33 to 40



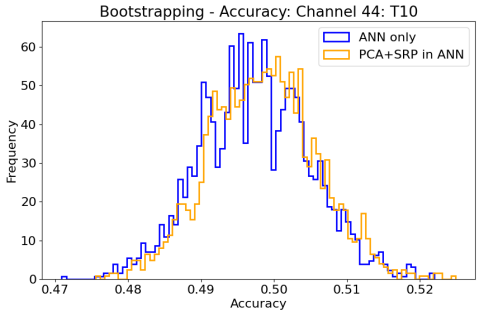
(a) Channel 41 T7



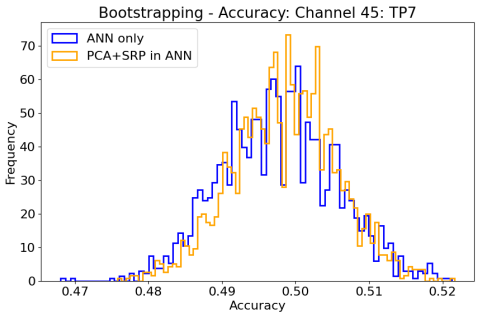
(b) Channel 42 T8



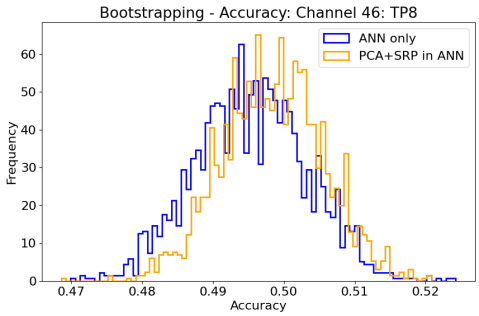
(c) Channel 43 T9



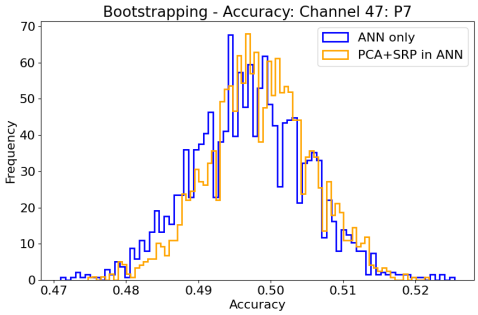
(d) Channel 44 T10



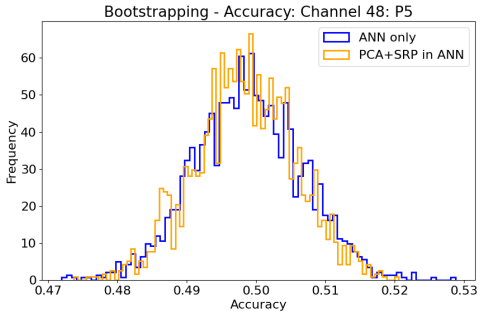
(e) Channel 45 TP7



(f) Channel 46 TP8

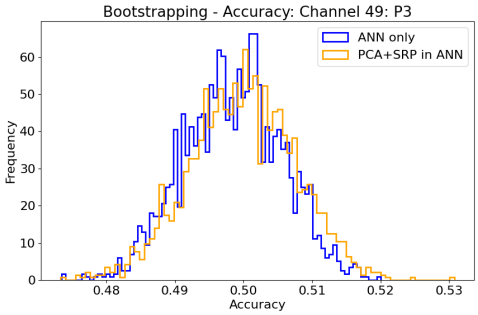


(g) Channel 47 P7

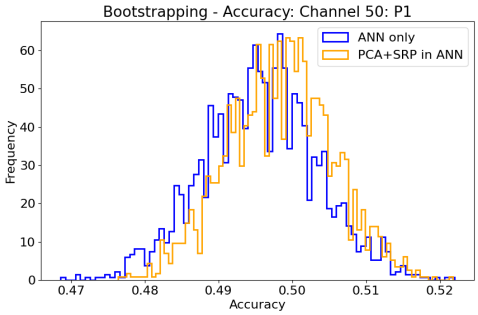


(h) Channel 48 P5

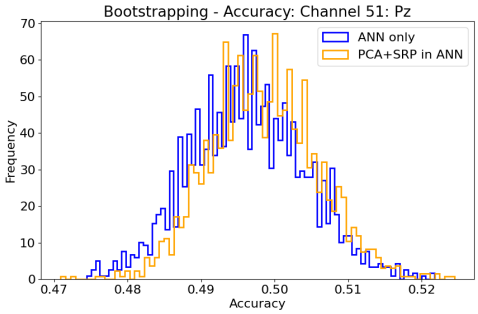
Fig. 6.25 Bootstrapping - Accuracy: Channel 41 to 48



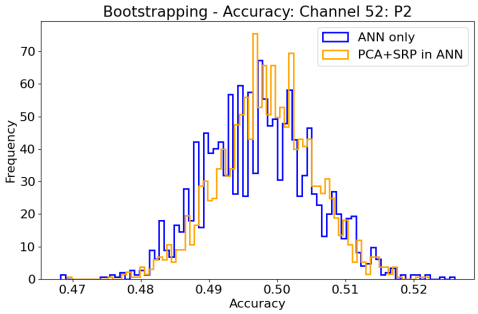
(a) Channel 49 P3



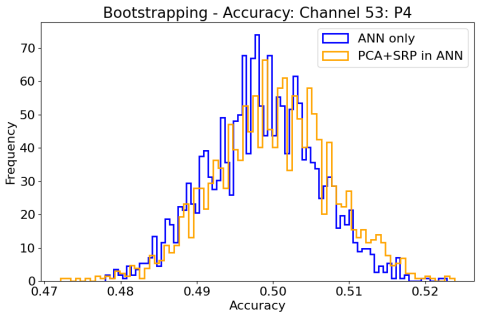
(b) Channel 50 P1



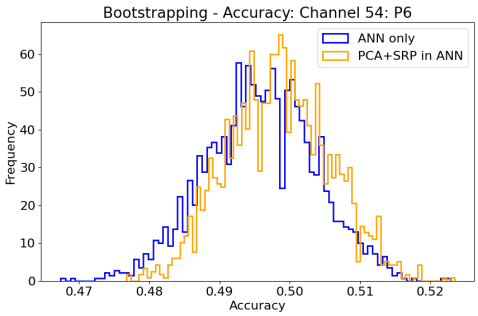
(c) Channel 51 Pz



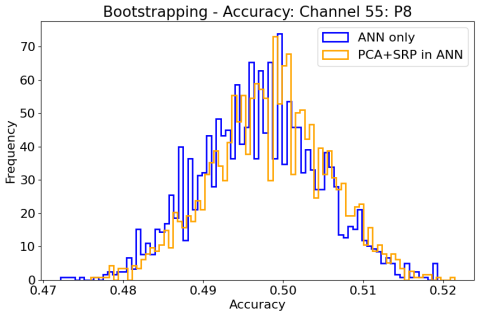
(d) Channel 52 P2



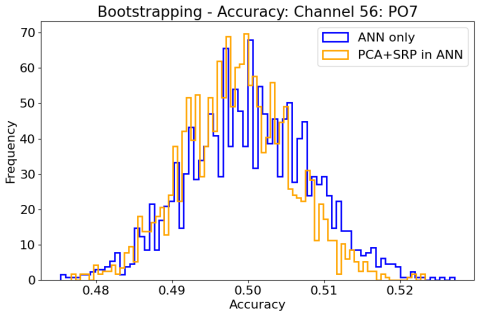
(e) Channel 53 P4



(f) Channel 54 P6

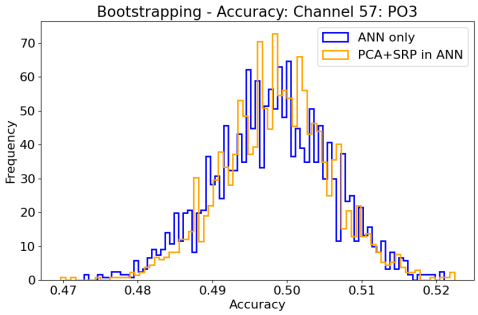


(g) Channel 55 P8

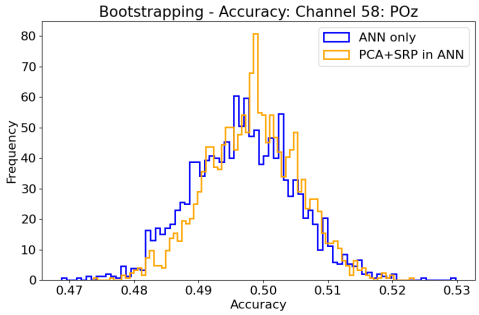


(h) Channel 56 PO7

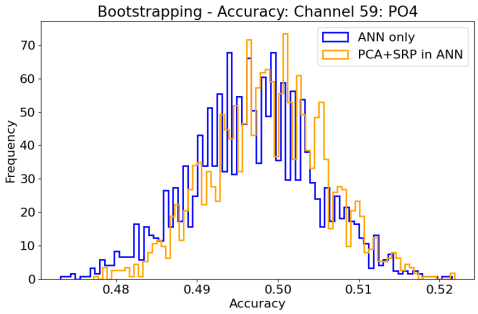
Fig. 6.26 Bootstrapping - Accuracy: Channel 49 to 56



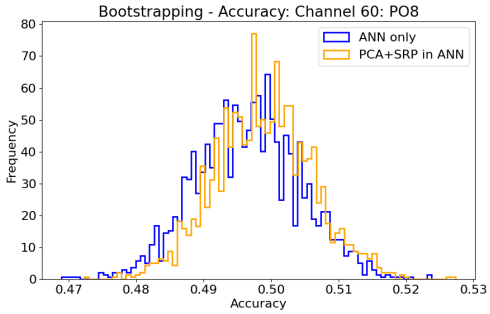
(a) Channel 57 PO3



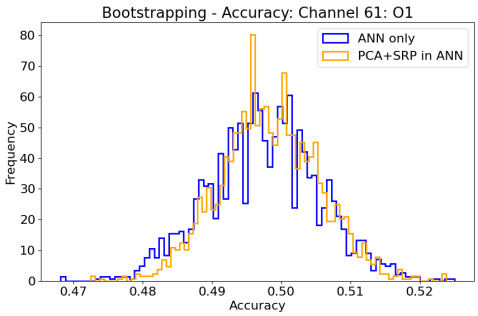
(b) Channel 58 POz



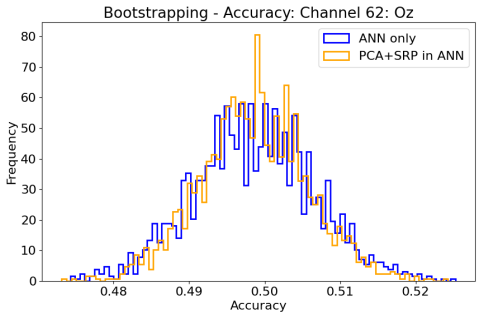
(c) Channel 59 PO4



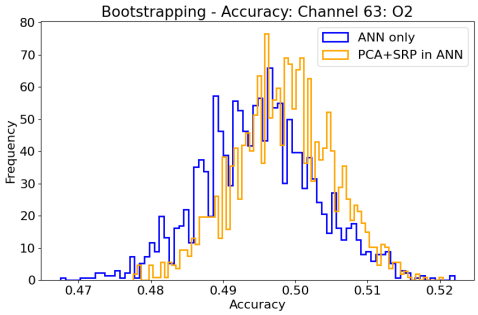
(d) Channel 60 PO8



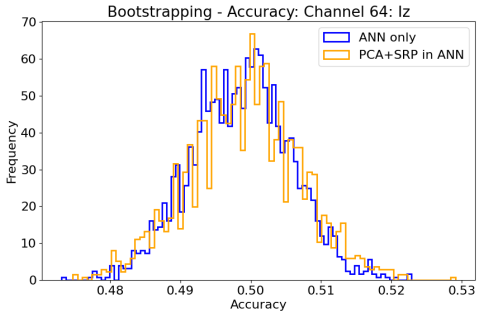
(e) Channel 61 O1



(f) Channel 62 Oz



(g) Channel 63 O2



(h) Channel 64 Iz

Fig. 6.27 Bootstrapping - Accuracy: Channel 57 to 64

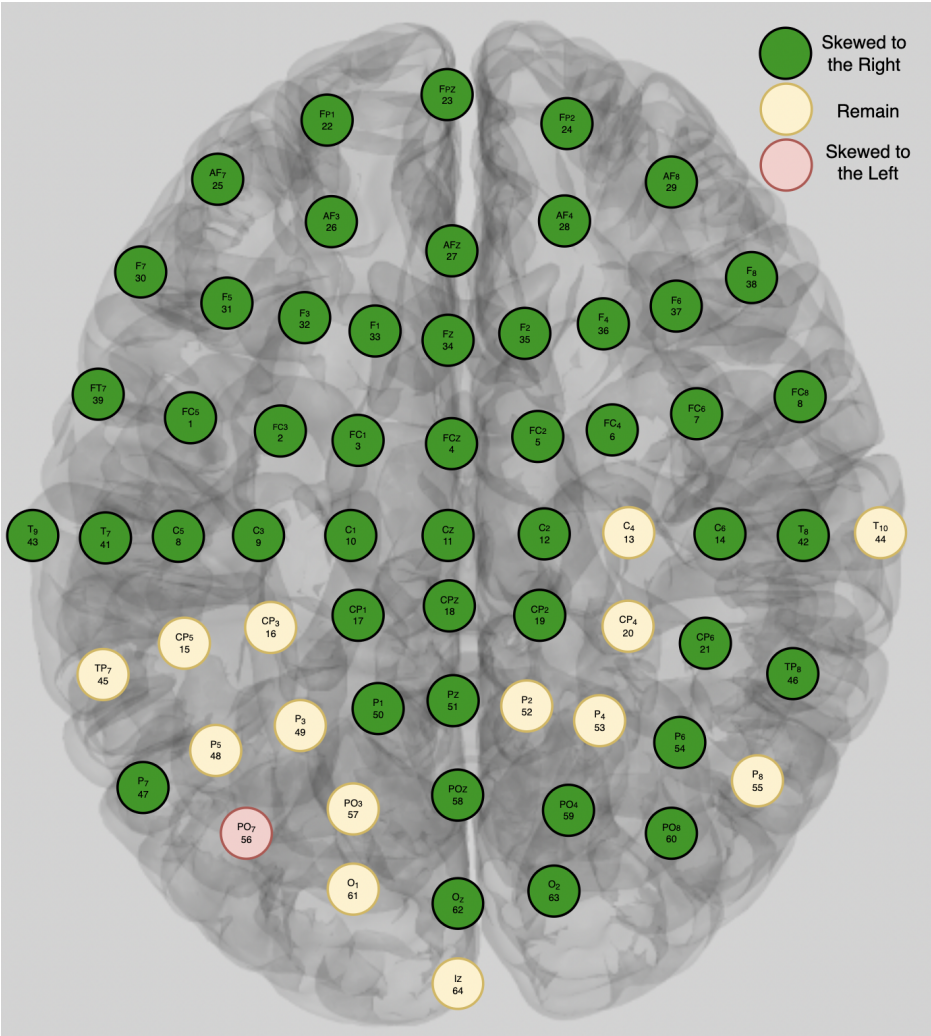


Fig. 6.28 Cranial Topography - ANN Bootstrapping Result

6.3 Conclusion

In this work, the impact of cognitive mental activity - Motor Movement and Imagery (Left - Right Hands and Feet Movement) on the performance of basic Feed Forward ANN Model using Fast Fourier Transform (FFT), Power Spectral Densities (PSD) by Simpson - Welch Rule Approximation and pass through a bandpass filter of EEG frequency bands (δ, θ, α, and β).Hence, the Principal Component Analysis - Sample Reduction Process (PCA-SRP) has been implemented to remove the least predictive samples.

The work has investigated the electrical physiological lobotomy attributes of the L/R MM dataset and the impact of PCA-SRP for training and testing the networks. Here are the findings of this chapter:

- It verifies that the left and right hemispheres of the brain control the 'opposite' side of the body.
- The anomalies are imminent in EEG signals, just like unexpected signal responses due to an individual's unique brain wiring and electrical reaction and electrode fixture misalignment.
- The Motor Movement EEG Datasets shows **non fully deterministic**, some subjects imply different mental electrical cognition, and every sensor has **non-unified reference base signal**. Mixing the same activity but different epochs characteristics may alter the weight of nodes in the ANN during the training process.
- In PC space, samples are distributed near [0,0,0] regardless of their motor movement, which means their variance is very low, showing that some samples have the slightest relationship with other samples.
- Using $Sc = PC_1$ based in the Equation 3.4, filtered the least predictive samples below the PC_1. Moreover, it increased the accuracy by **up to 7%** as shown in Figure 6.19.
- By comparing the ANN Bootstrapping Model Accuracies incorporation of PCA-SRP, it shows the majority of the region of the brain shows significant improvement, especially the region responsible for L/R Motor Movement such as Pre-Frontal and Frontal Lobes.

The overall conclusion is that there is a substantial increase in accuracy in using Principal Component Analysis - Sample Reduction Process (PCA-SRP) to cleanse the Motor Movement EEG dataset for training the basic Feed Forward ANN model. The flexibility of the proposed mathematical method through proper choice of Selectivity (Sc) and its variation depends on the quality of predictive samples included. It also provides the automatic selection of highly predictive samples that could result in systems that are easier to develop, deploy and use in various applications.

Future work will be focused on evaluating the robustness of this approach when collecting data mixed with different mental EEG signals from low-cost EEG sensors and quantifying the amount of randomness that make the PCA-SRP feasible is not yet identified.

Chapter 7

PCA-SRP in Physionet's EEG L/R Motor Movement (MM/I) with P300 Oddball Random Auditory Dataset

This chapter involves the two EEG datasets with different mental activities – L/R Motor Movement and Imagery (MM/I) and P300 Oddball Datasets [162] as a random EEG dataset, to remove outliers and to detect the presence of P300 waves and if the Principal Component Analysis – Sample Reduction Process (PCA-SRP) works mathematically and observing the behaviour using standard Classification Performance Metrics. This chapter concerns only the Motor Movement and Imagery (MM/I) Dataset (Left – Right Hands and Feet Movement) – the thorough evaluation of the EEG subject samples and their brain behaviour—and the effect of PCA-SRP in training the simple Artificial Neural Network.

7.1 Experimental Framework

The experimental framework included in the thesis is presented in this chapter. The publicly available EEG datasets are described in Section 1, the random EEG dataset is in Section 2, and Section 3 is the comparative visualisation diagram of the datasets in Sections 1 and 2, respectively. The evaluation methods are introduced in Section 4, followed by a discussion of the results.

7.1.1 EEG L/R Motor Movement and Imagery (MM/I) Dataset

Given in the Chapter 5 – dataset Section 6.0.1. The PhysioNet EEG Motor and Imagery dataset – Left / Right Hands and Feet Movement was used, with 109 subjects, 1500 (one to

two minutes EEG recordings) across 64 channels based on 10-20 Systems. Moreover, all 109 subjects combined their EEG signal dataset to its corresponding 64 EEG channels, as shown in Figure 6.2.

7.1.2 P300 Oddball Paradigm EEG Dataset

The P300 event-related brain potential ERP was elicited with auditory stimuli in two tasks. The oddball paradigm presented both target and standard stimuli; the single-stimulus paradigm presented a target but no standard tone stimulus, with the inter-target interval the same as that for the oddball condition.

Examining the brain activity of comatose people is one of the auditory p300's most intriguing uses. When conducting this experiment on an unconscious person and displaying a variation of the p300 signal in their EEG, it is very suggestive that they may be able to be awakened. Hence, the experiment is called the "Consciousness Detector" [163] [164].

This EEG dataset has used randomness due to audio disruption that occurs in the individual brain signal while performing an L/R Motor Movement and Imagery (MM/I) by combining these two EEG signals; and the different brain response of this EEG dataset to motor movement (MM) because of the different branch as shown in the Figure 5.1. Motor movement is Spontaneous, while P300 is in the branch of Evoked Potential, according to the recording.

Experimental Protocol

In reaction to a sensory, cognitive, or motor event, the P300 signal is an event-related potential (ERP), appearing on an EEG as a quick single potential shift. The signal's peak of the P300 since it typically occurs 300 milliseconds after the stimulus as shown in Figure 7.2 [162].

Fig. 7.1 P300 Oddball Experimental Setup.

- **Digit Targets (default)**. Two blocks were run, each with 40 trials (32 control, eight target/oddball trials). The stimulus set is the digits 1-5, spoken aloud. The stimulus duration is 800 ms, with 1000 ms between stimulus offset and subsequent stimulus onset.

- **Tone Targets**. This configuration uses the same conditions, timing, and trial counts as the default. However, the stimulus set is a set of five sine wave tones ranging in pitch from 200 to 2000 Hz.

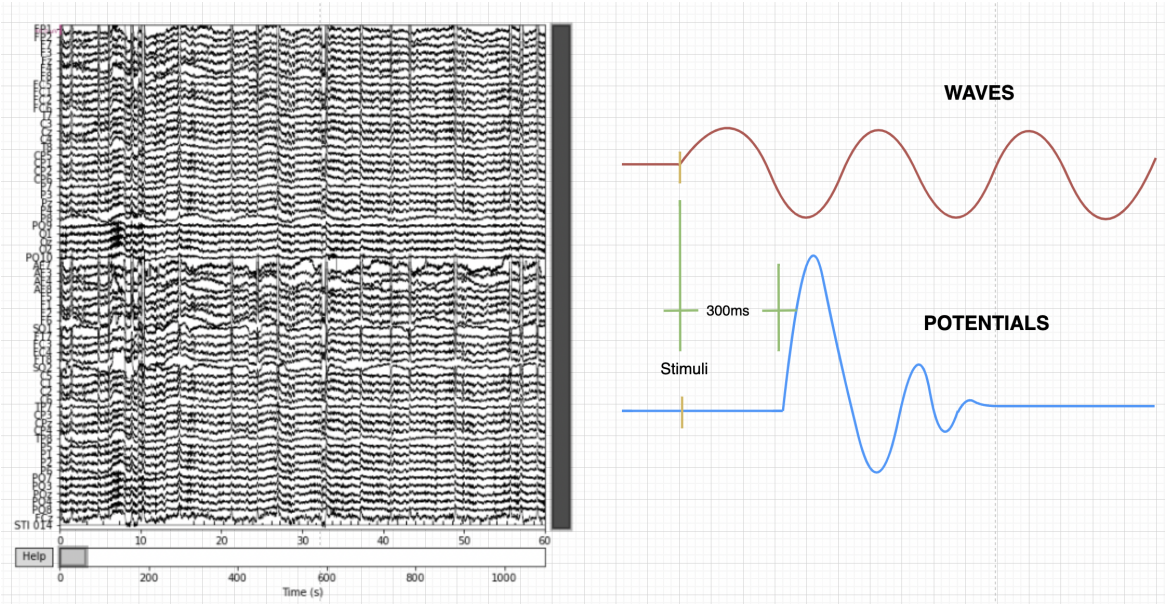


Fig. 7.2 P300 Event-Related Potentials.

In the following graphs, MNE [161] EEG Python processes describe and summarise the EEG L/R Motor Movement or Imagery (MM/I) and P300 Oddball Paradigm Dataset.

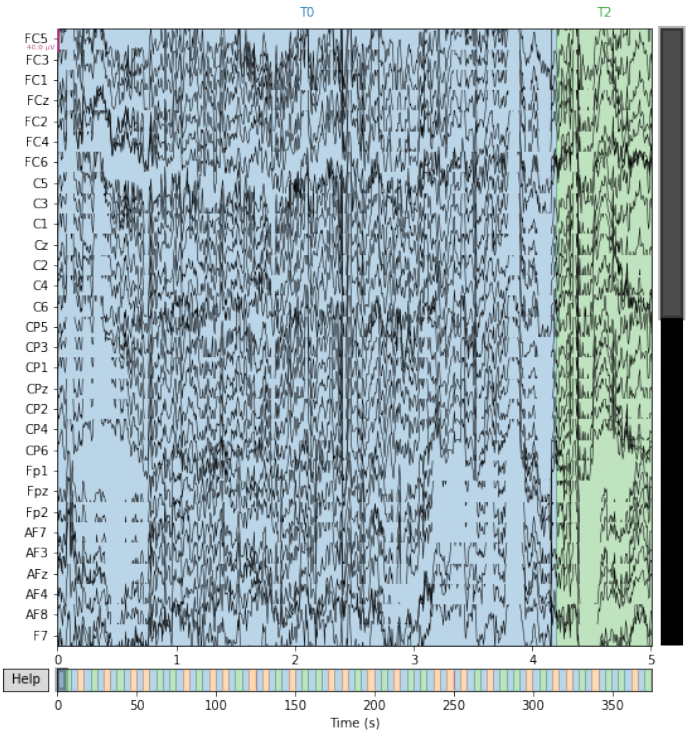
1. **Signal Waveform Diagram** that shows the Amplitude vs continuous time visualisation diagram of the EEG signal waveform of the particular electrodes concerning its EEG channel/s; it identifies the signal amplitude peaks and can compare it to other channels as shown in Figure 7.3. It appears that both signals have different reference signals and are biased.

2. **Power Spectrum Density (PSD) Diagram**, which shows the power in dB for the inclusion of EEG bands (0 – 50Hz) of each channel as shown in Figure 7.4. That manifested that both signals are different in the frequency response domain. Moreover, one of the frontal electrodes has a dissimilar and divergent frequency response from the rest of the other sP300 Oddball EEG signals.

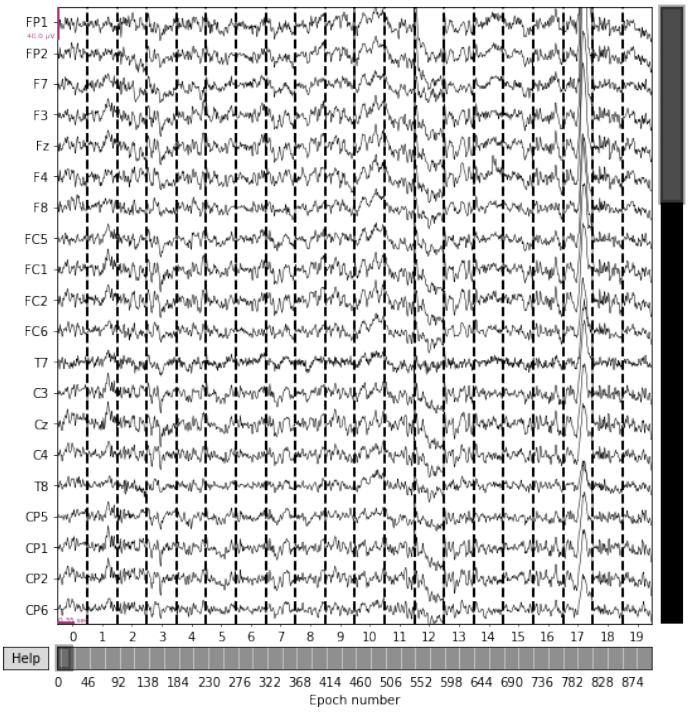
3. The **Event - Epoch ID Diagram** shows and visualises the time and epoch wherein the activity has been made as shown in Figure 7.5. It conveys a different epoch time window between L/R Motor Movement (MM) to P300 Oddball Dataset.

As shown in the Figures 7.4, 7.4, and 7.5 reveal that L/R Motor Movement (MM) and P300 Auditory EEG signals have different signal time and frequency signal responses, with the same time the epoch - time windowing.

The researcher chose the mixing of two different EEG signals for removing the other EEG signal (P300 Oddball Auditory) that has been tagged as "random" rather

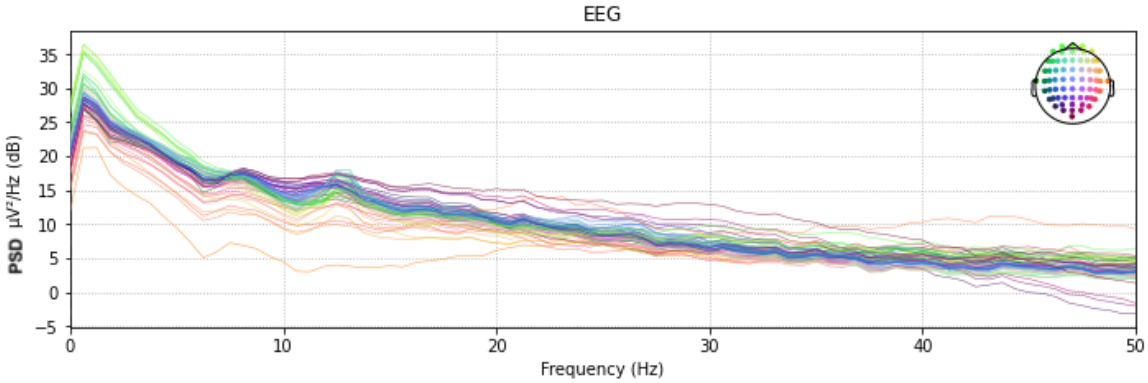


(a) MM/I Signal Waveform Diagram.

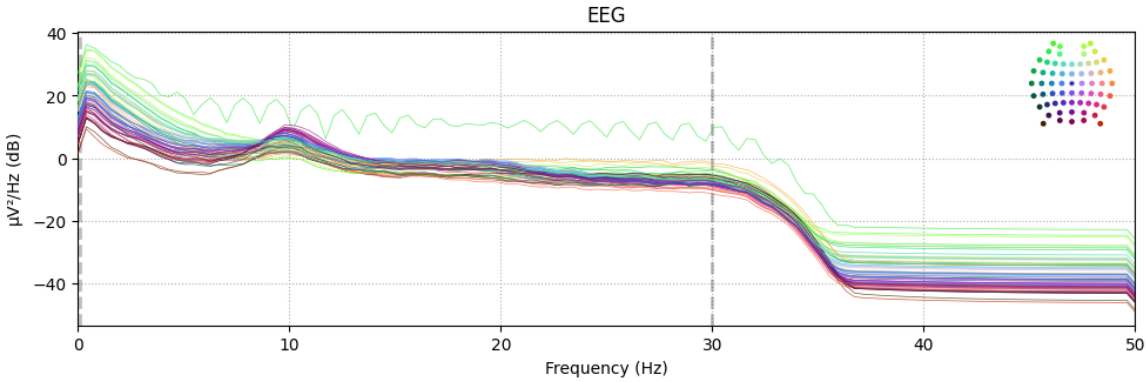


(b) P300 Oddball Auditory EEG Signal.

Fig. 7.3 Signal Waveform: L/R Motor Movement (MM) and Oddball EEG Signals



(a) L/R Motor Movement EEG Signal.



(b) P300 Oddball Auditory EEG Signal.

Fig. 7.4 Power Spectral Density (PSD): L/R Motor Movement (MM) and Oddball EEG Signals

than acquiring an EEG signal through actual EEG data gathering with ample subjects. Due to the time constraint, since the focus of this research is the cleaning of huge amounts of data is one of the difficulties that the researcher must have to resolve without sacrificing the proof of concept of the research study.

7.1.3 Experimental Procedure

Figure 7.6 shows the methodology of the experiment to determine the viability and effectiveness of the concept - Principal Component Analysis - Sample Reduction Process (PCA-SRP) in EEG datasets.

By mixing randomly distributed two different EEG signals and then extracting the features - (δ, θ, α, and β) using Fast Fourier Transform (FFT) in the conversion from time to frequency ($t \rightarrow s$) domain, Simpson's Estimation Method by easy approximating the definite integrals of different EEG signal bands, and lastly Welch's PSD calculation- to estimate the power spectra have been conferred in Chapter 5.2.

Employing the PCA-SRP, it will observe the behaviour of the said methodology in the given EEG dataset and yield some evaluation parameters such as True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

The Performance Evaluation Metrics estimates as discussed in Chapter 3.5 - Classification Model Evaluation Metrics; so, the significance classification is essential to test accuracy values that greatly exceed predetermined levels, theoretical levels of random categorisation are used as approaches and to give a complete description and behaviour of the PCA-SRP in randomly mixed distributed of the given EEG datasets.

7.2 Result and Discussion

This section discussed the result of the experiment by using and varying the Selectivity (Sc) in mixed EEG signal- L/R Motor Movement (MM) and P300 Oddball datasets. Table 7.1 conveys the combined FC_5 of all the 109 subjects as an example, it shows the epoch ID and its corresponding loading scores based on the computed Principal Component Analysis - Sample Reduction Process (PCA-SRP), and having a biased threshold - Selectivity (Sc) determine which samples will be in part of the train set.

Table 7.1 FC_5: Sorted Loading Score of MM/I and P300 Oddball (Random) Datasets

Sample / Epoch ID	Loading Scores
1688	0.014709
416	0.014709
4047	0.014709
2589	0.014709
234	0.014709
...	...
...	...
...	...
4509	0.000364
4664	0.000330
4371	0.000250
4436	0.000090
4571	0.000073

7.2.1 Evaluation by Visualization

One way to describe and verify the changes and behaviour of the dataset in an event is the complete depiction visually through **PC Space Scatter Plot**. Figure 7.7 shows a scatter plot in relation to its variance and scattering of the samples (both L/R Motor Movement and P300 Oddball randomness) in PC spaces ($PC1$, $PC2$, and $PC3$).

It also reveals its response in different Selectivity (Sc) in Principal Component Analysis - Sample Reduction Process (PCA-SRP) at Sc = 0 (without PCA-SRP), PC_1, 99%, and 99.85%. The blue dots signify the good samples (4358 L/R Motor Movement epoch samples), and the yellow ones are the random sample (approximately 436 P300 Oddball Auditory epoch samples). Most of the "random" samples are close to [0,0,0] space as expected because their variances are smaller than L/R Motor Movement epoch samples. The point of perspective revolves around different Sc due to the change of principal axes ($PC1$, $PC2$, and $PC3$).

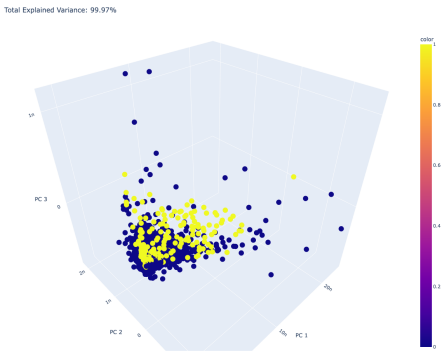
It has been observed that the different Selectivity (Sc) used in selecting the samples in the dataset changes the orientation and rotation of the Principal Component (PC) spaces. However, the location of each sample in the PC spaces has not changed relative to each distance from its neighbouring samples. It only means the PC_1, PC_2, ..., and PC_n change their rotational orientation in PC space depending on the general variance of each sample.

As the Selectivity increases in PCA-SRP, the number of epoch samples decreases for both EEG datasets. However, more epoch samples are removed in the P300 Oddball Auditory than in L/R Motor Movement EEG datasets. The first ones to fall short are the P300 Oddball samples, as observed in Figure 7.7 and 7.8.

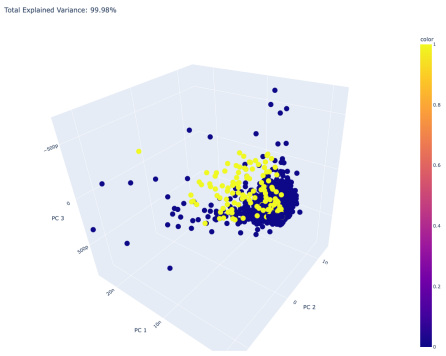
The **Parallel Coordinate Plot** shows in Figure 7.8 that the general connection of the samples to its features - EEG Frequency Bands (Δ, Θ, α, and β) and direction and magnitude of each sample. As observed, as the Sc increases, some of the L/R MM and P300 Oddball samples are lost in the dataset, but some MM/I samples also recovered.

The **Scatter Matrix Plot** shown Figure 7.9 verifies it by observing that most P300 Oddball random samples are concentrated in specific locations (approximately lower left of the scatter matrix) shown in Figure 7.7 that correspond in very low variance - nearly close to (0,0,0) in PC space, and also decreasing as the *Sc* increases.

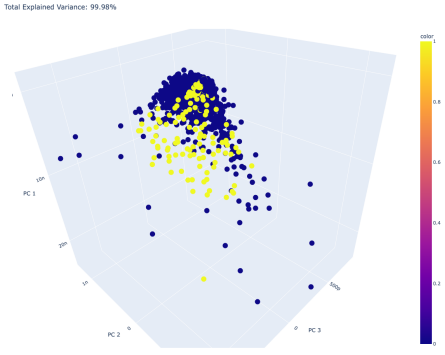
PC space Scatter Plot, Parallel Coordinate Plot, and Scatter Matrix Plot is utilized to clearly define the two EEG datasets (Motor Movement and P300 auditory dataset)in different perspective such as in PC space, the feature's general connection to its sample and concentration of low variance sample in PC space.



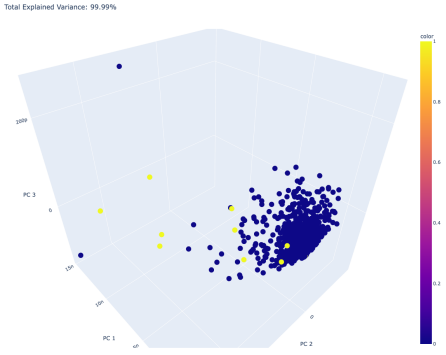
(a) EEG Datasets - Scatter Plot (without PCA-SRP).



(b) EEG Datasets - Scatter Plot (Sc = PC_1)



(c) EEG Datasets - Scatter Plot (Sc = 99%).



(d) EEG Datasets - Scatter Plot (Sc = 99.85%).

Fig. 7.7 FC_5: Scatter Plot with Different Selectivity (Sc).

7.2.2 Confusion Matrix Table -TP, FP, FN, and TN Evaluation

This Confusion Matrix Table for the increasing Selectivity (Sc) shows how the behaviour of True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) by the random distribution of both MM/I (4358 epoch samples) and 10% P300 Oddball random datasets (435 epoch samples).

By observation, as the Sc increases, the number of P, TP, and FP decreases abruptly, and the number of FN, TN, and N increases steeply, nearly or approaching 100 % Sc. At 91 % Selectivity, FN samples start accumulating, and in 100 %

The proper choice of Selectivity is crucial and also depends on the priority:

- **minimise the number of FP** - means avoiding random data injected in the dataset and maximising the FN.
- **minimise the number of FN** - means including all good samples in the dataset and maximising the FP.

Therefore, Sc will be in the range of 94 to approximately 100%.

$$94 \leq Sc \leq 100 \tag{7.1}$$

Table 7.2 Confusion Matrix Table Vs Increasing Sc (with 10% Randomness - 435 samples)

Sc	P	TP	FP	FN	TN	N
0	4793	4358	435	0	0	0
1	4793	4358	435	0	0	0
2	4793	4358	435	0	0	0
3	4791	4358	433	0	2	2
4	4787	4358	429	0	6	6
5	4784	4358	426	0	9	9
6	4784	4358	426	0	9	9
7	4784	4358	426	0	9	9
8	4784	4358	426	0	9	9
9	4783	4358	425	0	10	10
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
91	4649	4358	291	1	143	144
92	4641	4358	283	1	151	152
93	4635	4357	278	2	156	158
94	4586	4317	269	42	165	207
95	4518	4259	259	100	175	275
96	4438	4188	250	171	184	355
97	4299	4067	232	292	202	494
98	4094	3888	206	471	228	699
99	3673	3494	179	865	255	1120
100	1	1	0	4358	434	4792

Performance Metrics

"All models are wrong, but some are useful", George Box

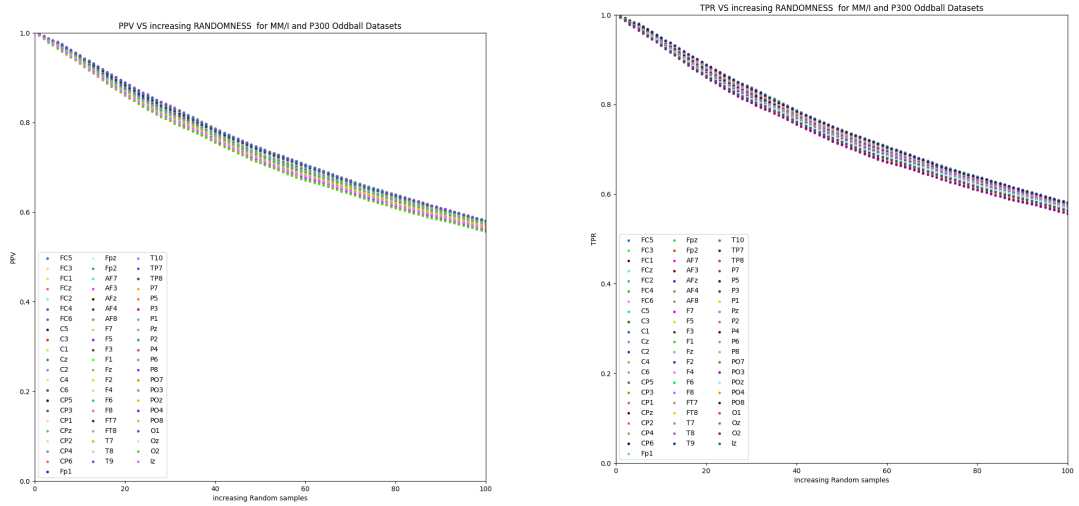
A supervised machine learning task known as classification aims to predict the category or class to which one or more observations belong. Evaluating the model's performance is a crucial step in any machine-learning process. Labelled data is used as the trained set to predict previously unobserved. Then, for classification, quantify these model predictions of how much is correctly identified.

As discussed in Chapter 5.2 - Classification Model Evaluation Metrics, Figure 7.10 shows the behaviour of classification accuracy in the increasing injected randomness at the first additional 10% of P300 Oddball to MM/I dataset. Moreover, after that, it will dive decreasingly as more random datasets are involved.

In practical classification problems, a model can only be partially accurate. So, it is essential to understand how "right" a model was and how "wrong" it was. Furthermore, accuracy is the evaluation that commonly to is used alone since it can be pretty deceptive when applied to imbalanced data. It has one class that is significantly bigger than another.

"RIGHTNESS" Classification Metric Evaluation

Figure 7.11 shows the decreasing of the Positive Predictive Value (PPV) and True Positive Rate (TPR) for all the EEG channels (64 channels) as the increasingly adding more random samples (P300 Oddball dataset) in the system. It depicted that PPV and TPR are losing their positive samples - L/R MM samples relative to the total number of samples, which is somehow not pleasing by interpretation. The accuracy drop implies the increase of FP in the cleaned data.



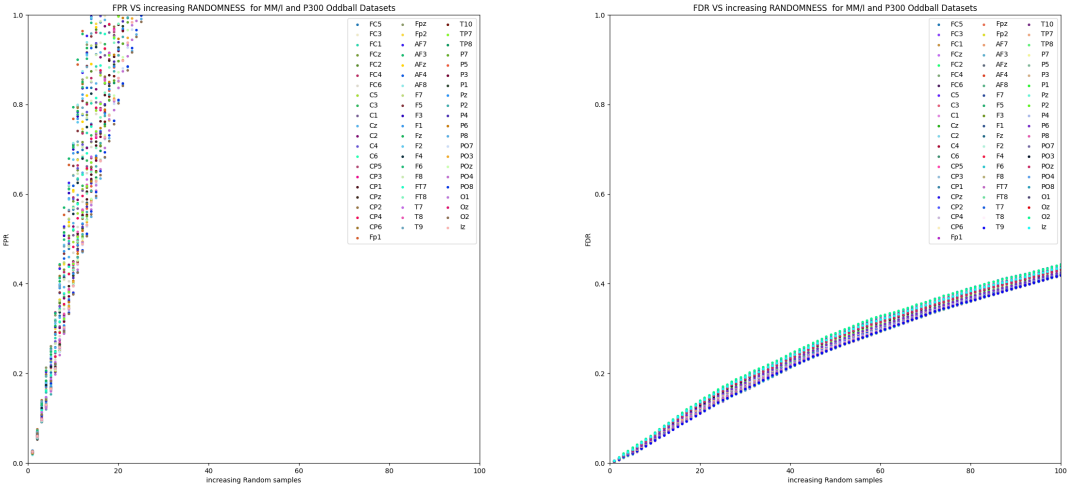
(a) Precision (PPV) vs increasing randomness

(b) True Positive Rate (TPR) vs increasing randomness

Fig. 7.11 "Rightness" Evaluations - PPV and TPR (Sc = PC_1)

"WRONGNESS" Classification Metric Evaluation

However, Figure 7.12 shows the increasing False Positive Rate and False Detection Rate for all the EEG channels (64 channels). This means increasing the P300 oddball random samples in the system. It evaluated that FPR and FDR are identifying more false positive samples added, which is undesirable for the classification training set.



(a) False Positive Rate (FPR) vs increasing Randomness (Sc = PC_1)

(b) False Detection Rate (FDR) vs increasing Randomness (Sc = PC_1)

Fig. 7.12 "Wrongness" Evaluations - FPR and FDR (Sc = PC_1)

Even though losing some of the positive samples (L/R MM) increases the additional random (P300 Oddball) samples as a drawback, the identification of negative samples is increasing, which might constitute a benefit or advantage; on the other hand, up until it meets the 20% additional random set.

Combined "Rightness" and "Wrongness" Classification Metric Evaluation

A machine learning assessment statistic called the **F1 Score** assesses a model's accuracy. It combines a Model's Precision and recalls ratings. **Precision** measures the correctness of the "positive" predictions; **Recall** measures the number of positive samples in the dataset that was identified correctly.

Ideally, both of the models identify all of our positive cases (Precision), and that is, at the same time, identify only positive cases (Recall) are necessarily called **Precision-Recall Tradeoff**.

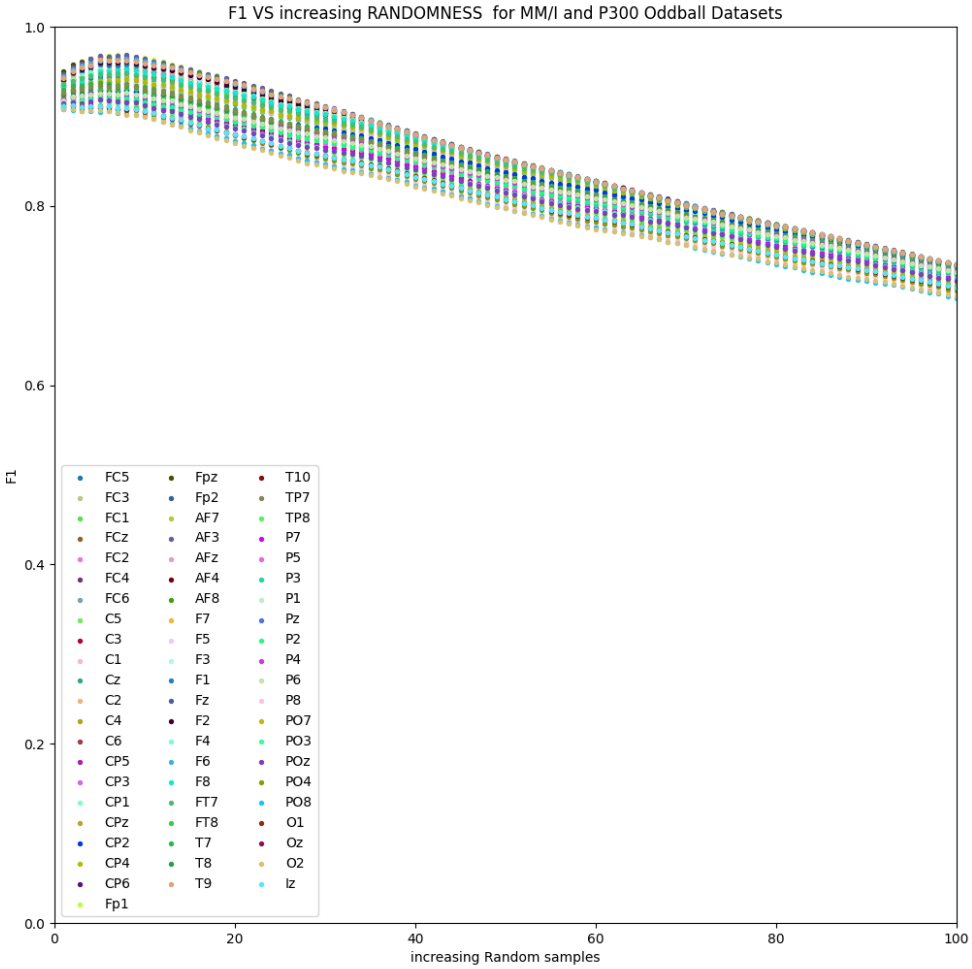


Fig. 7.13 F-Score Rate (F1) vs increasing Randomness (Sc = PC_1)

As Figure 7.13 shows, the quite increase in Precision-Recall tradeoff up until 10% increase of random samples, more than that gives us a decreasing tradeoff.

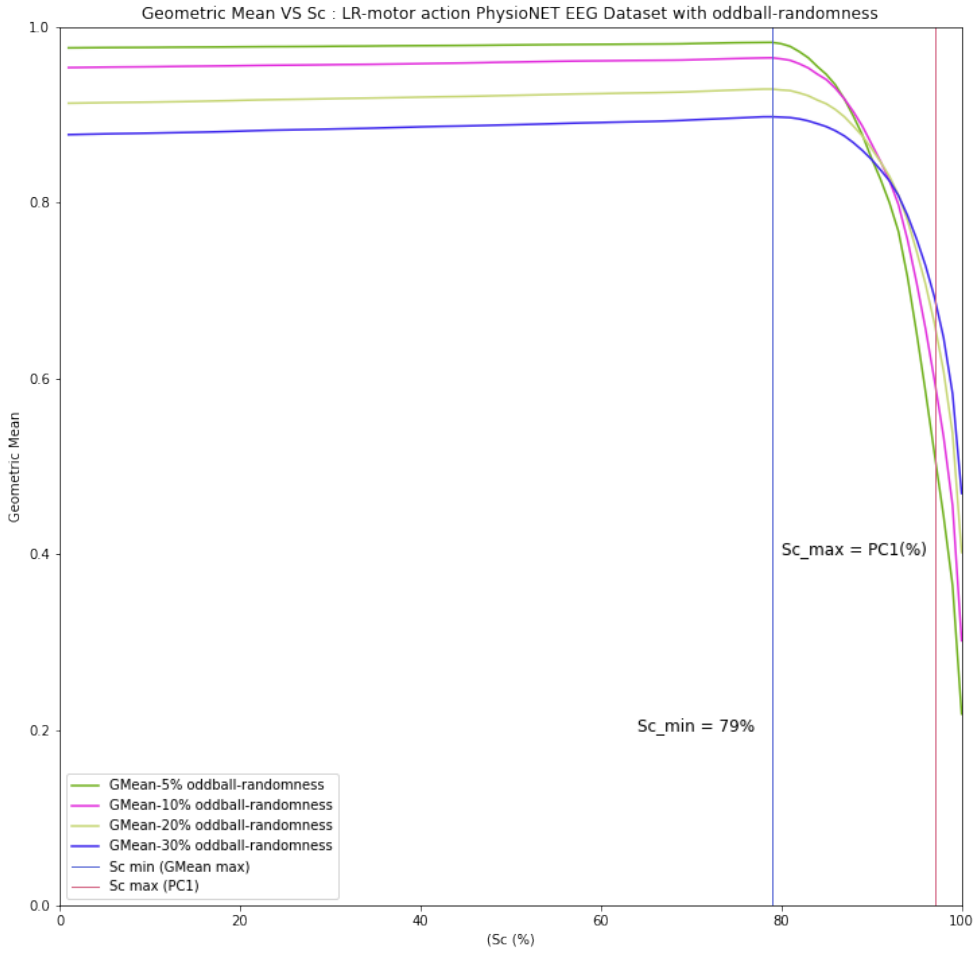


Fig. 7.14 Geometric Mean (Gmean) Rate vs Selectivity (Sc)

Using the **Geometric Mean (gmean)**, it measures tries to maximise the accuracy on each of the classes while keeping these accuracies balanced as the Figure 7.14 provides that the Selectivity range wherein Gmean max is identified at Sc = 79% in different additional random samples in the system and theoretical Sc = PC_1 that sets the standard limit of Sc.

As the increasing of random samples, Gmean behaves as the Figure 7.15, it shows that the Gmean peaked at 20% additional randomness and deteriorated afterwards for most of the EEG channels.

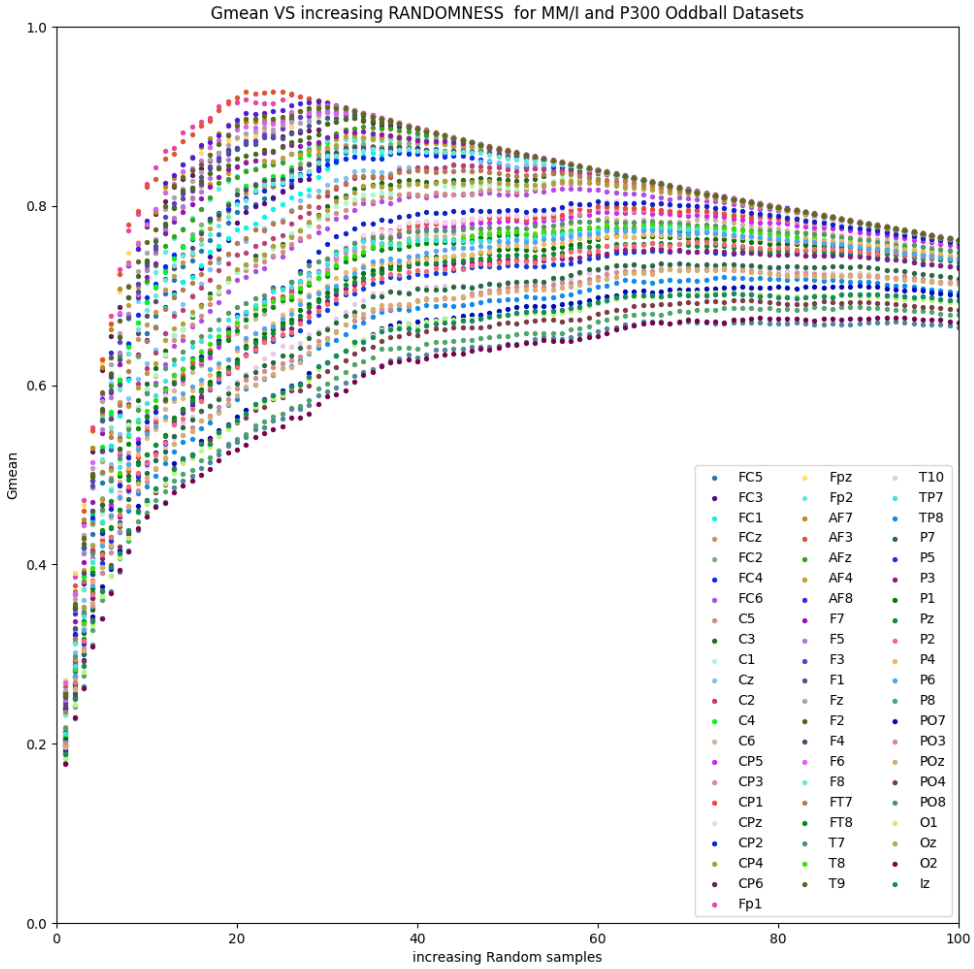


Fig. 7.15 Geometric Mean (Gmean) Rate vs increasing Randomness (Sc = PC_1)

By definition, **Cohen's Kappa Statistic** is a quantitative measure of reliability for two raters (positives and negatives) that are rating the same thing, corrected for how often the raters may agree by chance. It measures the number of predictions it produces that is not explicable by a random guess and eliminates the chance of the classifier and a random guess agreeing. Additionally, Cohen's kappa tries to correct the evaluation bias by considering the correct classification by a random guess.

It shows peaks at moderation agreement at 10% additional randomness, in a limit of approximately 0.18 to 0.55 for all the EEG channels. There is a sudden increase in Cohen's Kappa Statistic as it reaches the additionally injected 10 % randomness, which means the

agreement between the positives and negatives in random guessing is improving until 10 % and decreases afterwards. It is bearable to assume that it is acceptable up until a 15 % increase in randomness.

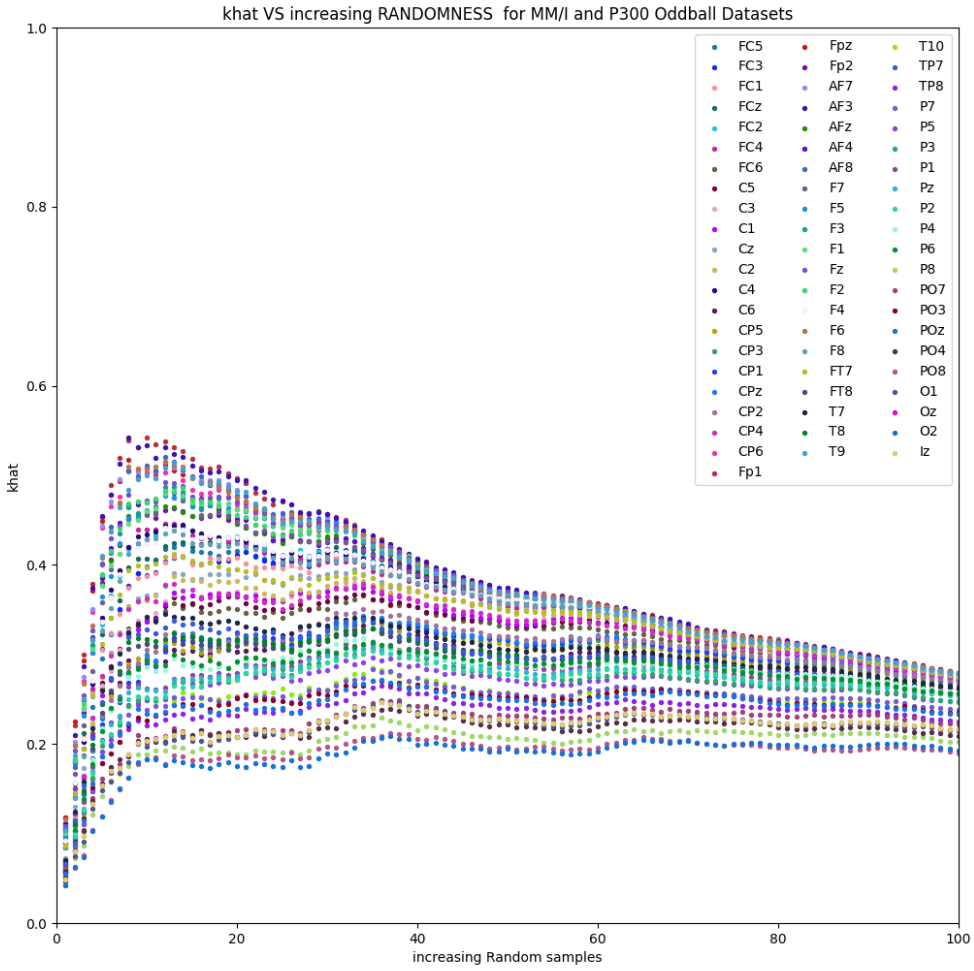


Fig. 7.16 Cohen's Kappa Statistic Rate (khat) vs increasing Randomness (Sc = PC_1)

As shown in Table 7.3, Frontal Lobe channels, especially FP_1 EEG channel and some Parental Lobe Channels show best (in most of the Performance Metric Evaluation) and the Occipital Lobe channels, especially O_2 channels mentioning the least

Note: Remaining of the Classification Metrics Evaluation Result, please see Appendix B

Table 7.3 MM/I and P300 Oddball Random Datasets - Performance Metrics Table Summary

Performance Metric	Maximum		Minimum	
	EEG Channel	*Value (%)*	*EEG Channel*	*Value (%)*
Accuracy	FC5	87.48	O2	70.297
Error Rate	O2	29.7	FP1	21.7718
Precision / PPV	FP1	76.011	O2	73.19
Recall / Sensitivity / TPR	FP1	76.011	O2	73.19
Specificity / TNR	FP1	92.59	O2	54.103
Balanced Accuracy	FP1	84.301	O2	63.64
F-score	FP1	85.5397	O2	80.123
Gmean	FP1	82.35	O2	59.84
Cohen Kappa (Khat)	FP1	38.38	O2	18.62
MCC	FP1	47.616	O2	21.92
FMI	FP1	76.011	O2	73.19
BMI	FP1	68.698	O2	27.294
LPR	PO7	150.184	FP1	109.63
LNR	O2	34.0686	FP1	2.88
NPV	FP1	92.597	O2	54.103
FPR	AF8	195.58	PO8	148.61
FNR	O2	8.2	FP1	0.599
FDR	O2	26.809	FP1	23.988
FOR	O2	45.896	FP1	7.402
Prevalence	FP1	89.725	PO8	84.792
Prevalence Threshold	AF3	59.82	PO8	56.426
CSI	FP1	75.459	O2	67.319
MK	FP1	68.608	O2	27.294
ROC-AUC	FP1	68.608	O2	27.294

Best
Worst

7.2.3 Receiver Operating Characteristic - Area Under The Curve (ROC-AUC)

Receiver operating characteristic (ROC) curve for hypothetical data is shown in Figure 7.17 showing the **Receiver Operating Characteristic - Area under the Curve (ROC-AUC)** - performance of a classification model at all classification thresholds - Selectivity (Sc) with sensitivity and false-positivity (1 — Specificity). A plot of these values yielded this ROC curve. The values represent the cut-off value(Sc) that each point on the curve corresponds to.

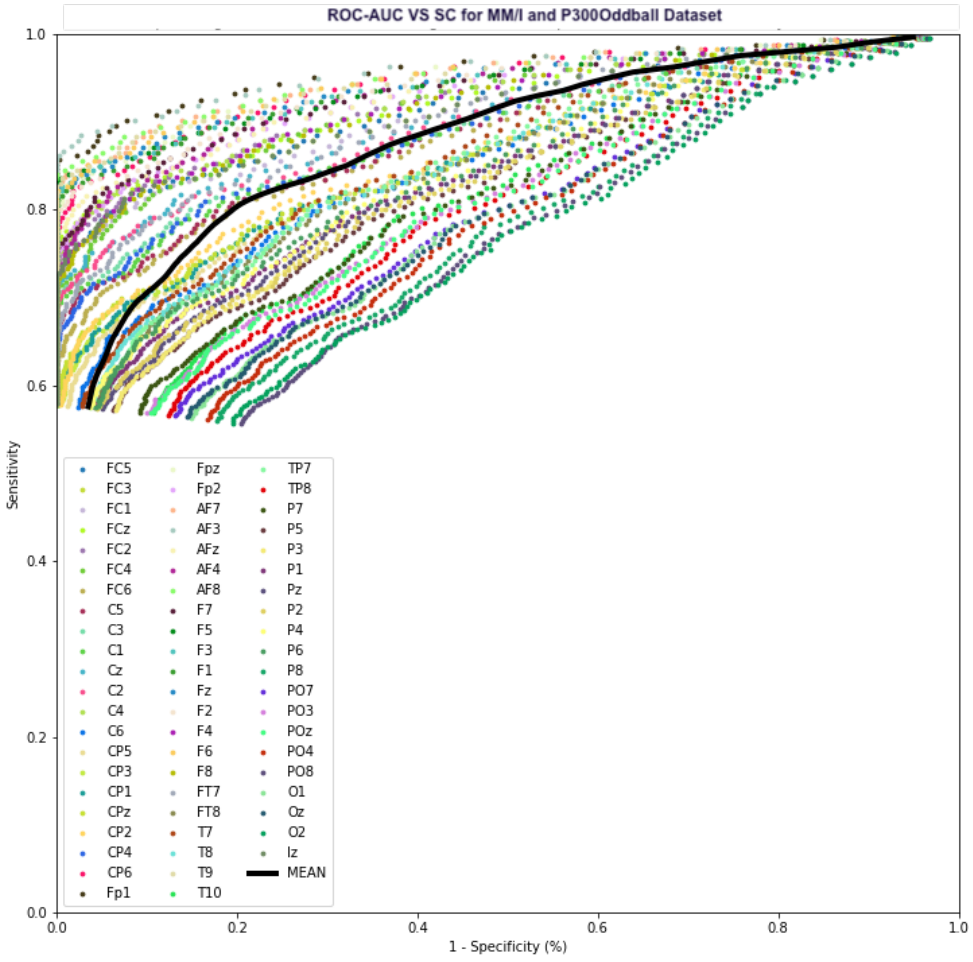


Fig. 7.17 Reciever Operation Characteristic - Area Under the Curve (ROC-AUC) Rate (Sc = PC_1)

As observed, all the channels move from sensitive to more specific as Selectivity (Sc) increases but never crosses down the diagonal discriminant cut-off, and the channels all passed the ROC-AUC requirements above the diagonal line, as shown in Figure 3.18.

ROC curves also permit a numeric assessment of the overall performance of diagnostic tests. This is done by estimating the area under (i.e., to the right of and beneath) the curve and is expressed as a proportion of the total area of the square in which the curve is drawn. A test with higher sensitivity and specificity would reach closer to the left upper corner and have a higher area under the curve. The comparison of the EEG channels is also shown, which perform well.

7.3 Conclusion

In this work, the impact of the Principal Component Analysis - Sample Reduction Process (PCA-SRP) of MM/I EEG Dataset mixed by **foreign** randomness samples (P300 Oddball Auditory dataset) in the selecting of training sets for the performance of Artificial Neural Networks (ANN) - Model Accuracy using Power Spectrum Distribution with the aid of Simpson's and Welch Estimation has been explored. The work has investigated the impact of a controlled randomness sample(P300 Oddball Paradigm dataset) in the L/R Motor Movement (MM) dataset through Data Visualization, Confusion Matrix, some classification and performance metrics evaluations.

Here are the findings of this chapter:

- As Selectivity (Sc) increases, some positive and negative samples are removed from the dataset.
- Selectivity (Sc) Threshold is described as $94 \leq Sc < 100$ in the Equation 7.1, as shown also in Figure 7.18.
- In selecting Sc is based on the priority, either sensitive (having all the good ones) or selective (removing all the bad ones). The higher the Selectivity (Sc) the curve moves from sensitive to selective.
- The FP_1 EEG channel performs better in most Performance Metric Evaluations, and O_2 shows the least one.
- It is limited only up to 15-20% additional randomness, more than compromising its effectiveness as shown in Figure 7.19.

Overall, there is substantial flexibility in using PCA-SRP on the MM/I dataset for training ANN systems. The work has also been viable from different types of mental activity that may be aggregated to provide more robust system training.

Future work will be focused on the internal interference of EEG signal if it will isolate minority mental activity in the EEG dataset, furthermore, by evaluating the robustness of this approach when collecting data with long time intervals between training and testing as well as data from low-cost EEG sensors.

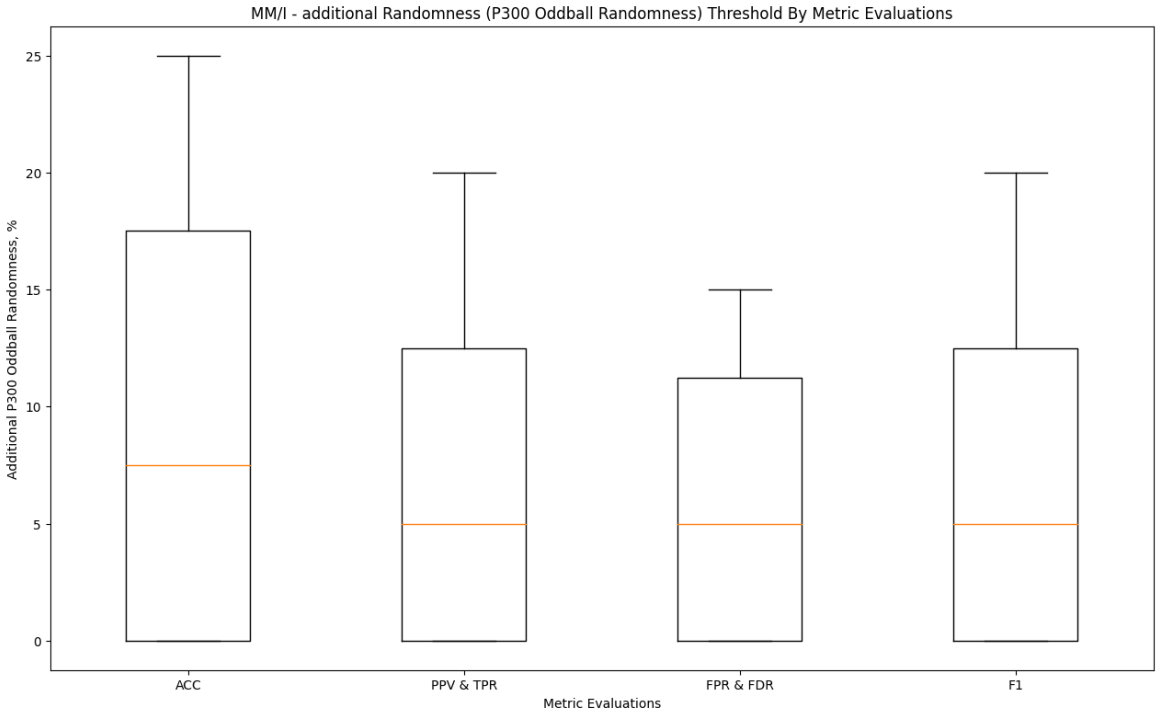
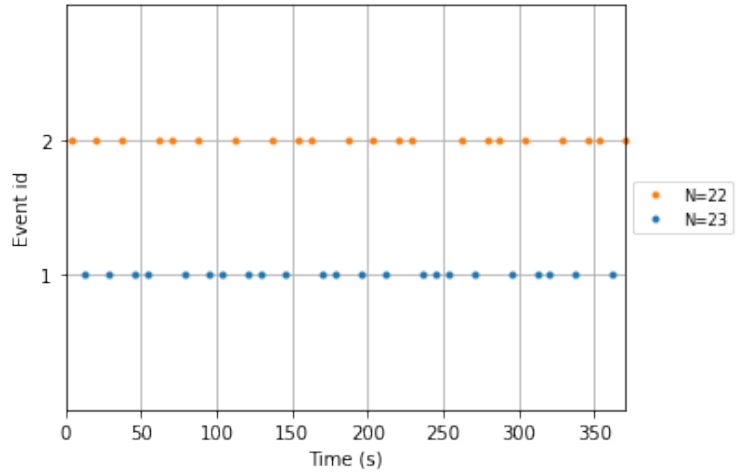
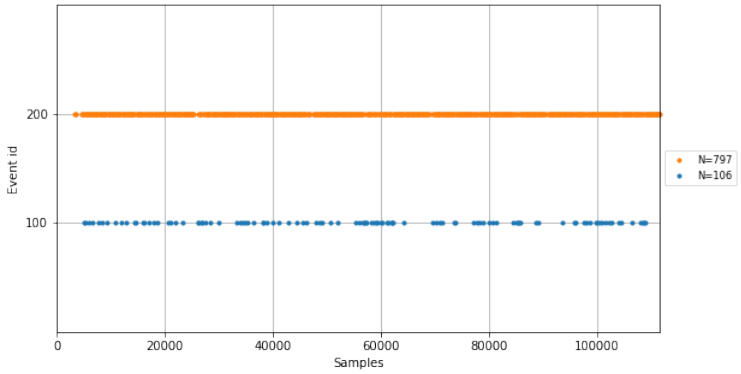


Fig. 7.19 Additional Randomness Threshold by Metric Evaluations



(a) L/R Motor Movement EEG Signal.



(b) P300 Oddball Auditory EEG Signal.

Fig. 7.5 Event ID Diagram : L/R Motor Movement (MM) and Oddball EEG Signals

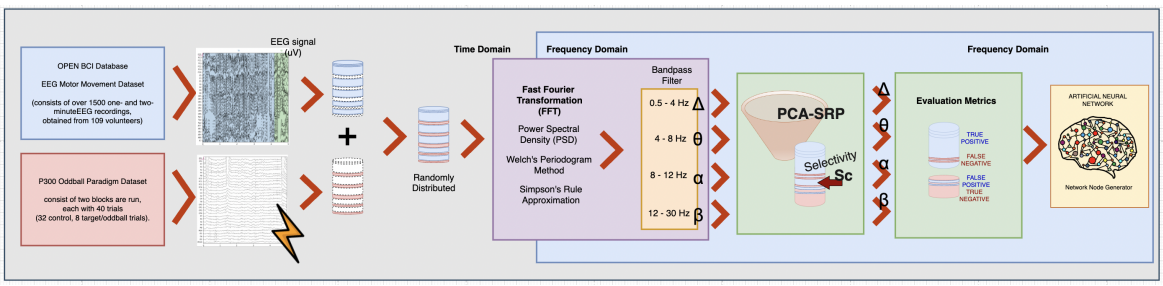
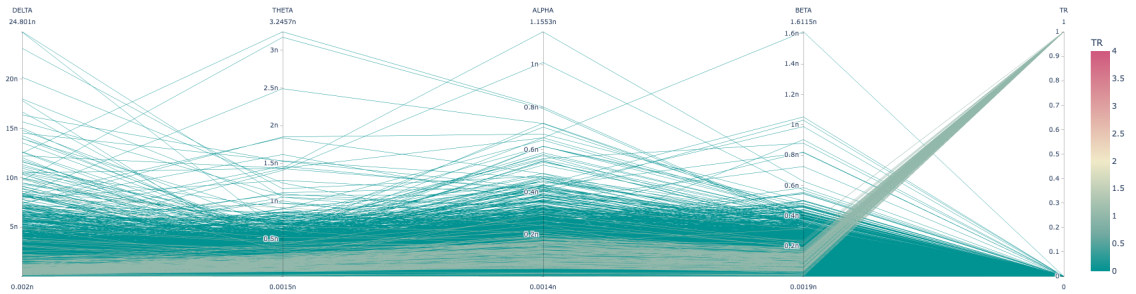
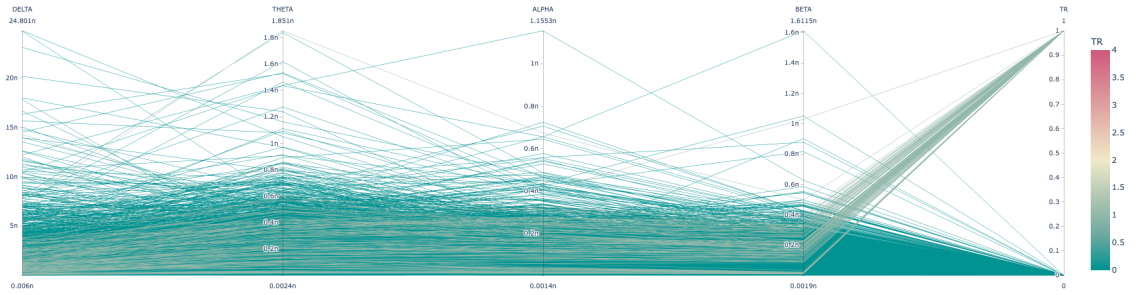


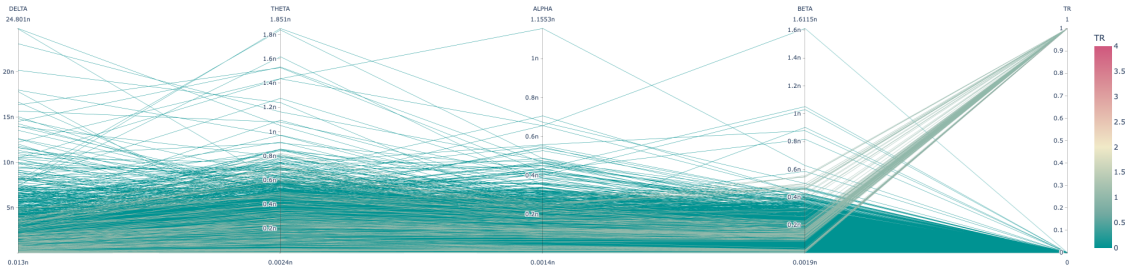
Fig. 7.6 Experimental Methodological Procedure.



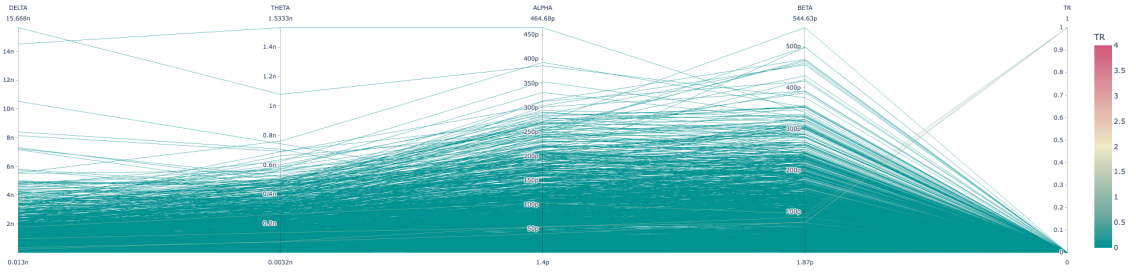
(a) EEG Datasets - Parallel Coordinate Plot (without PCA-SRP) .



(b) EEG Datasets - Parallel Coordinate Plot (Sc = PC_1) .

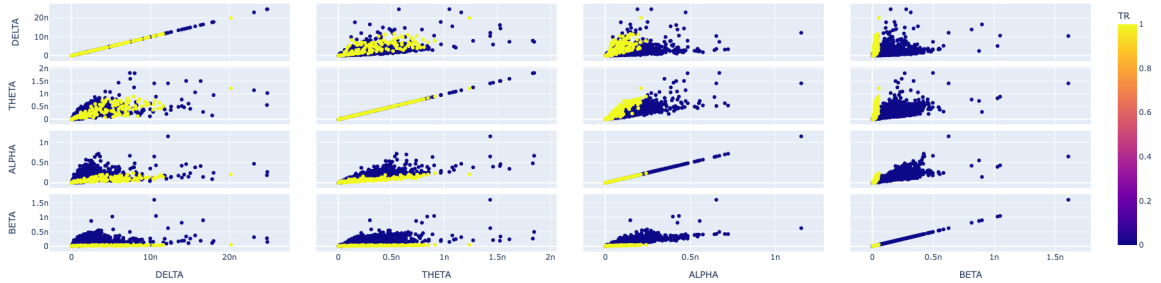


(c) EEG Datasets - Parallel Coordinate Plot (Sc = 99%).

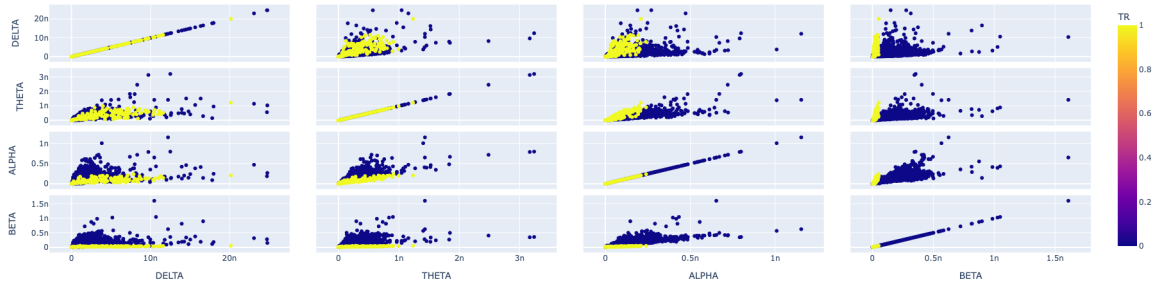


(d) EEG Datasets - Parallel Coordinate Plot (Sc = 99.85%).

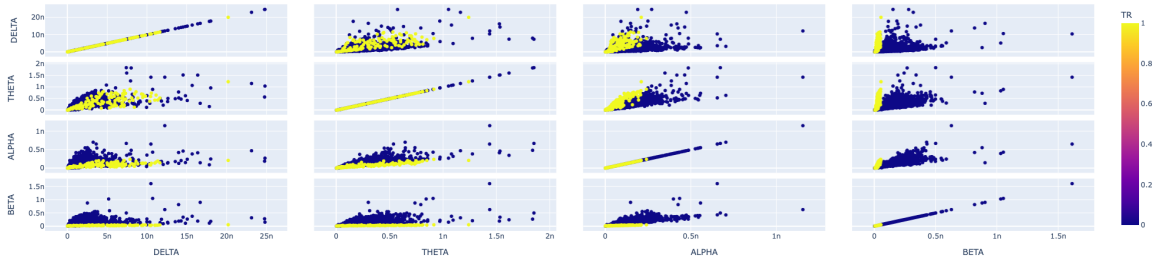
Fig. 7.8 EEG Datasets - Parallel Coordinate Plot with Different Selectivity (*Sc*).



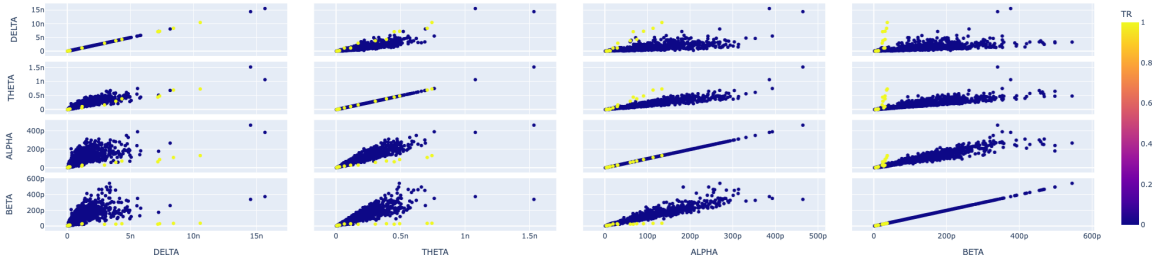
(a) EEG Datasets - Scatter Matrix Plot (without PCA-SRP).



(b) EEG Datasets - Scatter Matrix Plot (Sc = PC_1).



(c) EEG Datasets - Scatter Matrix Plot (Sc = 99%).



(d) EEG Datasets - Scatter Matrix Plot (Sc = 99.85%).

Fig. 7.9 EEG Datasets - Scatter Matrix Plot with Different Selectivity (*Sc*)

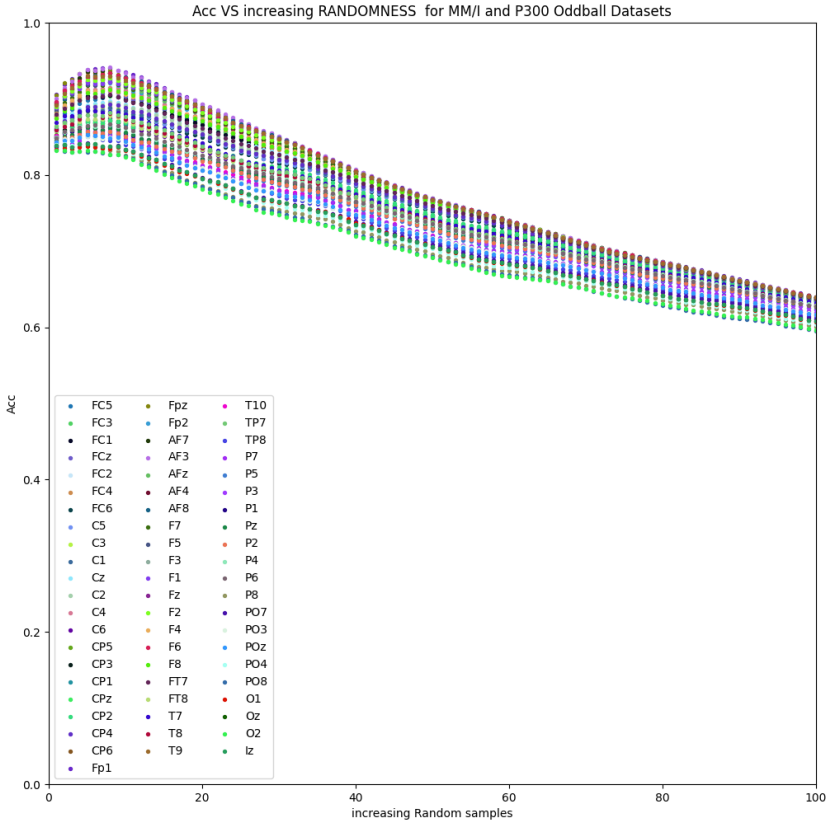


Fig. 7.10 Accuracy Rate vs increasing Randomness (Sc = PC_1)

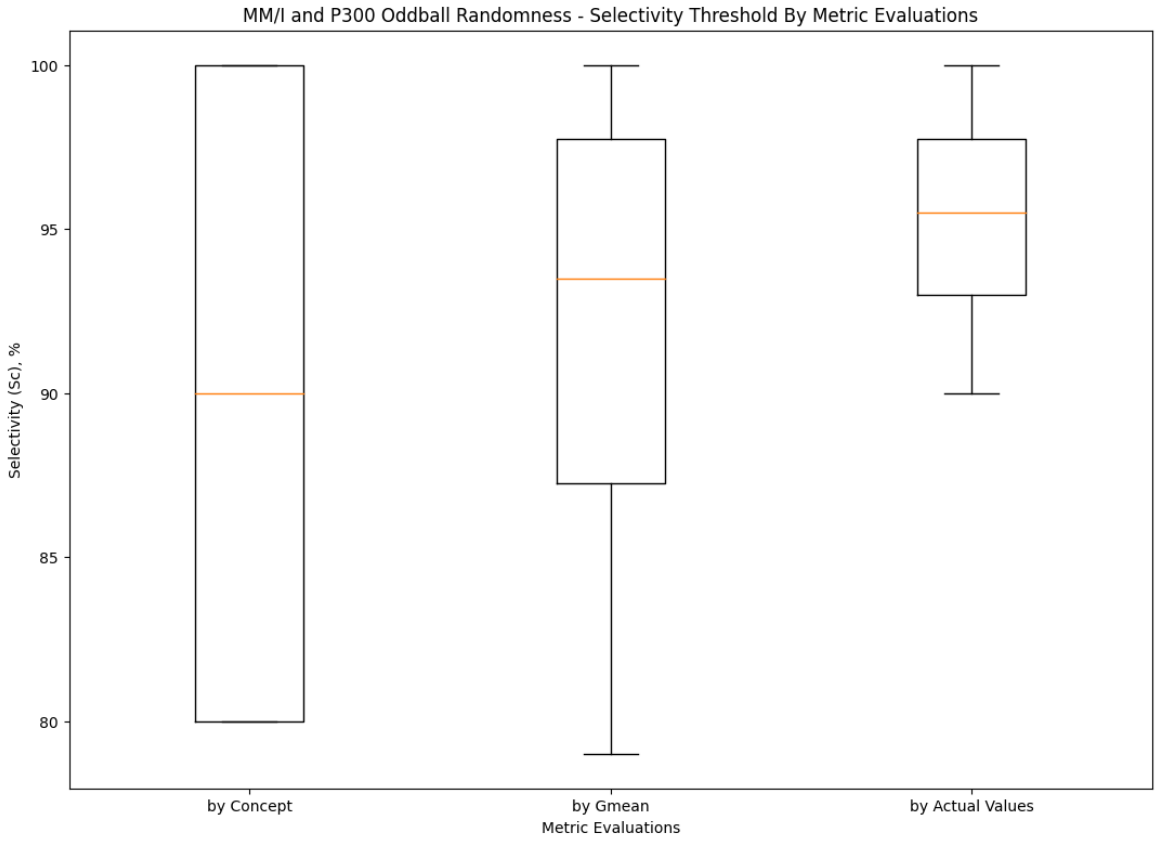


Fig. 7.18 Selectivity Threshold by Metric Evaluations

Chapter 8

Conclusion and Future Work

This chapter provides a summary of the work presented in this dissertation, highlighting the significant contributions to the research and discussing suggestions for future work regarding the use of Principal Component Analysis - Sample Reduction Process (PCA-SRP) as a data cleansing agent.

8.1 Contribution

This section highlights the contributions of this dissertation, which are detailed in the respective chapters. The key contributions are as follows:

- A novel application of PCA-SRP in data cleaning for Big Data.
  - Development of a mathematical model for standard data cleaning methodology.
  - Introduction of a novel feature for cleansing large and highly dimensional physiological datasets.
  - Proposal of a new pre-processing scheme for selecting and sorting training sample sets for basic Artificial Neural Network (ANN) models.
  - Investigation of classification performance metrics for physiological datasets, identifying the limits of the proposed data cleaning process, such as its biased threshold - Selectivity (Sc) - and its effect on injecting increasing randomness into the dataset.
  - Analysis of the effect of the data cleaning process as a training set for ANNs.
- Implementation of PCA-SRP as a data cleaning technique for L/R Motor Movement (MM) EEG datasets.

- Identification of anomalies using different data visualizations of EEG signals.
- Proposal of a new data cleansing scheme for selecting time epochs in EEG data for training basic ANN models.
- Development of a bootstrapping process for basic ANN models for each EEG channel, evaluating the effectiveness of PCA-SRP through ANN model accuracy.
- Presentation of a thorough approach and interpretation of different classification performance metrics in creating a confusion matrix with increasing injected randomness in different EEG channels.
- Identification of the maximum limitation of increasing injected randomness through the L/R Motor Movement (MM) EEG dataset.
- Demonstration that PCA-SRP achieves state-of-the-art results in cleaning training sample sets of the L/R Motor Movement (MM) EEG dataset.

8.2 Discussion and Conclusions

The application of **Principal Component Analysis - Sample Reduction Process (PCA-SRP)** as a data cleansing agent has shown significant improvements in the classification performance of physiological datasets using ANNs. The evaluations used in this dissertation include:

- Accuracy testing, including the observation of parameters with increasing randomness injected into the system.
- ANN bootstrapping using model accuracy parameters of different EEG channels.
- Classification performance metrics (Confusion Metrics, Sensitivity, Specificity, ROC-AUC, etc.).

In accuracy testing, there is a notable increase in the accuracy of simple ANN models by up to 7

This study also examines the impact of cognitive mental activity—Motor Movement and Imagery (Left-Right Hands and Feet Movement)—on the performance of simple ANNs using Fast Fourier Transform (FFT), Power Spectral Densities (PSD) by Simpson-Welch Rule Approximation, and bandpass filtering of EEG frequency bands (δ, θ, α, and β). Using eigenvectors, PCA-SRP effectively removes the least predictive samples.

Given the variability in EEG signals among subjects, some individuals exhibit unique electro-physiological attributes that can cause different interpretations in ANN classification

due to the non-deterministic and non-stationary nature of brainwave signals and sensor biases. Bootstrapping, a statistical procedure that resamples a dataset to create simulated samples, was used to calculate ANN model accuracy. The results indicate increased accuracy in most EEG channels.

The Motor Movement EEG Datasets show non-fully deterministic behavior, with different mental electrical cognition among subjects and non-unified reference base signals for each sensor. Mixing epochs with different characteristics can alter the weight of nodes in the ANN during training.

Visualizing EEG signals in PC space reveals that most random signals are near the origin [0,0,0], indicating poor covariance or loading scores, which are easily removed from the training set. However, some random signals with similar loading scores to cleaned EEG datasets, especially those with the same motor action, pose challenges.

Selecting the Selectivity (*Sc*) parameter is based on priority, either sensitivity (retaining all good samples) or selectivity (removing all bad samples). Higher Selectivity (*Sc*) shifts the focus from sensitivity to selectivity.

The proposed mathematical method, with proper choice of Selectivity (*Sc*), offers flexibility and automatic selection of highly predictive samples, resulting in systems that are easier to develop, deploy, and use in various applications.

The PCA-SRP method, tested by injecting foreign randomness samples (P300 Oddball Auditory dataset) into the training sets for ANN performance, demonstrates significant improvement in classification metrics and ANN model accuracy.

Overall, PCA-SRP provides substantial flexibility for training basic Feed Forward ANN models on the L/R MM EEG dataset, identifying the most and least predictive samples.

8.3 Future Work and Limitations

This dissertation introduces a novel data cleaning technique using PCA through Sample Reduction Process (SRP) for sorting and selecting training sample datasets in ANN models. While the methods demonstrate significant improvement, further investigation is warranted. The limitations of this research study include:

- The datasets are limited to physiological data such as heart disease, voice and speech analysis for gender recognition, breast cancer classification, cancer patient data, and EEG motor and imagery for BCI applications. Future research could explore other applications, such as real-time biomedical automation, image-based medical diagnosis classification, and other scientific fields.

- The acquired EEG signal is limited to the benchmark EEG dataset (L/R Motor Movement and Imagery (MM/I)), due to time constraints. While it is a standard EEG dataset, the sample size may be insufficient to fully justify the study's objectives. Some EEG sensor providers do not allow manipulation and injection of random and noise data into their systems due to cloud database integrity. Future research could involve acquiring a larger EEG data repository.

- The EEG feature extraction is limited to Power Spectrum Density (PSD) in the following bands: Delta (δ), Theta (θ), Alpha (α), and Beta (β). Future research could explore other feature extraction methods such as Wavelet Transformation or Hilbert-Huang Transformation.

- This research focuses on the classification problem of basic Feed Forward Artificial Neural Network Models, as justified in Chapter 3.2. Future studies could apply this data cleaning technique to other machine learning models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Graph Neural Networks (GNN), or Generative Models.

- The study is also viable for different types of mental activities, which could provide more robust system training using low-cost wearable EEG or similar devices to extract brainwave signals, along with other biomedical equipment such as Electrocardiogram (ECG), Electromyogram (EMG), and Electrooculogram (EOG).

8.4 Published Paper

- **Adolfo, C. M. S., Chizari, H., Win, T. Y., and Al-Majeed, S. (2021). Sample Reduction for Physiological Data Analysis Using Principal Component Analysis in an Artificial Neural Network. Applied Sciences, 11(17), 8240.**

References

[1] Michael Thompson. Artificial intelligence in processors to harness data. https://www.synopsys.com/designware-ip/technical-bulletin/using-ai-harness-data-explosion.html, sep 16 2023. [Online; accessed 2023-10-16].

[2] The Economist. The world's most valuable resource is no longer oil, but data, May 2017. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data [Online; accessed 19-July-2022].

[3] MP Barnett. The information explosion. *Nature*, 203:585–585, 1964.

[4] Francois Chung. Neural networks and deep learning. https://francoischung.com/en/my-trainings/neural-networks-and-deep-learning, 2018.

[5] Ben Lutkevich. Ai winter. *Enterprise AI*, Aug 2022.

[6] Amirhosein Toosi, Andrea Bottino, Babak Saboury, Eliot L. Siegel, and Arman Rahmim. A brief history of AI: how to prevent another winter (a critical review). *CoRR*, abs/2109.01517, 2021.

[7] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

[8] A Georghiades, P Belhumeur, and D Kriegman. Yale face database. *Center for computational vision and control at Yale University*, 2(6):33, 1997.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Christos Dimitrakakis and Samy Bengio. Online policy adaptation for ensemble algorithms. Technical report, IDIAP, 2002.

[11] Mark Fanty and Ronald Cole. Spoken letter recognition. *Advances in neural information processing systems*, 3, 1990.

[12] Victor Zue, Stephanie Seneff, and James Glass. Speech database development at mit: Timit and beyond. *Speech communication*, 9(4):351–356, 1990.

[13] Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao, and Limsoon Wong. Deeps: A new instance-based lazy discovery and classification system. *Machine learning*, 54(2):99–124, 2004.

[14] Gordon H Pettengill, Peter G Ford, William TK Johnson, R Keith Raney, and Laurence A Soderblom. Magellan: Radar performance and data products. *Science*, 252(5003):260–265, 1991.

[15] Lester Ingber. Statistical mechanics of neocortical interactions: Canonical momenta indicatorsof electroencephalography. *Physical Review E*, 55(4):4578, 1997.

[16] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[17] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. SPIE, 1993.

[18] Zi-Quan Hong and Jing-Yu Yang. Optimal discriminant plane for a small number of samples and design method of classifier on the plane. *pattern recognition*, 24(4):317–324, 1991.

[19] AM Bagirov, AM Rubinov, NV Soukhoroukova, and J Yearwood. Unsupervised and supervised data classification via nonsmooth and global optimization. *Top*, 11(1):1–75, 2003.

[20] *Proceedings of the Second Australian Conference on Applications of Expert Systems*, USA, 1987. Addison-Wesley Longman Publishing Co., Inc.

[21] David Clark, Zoltan Schreter, and Anthony Adams. A quantitative comparison of dystal and backpropagation. In *Australian conference on neural networks*, pages 132–137, 1996.

[22] Steinbrunn William Pfisterer Matthias Janosi, Andras and Robert Detrano. Heart Disease. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C52P4X.

[23] KORY BECKER. Gender recognition by voice, 2014.

[24] Abien Fred M. Agarap. On breast cancer detection: an application of machine learning algorithms on the wisconsin diagnostic dataset. In *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing - ICMLSC '18*, pages 5–9, Phu Quoc Island, Viet Nam, 2018. ACM Press.

[25] prithivraj. Lung cancer data - dataset by cancerdatahp, March 2016. https://data.world/cancerdatahp/lung-cancer-data [Online; accessed 19-July-2023].

[26] Zoë Corbyn. Ai pioneer fei-fei li: "i'm more concerned about the risks that are here and now". *The Observer*, Nov 2023.

[27] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[28] B.C.M. Fung, K. Wang, and P.S. Yu. Top-down specialization for information and privacy preservation. In *21st International Conference on Data Engineering (ICDE'05)*, pages 205–216, April 2005.

[29] Michael Segner. The annual state of data quality survey, 2024, May 2023.

[30] Gil Press. Cleaning big data: Most time-consuming, least enjoyable data science task, survey says. *Forbes, March*, 23:15, 2016.

[31] Lili Zhang and Xiaoming Wu. Recent progress in challenges of wireless biomedical sensor network. In *2009 3rd International Conference on Bioinformatics and Biomedical Engineering*, pages 1–4, June 2009.

[32] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[33] Craig Stedman. What is data cleansing (data cleaning, data scrubbing)?, Jan 2022.

[34] Lin Li, Taoxin Peng, and Jessie Kennedy. A rule based taxonomy of dirty data. *GSTF Journal on Computing (JoC)*, 1(2), 2014.

[35] Heiko Müller and Johann Christoph Freytag. *Problems, methods, and challenges in comprehensive data cleansing*. Professoren des Inst. Für Informatik, 2005.

[36] Erhard Rahm, Hong Hai Do, et al. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[37] Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. A taxonomy of dirty data. *Data mining and knowledge discovery*, 7:81–99, 2003.

[38] Fu-jun FENG, Jun-ping YAO, and Xiao-jun LI. Research on the technology of data cleaning in big data, 2018.

[39] F. Lian, M. Fu, and X. Ju. An improvement of data cleaning method for grain big data processing using task merging. *Journal of Computer and Communications*, 08:1–19, 2020.

[40] Fakhitah Ridzuan and Wan Mohd Nazmee Wan Zainon. A review on data cleansing methods for big data. *Procedia Computer Science*, 161:731–738, 2019.

[41] F. Zhang, X. Hui-feng, D. Xu, Y. Zhang, and F. You. Big data cleaning algorithms in cloud computing. *International Journal of Online and Biomedical Engineering (iJOE)*, 9:77, 2013.

[42] J. M. Z. H, J. Hossen, S. Sayeed, C. Ho, K. Tawsif, A. Rahman, and E. Arif. A survey on cleaning dirty data using machine learning paradigm for big data analytics. *Indonesian Journal of Electrical Engineering and Computer Science*, 10:1234, 2018.

[43] E. Rendon, R. Alejo, C. Castorena, F. J. Isidro-Ortega, and E. E. Granda-Gutiérrez. Data sampling methods to deal with the big data multi-class imbalance problem. *Applied Sciences*, 10:1276, 2020.

[44] Hongzhi Wang, Mingda Li, Yingyi Bu, Jianzhong Li, Hong Gao, and Jiacheng Zhang. Cleanix: A big data cleaning parfait. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, page 2024–2026, New York, NY, USA, 2014. Association for Computing Machinery.

[45] Mohamed Yakout, Laure Berti-Equille, and Ahmed Elmagarmid. Don't be scared: Use scalable automatic repairing with maximal likelihood and bounded changes. pages 553–564, 06 2013.

[46] Zuhair Khayyat, Ihab Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. Bigdansing: A system for big data cleansing. 06 2015.

[47] D. Martinez-Mosquera and S. Luján-Mora. Data cleaning technique for security big data ecosystem. *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*, 2017.

[48] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1247–1261, New York, NY, USA, 2015. Association for Computing Machinery.

[49] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 2201–2206, New York, NY, USA, 2016. Association for Computing Machinery.

[50] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller. Continuous data cleaning. *2014 IEEE 30th International Conference on Data Engineering*, 2014.

[51] J. K. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, and N. Tang. Interactive and deterministic data cleaning. *Proceedings of the 2016 International Conference on Management of Data*, 2016.

[52] Huan Liu and Hiroshi Motoda. Less is more. In *Computational methods of feature selection*, pages 19–34. Chapman and Hall/CRC, 2007.

[53] Dwight Steward and Roberto Cavazos. *Big data analytics in US courts: uses, challenges, and implications*. Springer Nature, 2019.

[54] Md Abdul Malek. Bigger is always not better; less is more, sometimes: The concept of data minimization in the context of big data. *Eur. J. Privacy L. & Tech.*, page 212, 2021.

[55] Jane E Kirtley and Michael Shally-Jensen. *Privacy Rights in the Digital Age*. Grey House Publishing, 2019.

[56] Kim Kyung Hwan and Kim Sung June. Neural spike sorting under nearly 0-db signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier. *IEEE Transactions on Biomedical Engineering*, 47(10):1406–1411, 2000.

[57] Choh-Man Teng. Correcting noisy data. In *ICML*, volume 99, pages 239–248. Citeseer, 1999.

[58] Jonathan Maletic and Andrian Marcus. Data cleansing: Beyond integrity analysis. pages 200–209, 01 2000.

[59] Ga Young Lee, Lubna Alzamil, Bakhtiyar Doskenov, and Arash Termehchy. A survey on data cleaning methods for improved machine learning model performance. *arXiv preprint arXiv:2109.07127*, 2021.

[60] Ángel Alsina and María Salgado. Understanding early mathematical modelling: First steps in the process of translation between real-world contexts and mathematics. *International Journal of Science and Mathematics Education*, 20(8):1719–1742, 2022.

[61] x engineer.org. What is mathematical modeling. https://x-engineer.org/mathematical-modeling/, 2023. [Online; accessed 2023-10-16].

[62] Dibya Sampad Barik. Unleashing the power of mathematical modeling in machine learning, Aug 2023.

[63] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–7, 2006.

[64] Håkan Brunzell and Jonny Eriksson. Feature reduction for classification of multidimensional data. *Pattern Recognition*, 33(10):1741–1748, 2000.

[65] Tahseen A Jilani, Huda Yasin, and Madiha Mohammad Yasin. Pca-ann for classification of hepatitis-c patients. *International Journal of Computer Applications*, 14(7):1–6, 2011.

[66] Narayan T Deshpande and S Ravishankar. Face detection and recognition using viola-jones algorithm and fusion of pca and ann. *Advances in Computational Sciences and Technology*, 10(5):1173–1189, 2017.

[67] Harun Uğuz. A biomedical system based on artificial neural network and principal component analysis for diagnosis of the heart valve diseases. *Journal of Medical Systems*, 36(1):61–72, 2012.

[68] NP Narendra and Paavo Alku. Dysarthric speech classification from coded telephone speech using glottal features. *Speech Communication*, 110:47–55, 2019.

[69] Letícia Parada Moreira, Landulfo Silveira, Marcos Tadeu Tavares Pacheco, Alexandre Galvão da Silva, and Debora Dias Ferraretto Moura Rocco. Detecting urine metabolites related to training performance in swimming athletes by means of raman spectroscopy and principal component analysis. *Journal of Photochemistry and Photobiology B: Biology*, 185:223–234, 2018.

[70] H Benhar, A Idri, and JL Fernández-Alemán. Data preprocessing for heart disease classification: A systematic literature review. *Computer Methods and Programs in Biomedicine*, page 105635, 2020.

[71] Sadık Kara, Ayşegül Güven, and Semra İçer. Classification of macular and optic nerve disease by principal component analysis. *Computers in Biology and Medicine*, 37(6):836–841, 2007.

[72] A. Krisciukaitis, M. Tamosiunas, P. Jakuska, R. Veteikis, R. Lekas, V. Saferis, and R. Benetis. Evaluation of ischemic injury of the cardiac tissue by using the principal component analysis of an epicardial electrogram. *Comput Methods Programs Biomed*, 82(2):121–9, 2006.

[73] Vinod Kumar, Jainy Sachdeva, Indra Gupta, Niranjan Khandelwal, and Chirag Kamal Ahuja. Classification of brain tumors using pca-ann. In *2011 World Congress on Information and Communication Technologies*, pages 1079–1083, Dec 2011.

[74] Kishor Walse, Rajiv Dharaskar, and V. M. Thakare. *PCA Based Optimal ANN Classifiers for Human Activity Recognition Using Mobile Sensors Data*, volume 50, pages 429–436. 01 2016.

[75] Alberto Landi, Paolo Piaggi, and Giovanni Pioggia. Backpropagation-based non linear pca for biomedical applications. In *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 635–640, Nov 2009.

[76] A Mostaar, MR Sattari, S Hosseini, and MR Deevband. Use of artificial neural networks and pca to predict results of infertility treatment in the icsi method. *Journal of biomedical physics and engineering*, 9(6):679, 2019.

[77] Deepak Mehta and Chaman Verma. *Prediction of Cancer Diagnosis Patients from Fine-Needle Aspirates Using Machine Learning*, pages 337–348. 01 2020.

[78] Bernhard Geiger and Gernot Kubin. Relative information loss in the pca. pages 562–566, 09 2012.

[79] Ganesh R Naik. *Advances in principal component analysis: research and development*. Springer, 2017.

[80] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[81] Avijeet Biswal. rincipal Component Analysis in Machine Learning | Simplilearn, April 2023. https://www.simplilearn.com/tutorials/machine-learning-tutorial/principal-component-analysis [Online; accessed 19-July-2023].

[82] Di-Yuan Tzeng and Roy S Berns. A review of principal component analysis and its applications to color technology. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 30(2):84–98, 2005.

[83] Moisés Silva, Adam Santos, Reginaldo Santos, Eloi Figueiredo, Claudomiro Sales, and João CWA Costa. Deep principal component analysis: An enhanced approach for structural damage identification. *Structural Health Monitoring*, 18(5-6):1444–1463, 2019.

[84] Cid Mathew Santiago Adolfo, Hassan Chizari, Thu Yein Win, and Salah Al-Majeed. Sample reduction for physiological data analysis using principal component analysis in artificial neural network. *Applied Sciences*, 11(17):8240, 2021.

[85] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[86] Afshin Gholamy, Vladik Kreinovich, and Olga Kosheleva. Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation. 2018.

[87] Mateusz. Kwasniak. How to decide on learning rate, Jun 2021. https://towardsdatascience.com/how-to-decide-on-learning-rate-6b6996510c98 [Online; accessed 19-July-2022].

[88] Yingjun Chen and Yijie Hao. Integrating principle component analysis and weighted support vector machine for stock trading signals prediction. *Neurocomputing*, 321:381–402, 2018.

[89] Xiao Zhong and David Enke. Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, 5(1):1–20, 2019.

[90] Juliana Adeola Adisa, Samuel Olusegun Ojo, Pius Adewale Owolawi, and Agnieta Beatrijs Pretorius. Financial distress prediction: Principle component analysis and artificial neural networks. In *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–6, Nov 2019.

[91] Ngoc M. Tran, Petra Burdejová, Maria Ospienko, and Wolfgang K. Härdle. Principal component analysis in an asymmetric norm. *Journal of Multivariate Analysis*, 171:1–21, 2019.

[92] John E. Richards. Recovering dipole sources from scalp-recorded event-related-potentials using component analysis: principal component analysis and independent component analysis. *International Journal of Psychophysiology*, 54(3):201–220, 2004.

[93] Devansh Shah, Samir Patel, and Santosh Kumar Bharti. Heart disease prediction using machine learning techniques. *SN Computer Science*, 1:1–6, 2020.

[94] Lakhan Jasuja, Akhtar Rasool, and Gaurav Hajela. Voice gender recognizer recognition of gender from voice using deep neural networks. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pages 319–324. IEEE, 2020.

[95] Dewi Nasien, Veren Enjeslina, M Hasmil Adiya, and Zirawani Baharum. Breast cancer prediction using artificial neural networks back propagation method. In *Journal of Physics: Conference Series*, volume 2319, page 012025. IOP Publishing, 2022.

[96] Ibrahim M Nasser and Samy S Abu-Naser. Lung cancer detection using artificial neural network. *International Journal of Engineering and Information Systems (IJEAIS)*, 3(3):17–23, 2019.

[97] Benyamin Ghojogh and Mark Crowley. Principal sample analysis for data reduction. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 350–357. IEEE, 2018.

[98] Akash Dubey. The Mathematics Behind Principal Component Analysis, December 2018.

[99] Ian T Jolliffe. Rotation of principal components: some comments. *Journal of Climatology*, 7(5):507–510, 1987.

[100] Zhiyuan Chen, Samson Oni, Susan Hoban, and Onimi Jademi. A comparative study of data cleaning tools. *Int. J. Data Warehous. Min.*, 15(4):48–65, 2019.

[101] Edgar P. Torres, Edgar A. Torres, Myriam Hernández-Álvarez, and Sang Guun Yoo. Eeg-based bci emotion recognition: A survey. *Sensors*, 20(18), 2020.

[102] Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[103] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[104] Luciana Nieto and Adrian Correndo. Classification performance metrics and indices, March 2023. https://adriancorrendo.github.io/metrica/articles/available_metrics_classification.html [Online; accessed 19-July-2023].

[105] U. Michelucci. Code for *estimating neural network's performance with bootstrap: a tutorial.* https://github.com/toelt-llc/NN-Performance-Bootstrap-Tutorial, 2021.

[106] Se-Il Suk, Jin-Hyok Kim, Won-Joong Kim, Sang-Min Lee, Ewy-Ryong Chung, and Ki-Ho Nah. Posterior Vertebral Column Resection For Severe Spinal Deformities:. *Spine*, 27(21):2374–2382, November 2002.

[107] Shruti Vyas, Subhabrata Das, and Yen-Peng Ting. Predictive modeling and response analysis of spent catalyst bioleaching using artificial neural network. *Bioresource Technology Reports*, 9:100389, 2020.

[108] Christofer N Yalung and Cid Mathew S Adolfo. Analysis of obstacle detection using ultrasonic sensor. *International Research Journal and Engineering Technology*, pages 1015–1019, 2017.

[109] Praveen Tirupattur, Yogesh Singh Rawat, Concetto Spampinato, and Mubarak Shah. Thoughtviz: Visualizing human thoughts using generative adversarial network. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 950–958, New York, NY, USA, 2018. Association for Computing Machinery.

[110] Christofer N. Yalung, Salah S. Al-Majeed, Cid Mathew Adolfo, Jalal Karam, and Lela Mirtskhulava. Gyroscope explorer terrain angles classification. *2016 IEEE East-West Design & Test Symposium (EWDTS)*, pages 1–5, 2016.

[111] Christofer N Yalung and Cid Mathew S Adolfo. Analysis of obstacle detection using ultrasonic sensor. *Int. Res. J. Eng. Technol*, 4(1):1015–1019, 2017.

[112] Julie Ann B Susa, Erwin Mariquina, Meriam L Tria, Cid Mathew Adolfo, and Julius C Castro. Cap-eye-citor: A machine vision inference approach of capacitor detection for pcb automatic optical inspection. In *2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–5. IEEE, 2020.

[113] Christofer Yalung, Salah Al Majeed, Cid Mathew Adolfo, Jalal Karam, and Lela Mirtskhulava. Gyroscope explorer terrain angles classification. In *2016 IEEE East-West Design & Test Symposium (EWDTS)*, pages 1–5. IEEE, 2016.

[114] E Niedermyer. Electroen-cephalography. *Basic Principles Clinical Applications, and Related Fields*, 1999.

[115] Barbara E Swartz. The advantages of digital over analog recording techniques. *Electroencephalography and clinical neurophysiology*, 106(2):113–117, 1998.

[116] Serap Aydın, Hamdi Melih Saraoğlu, and Sadık Kara. Log energy entropy-based eeg classification with multilayer neural networks in seizure. *Annals of biomedical engineering*, 37:2626–2630, 2009.

[117] Christofer N Yalung, Salah Al Majeed, and Jalal Karam. Analysis and interpretation of brain wave signals. In *Proceedings of the International Conference on Internet of things and Cloud Computing*, pages 1–8, 2016.

[118] Wlodzimierz Klonowski. Everything you wanted to ask about eeg but were afraid to get the right answer. *Nonlinear biomedical physics*, 3(1):1–5, 2009.

[119] José Manuel Rodríguez Delgado. *Physical control of the mind: Toward a psychocivilized society*, volume 41. World Bank Publications, 1969.

[120] Jacques J Vidal. Toward direct brain-computer communication. *Annual review of Biophysics and Bioengineering*, 2(1):157–180, 1973.

[121] Rajesh PN Rao. *Brain-computer interfacing: an introduction*. Cambridge University Press, 2013.

[122] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M Vaughan. Brain–computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791, 2002.

[123] Mamunur Rashid, Norizam Sulaiman, Anwar PP Abdul Majeed, Rabiu Muazu Musa, Ahmad Fakhri Ab Nasir, Bifta Sama Bari, and Sabira Khatun. Current status, challenges, and possible solutions of eeg-based brain-computer interface: a comprehensive review. *Frontiers in neurorobotics*, page 25, 2020.

[124] Laurens R. Krol. EEG electrode positions in the 10-10 system using modified combinatorial nomenclature, Nov 2020. https://commons.wikimedia.org/wiki/File:EEG_10-10_system_with_additional_information.svg [Online; accessed 19-July-2022].

[125] Otis M Solomon Jr. Psd computations using welch's method. *NASA STI/Recon Technical Report N*, 92:23584, 1991.

[126] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[127] Arnaud Delorme and Scott Makeig. Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *Journal of neuroscience methods*, 134(1):9–21, 2004.

[128] Alois Schlögl and Clemens Brunner. Biosig: a free and open source software library for bci research. *Computer*, 41(10):44–50, 2008.

[129] François Tadel, Sylvain Baillet, John C Mosher, Dimitrios Pantazis, and Richard M Leahy. Brainstorm: a user-friendly application for meg/eeg analysis. *Computational intelligence and neuroscience*, 2011:1–13, 2011.

[130] Robert Oostenveld, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen. Fieldtrip: open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Computational intelligence and neuroscience*, 2011:1–9, 2011.

[131] V Litvak, J Mattout, S Kiebel, C Phillips, R Henson, J Kilner, G Barnes, R Oostenveld, J Daunizeau, G Flandin, et al. Eeg and meg data analysis in spm8 computational intelligence and neuroscience. 2011.

[132] Forrest Sheng Bao, Xin Liu, Christina Zhang, et al. Pyeeg: an open source python module for eeg/meg feature extraction. *Computational intelligence and neuroscience*, 2011, 2011.

[133] Sarang S Dalal, Johanna M Zumer, Adrian G Guggisberg, Michael Trumpis, Daniel DE Wong, Kensuke Sekihara, and Srikantan S Nagarajan. Meg/eeg source reconstruction, statistical evaluation, and visualization with nutmeg. *Computational intelligence and neuroscience*, 2011, 2011.

[134] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, et al. Meg and eeg data analysis with mne-python. *Frontiers in neuroscience*, page 267, 2013.

[135] Lorenz Esch, Christoph Dinh, Eric Larson, Denis Engemann, Mainak Jas, Sheraz Khan, Alexandre Gramfort, and Matti S Hämäläinen. Mne: software for acquiring, processing, and visualizing meg/eeg data. *Magnetoencephalography: From Signals to Dynamic Cortical Networks*, pages 355–371, 2019.

[136] Christian Andreas Kothe and Scott Makeig. Bcilab: a platform for brain–computer interface development. *Journal of neural engineering*, 10(5):056014, 2013.

[137] Jessica Schrouff, Maria J Rosa, Jane Rondina, Andre F Marquand, Carlton Chu, Jonas Richiardi, Christophe Phillips, and Janaina Mourao-Miranda. Pattern recognition for neuroimaging toolbox.

[138] Bastian Venthur and Benjamin Blankertz. Wyrm, a pythonic toolbox for brain-computer interfacing. *arXiv preprint arXiv:1412.6378*, 2014.

[139] A Barachant and JR King. pyriemann v0. 2.2, 2015.

[140] Nikolaas N Oosterhof, Andrew C Connolly, and James V Haxby. Cosmomvpa: multi-modal multivariate pattern analysis of neuroimaging data in matlab/gnu octave. *Frontiers in neuroinformatics*, 10:27, 2016.

[141] April R Levin, Adriana S Méndez Leal, Laurel J Gabard-Durnam, and Heather M O'Leary. Beapp: the batch electroencephalography automated processing platform. *Frontiers in neuroscience*, 12:513, 2018.

[142] Vinay Jayaram and Alexandre Barachant. Moabb: trustworthy algorithm benchmarking for bcis. *Journal of neural engineering*, 15(6):066011, 2018.

[143] Zied Tayeb, Nicolai Waniek, Juri Fedjaev, Nejla Ghaboosi, Leonard Rychly, Christian Widderich, Christoph Richter, Jonas Braun, Matteo Saveriano, Gordon Cheng, et al. Gumpy: A python toolbox suitable for hybrid brain–computer interfaces. *Journal of neural engineering*, 15(6):065003, 2018.

[144] Etienne Combrisson, Raphael Vallat, Christian O'Reilly, Mainak Jas, Annalisa Pascarella, Anne-lise Saive, Thomas Thiery, David Meunier, Dmitrii Altukhov, Tarek Lajnef, et al. Visbrain: a multi-purpose gpu-accelerated open-source suite for multimodal brain data visualization. *Frontiers in Neuroinformatics*, 13:14, 2019.

[145] Zitong Lu and Yixuan Ku. Neurora: A python toolbox of representational analysis from multi-modal neural data. *Frontiers in neuroinformatics*, 14:563669, 2020.

[146] Etienne Combrisson, Timothy Nest, Andrea Brovelli, Robin AA Ince, Juan LP Soto, Aymeric Guillot, and Karim Jerbi. Tensorpac: An open-source python toolbox for tensor-based phase-amplitude coupling measurement in electrophysiological brain signals. *PLoS computational biology*, 16(10):e1008302, 2020.

[147] Thomas P Urbach and Andrey S Portnoy. fitgrid: A python package for multi-channel event-related time series regression modeling. *Journal of open source software*, 6(63):3293, 2021.

[148] Avraam D Marimpis, Stavros I Dimitriadis, and Rainer Goebel. Dyconnmap: Dynamic connectome mapping—a neuroimaging python module. *Human Brain Mapping*, 42(15):4909–4939, 2021.

[149] Aurélien Appriou, Léa Pillette, David Trocellier, Dan Dutartre, Andrzej Cichocki, and Fabien Lotte. Biopyc, an open-source python toolbox for offline electroencephalographic and physiological signals classification. *Sensors*, 21(17):5740, 2021.

[150] Luis Cabanero-Gomez, Ramon Hervas, Ivan Gonzalez, and Luis Rodriguez-Benitez. eeglib: a python module for eeg feature extraction. *SoftwareX*, 15:100745, 2021.

[151] Simon Legeay, Gustavo Caetano, Patricia Figueiredo, and Athanasios Vourvopoulos. Neuxus: A biosignal processing and classification pipeline for real-time brain-computer interaction. In *2022 IEEE 21st Mediterranean Electrotechnical Conference (MELECON)*, pages 424–429. IEEE, 2022.

[152] David López-García, Jose MG Peñalver, Juan M Górriz, and María Ruz. Mvpalab: A machine learning decoding toolbox for multidimensional electroencephalography data. *Computer Methods and Programs in Biomedicine*, 214:106549, 2022.

[153] Clemens Brunner, Giuseppe Andreoni, Lugi Bianchi, Benjamin Blankertz, Christian Breitwieser, Shin'ichiro Kanoh, Christian A Kothe, Anatole Lécuyer, Scott Makeig, Jürgen Mellinger, et al. Bci software platforms. In *Towards Practical Brain-Computer Interfaces*, pages 303–331. Springer, 2012.

[154] Mamunur Rashid, Norizam Sulaiman, Anwar PP Abdul Majeed, Rabiu Muazu Musa, Bifta Sama Bari, and Sabira Khatun. Current status, challenges, and possible solutions of eeg-based brain-computer interface: a comprehensive review. *Frontiers in neurorobotics*, page 25, 2020.

[155] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[156] Yousef Rezaei Tabar and Ugur Halici. A novel deep learning approach for classification of eeg motor imagery signals. *Journal of neural engineering*, 14(1):016003, 2016.

[157] Carlos Daniel Virgilio Gonzalez, Juan Humberto Sossa Azuela, Elsa Rubio Espino, and Victor H Ponce Ponce. Classification of motor imagery eeg signals with csp filtering through neural networks models. In *Mexican International Conference on Artificial Intelligence*, pages 123–135. Springer, 2018.

[158] Navneet Tibrewal, Nikki Leeuwis, and Maryam Alimardani. Classification of motor imagery eeg using deep learning increases performance in inefficient bci users. *Plos one*, 17(7):e0268880, 2022.

[159] Amjed S Al-Fahoum and Ausilah A Al-Fraihat. Methods of eeg signal features extraction using linear analysis in frequency and time-frequency domains. *International Scholarly Research Notices*, 2014, 2014.

[160] Seyed Navid Resalat and Valiallah Saba. A study of various feature extraction methods on a motor imagery based brain computer interface system. *Basic and clinical neuroscience*, 7(1):13, 2016.

[161] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013.

[162] Gregg Cage and Kylie Smith. Experiment: The consciousness detector - eeg, oddball task, and p300, 2017.

[163] Samuel Sutton, Margery Braren, Joseph Zubin, and ER John. Evoked-potential correlates of stimulus uncertainty. *Science*, 150(3700):1187–1188, 1965.

[164] R Johnson. The amplitude of the p300 component of the event-related potential: Review and synthesis. *Advances in psychophysiology*, 3(April):69–137, 1988.

[165] Understanding Principle Component Analysis(PCA) step by step. https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9.

Appendix A

Python Implementation

Principal Component Analysis - Sample Reduction Process (PCA_{SRP})

NOTE: This is Python 3 code and was tested with matplotlib v. 2.1.0

Import packages that we will be working with.

```
import os
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn import preprocessing
import matplotlib.pyplot as plt
import tensorflow as tf
optim = tf.keras.optimizers.Adam()
```

PCA-SRP Generation Code Dataset I/O Directory

```
filename = "11 heart"
file_loc = "/content/gdrive/MyDrive/Colab Notebooks/
Neural Network Program Python/"
file_loc_input =  open( file_loc + filename + ".csv","r")
file_loc_input_new = file_loc + filename + " new.csv"
load_score_loc = file_loc + filename + " sorted loading
scores.csv"
```

Dataset Classification Column Name and Data extraction in CSV UTF-8 file

```
target = 'target'
data = pd.read_csv( file_loc_input)
print( data )
print( data.describe(include='all')) # Print data descriptions
maxsample = data[target].count() # get the number of samples
```

Perform PCA on the dataset

```
scaled_data = preprocessing.scale(data.T) # First center and transpose the data
pca = PCA() # create a PCA object and do the math
pca.fit(scaled_data) # scale and fit the data
pca_data = pca.transform( scaled_data) # get PCA
coordinates for scaled_data
```

The following code constructs the Scree plot

```
print( pca.explained_variance_)
per_var = np.round( pca.explained_variance_ratio_* 100,
decimals = 5)
labels = ['PC' + str(x) for x in range( 1,  len( per_var) + 1)]
plt.bar( x = range( 1,  len(per_var) + 1),  height = per_var,
tick_label = labels)
plt.ylabel( 'Percentage of Explained Variance' )
plt.xlabel( 'Principal Component' )
plt.title( 'Scree Plot' )
plt.show()
```

The following code makes a fancy looking plot using PC1 and PC2

```
pca_df = pd.DataFrame( pca_data, columns = labels)
plt.scatter( pca_df.PC1, pca_df.PC2)
plt.title( 'My PCA Graph' )
plt.xlabel( 'PC1 - {0}%'.format(per_var[0]) )
plt.ylabel( 'PC2 - {0}%'.format(per_var[1]) )
```

Plot the PC1 and PC2

```
for sample in pca_df.index:
    plt.annotate( sample,  ( pca_df.PC1.loc[sample],  pca_df.PC2.loc[sample] ) )
plt.show()
```

Generating Parameters

```
print( 'list of PC variance: \n ',  per_var )
```

Generating loading score of every sample in PC_1 Sort the loading scores based on their magnitude Print the top samples and their scores (and +/- sign)

```
loading_scores = pd.Series(pca.components_[0])
sorted_loading_scores = loading_scores.abs().sort_values( ascending=False )
```

Save in .csv and record the sorted loading scores of every samples

```
temp = sorted_loading_scores
temp.to_csv( load_score_loc,  index=True )
print( temp)
```

Determining the Sc - min, test, max

```
Sc_min = 2 * per_var[0] - 100  # from equation 25
```

Sc ranges

```
print(  Sc_min, "  > Sc  > ",  Sc_max, ",  for recommended normal cleaning")
print(  Sc_max, " > Sc  > 1.00,  for recommended ANN
app cleaning")
```

Input Selectivity (*% Sc*)

```
Sc = Sc_max / 100
```

Cleaning Process

```
sorted_loading_scores.drop( sorted_loading_scores[sorted_loading_scores <
Sc * sorted_loading_scores.max()].index, inplace = True )
passrate = ( len( sorted_loading_scores ) / maxsample * 100)
maxtopsamples = int( maxsample * passrate / 100 )
print("\n",  len( sorted_loading_scores ), " / ",  maxsample )
```

Removing the unwanted samples

```
new_sorted_loading_scores = sorted_loading_scores
[ 0 : maxtopsamples ]
print( new_sorted_loading_scores )
```

Get the indexes of the cleaned samples

```
top_samples = new_sorted_loading_scores.index.
sort_values( ascending = True )
```

Getting the data of the top samples from original

```
data_new = []
for i in range( maxtopsamples ) :
    n = top_samples[i]
    data_new.append( ( data.iloc[n] ) )
```

Append and save in csv file

```
data_new = pd.DataFrame( data_new )
```

Save the new file

```
data_new.to_csv( file_loc_input_new,  index = False )
print( '\n new sorted loading dataset generated as',
file_loc_input_new )
```

PCA-SRP Simplified

```
def PCA_SRP(data, file_loc, f_name):

  maxsample = len(data.index)
  scaled_data = preprocessing.scale(data.T)
  # First, centre and transpose the data
  pca = PCA() # Create a PCA object and do the math
  pca.fit(scaled_data) #scale and fit the data
  pca_data = pca.transform(scaled_data)
  # get PCA coordinates for scaled_data

  loading_scores = pd.Series(pca.components_[0]).abs()
  #generating the loading score of every sample in PC1
  per_var = np.round(pca.explained_variance_ratio_* 100, decimals=5)
  # generated pca_variance
  Sc = (per_var[0] + per_var[1])/100  # selectivity (Sc) parameter FORMULA

  result = pd.concat([data, loading_scores], axis=1)
  # concatinate the data and its respective loading scores
  sorted_loading_scores = loading_scores.abs().sort_values(ascending=False)
  # Sorting the PCA result in descending order based on its loading scores.
  threshold = sorted_loading_scores.iloc[int(Sc * maxsample)]
  # Setting the threshold based on the Selectivity (Sc)

  min_drop = 0 #value of the drop samples
  result = result.where(result.iloc
   [ :, len(data.columns)] >= threshold, other= min_drop)
   # dropping all samples below the threshold
  del result[0]
```

```
  #deleting the last column (loading score)

  passrate = (len(result[result.iloc
   [ :, len(data.columns)-1 ] != min_drop])) / maxsample*100
  # compute the passing rate
  #print(passrate, " %")
  # printing the passing rate

  result.to_csv(file_loc + f_name + " new.csv", index=False)
  # save the generated PCA-SRP result to CSV
  #print(result)
  # printing the result dataset
  #print("PCA-SRP done \n")
  return pass rate
```

A.1 Cross-Validation Split

Setting the test size for cross-validation.

```
size_test = 0.2
```

Here is the command to separate them into [X, Y]. Where our target labels are 'Y', and 'X' is our training data.

```
Y = data_old.target.values
X = data_old.drop([class_name],  axis = 1)
print( '\n - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  \n')
```

Now split to train/test, where cross-validation split took place.

```
X_train,  X_test,  Y_train,  Y_test = train_test_split
( X,  Y,  test_size = size_test,  random_state = 42 )
print( '\n - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  \n')
```

Artificial Neural Network (*ANN*) Testing

NOTE: This was tested with matplotlib v. 2.1.0 and Google Colab

Import packages that we will be working with.

```
import os
import math
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn import preprocessing
import matplotlib.pyplot as plt
import tensorflow as tf
optim = tf.keras.optimizers.Adam()
from pandas.plotting import scatter_matrix
from keras.layers import Dense
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
#from tensorflow.python.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
accuracy_score
from sklearn.preprocessing import MinMaxScaler
from scipy.interpolate import UnivariateSpline
import timeit
```

Setting the time

```
start = timeit.default_timer()
```

Setting ANN hyperparameters

```
target = 'target'
class_name = target
learning_rate = 0.1
epoch = 100
size_test = 0.2
```

Creating a dataframe

```
data = pd.DataFrame()
```

Setting Data and File Directory

```
file_loc = "/content/gdrive/MyDrive/Colab Notebooks/
Neural Network Program Python/"
filename = "16 cancer patient"
other = " random"
```

Getting the CSV file

```
file_loc_input =  ( file_loc + filename +  ".csv")
data = pd.read_csv( file_loc_input)
data_old = data
```

ANN Part

Input of number of input column and output classification

```
input_column = len( data_old.columns) - 1
output_class = len( data_old.groupby( class_name).size() )
```

At this moment we have a dataframe that contains all of the dataframe .csv data. However we need to separate them to [X, Y].

Where our target labels are 'Y', and 'X' is our training data.

```
Y = data_old.target.values
X = data_old.drop([class_name],  axis = 1)
```

Now split to train/test

```
X_train,  X_test,  Y_train,  Y_test = train_test_split
( X,  Y,  test_size = size_test,  random_state = 42 )
```

Define a Neural Network Model

```
def NN_model( learning_rate ):
    model = Sequential()
    model.add(Dense( 32,  input_dim = input_column,
     kernel_initializer = 'normal', activation = 'relu'))
    model.add( Dense( 16,  kernel_initializer = 'normal',
     activation = 'relu'))
    model.add( Dense( output_class,  activation = 'softmax'))
    #Adam( lr = learning_rate)
    model.compile( loss = 'sparse_categorical_crossentropy',
     optimizer = 'Adam', metrics = ['accuracy'] )
    return model
```

Build a NN-model, and start training

```
model = NN_model( learning_rate )
print(model.summary() )
history = model.fit( X_train,  Y_train,  validation_data =
( X_test,  Y_test),  epochs = epoch,  batch_size = 16,
verbose = 0)
```

Calculate the accuracy and classification report

```
predictions = np.argmax( model.predict( X_test ),  axis = 1 )
model_accuracy = accuracy_score( Y_test,  predictions )
* 100
print( "Model Accuracy:  ",  model_accuracy )
```

Appending the accuracy

```
ModAcc = ModAcc.append( { 'Model Accuracy' :
model_accuracy },  ignore_index = True )
```

T, F, P, TP, FP, FN, TN, and N Tally

```
T = int(n_events)
F = int(add_events)

new_sorted_loading_scores=sorted_loading_scores
new_sorted_loading_scores.drop(new_sorted_loading_scores
[new_sorted_loading_scores > threshold].index, inplace=True)
FN_TN = new_sorted_loading_scores.count()
new_sorted_loading_scores.drop(new_sorted_loading_scores
[new_sorted_loading_scores.index > T -1].index, inplace=True)

FN = new_sorted_loading_scores.count()
TN = FN_TN - FN
TP = T - FN
FP = F - TN
P = TP + FP
```

```
  N = FN + TN
  data = [(T, F, P, TP, FP, FN, TN, N )]
```

Confusion Matrix Generation

```
def confusion_matrix_parameters(  df, file_loc, fname, nchannel):

  parameter = pd.DataFrame()
  for n in range(100):
    T, F, P, TP, FP, FN, TN, N = df.iloc[n]
    data = T, F, P, TP, FP, FN, TN, N
    data = pd.DataFrame(data).transpose()

    conf_param = confusion_matrix_param(T, F, P, TP, FP, FN, TN, N)
    conf_param = pd.DataFrame(conf_param).transpose()

    temp = pd.concat([data , conf_param], axis=1)

    parameter = pd.concat([parameter, temp], axis=0,ignore_index=True)

  parameter.columns = ["T", "R", "P" , "TP", "FP", "FN", "TN", "N",
  "Acc", "Error_rate", "PPV", "TPR", "TNR", "BA", "F1", "Gmean",
  "khat", "MCC", "FM", "BM", "LRP", "LRN", "DOR", "NPV", "FPR",
  "FNR", "FDR", "FOR", "Prev", "PrevT", "TS", "MK", "AUC_ROC"]

  #print(parameter)
  parameter.to_csv(file_loc + fname + str(nchannel +1) + "
  Confusion_Matrix_Summary.csv", index=False)
  print( "  saved")

  return
```

Classification Performance Metric Computation

```
def confusion_matrix_param(T, F, P, TP, FP, FN, TN, N):
```

```
if FN == 0:
  FN = 0.00000001
else:
  FN=FN

Acc = (TP+TN)/(P+N) # accuracy
Error_rate = (FP + FN) / (P + N) #Error Rate
PPV = TP /(TP+FP)  # precision / positive predictive value
TPR = TP/P   # sensitivity / recall / hit rate / true positive rate (TPR)
TNR = TN/N   # specificity /  true negative rate
BA = (TPR+TNR)/2   # balanced accuracy
F1 = 2*TP/(2*TP+FP+FN) # harmonic mean of precision and sensitivity
Gmean = math.sqrt(TPR * TNR) #G-mean
khat = (2* (TP *TN - FN * FP))/ ((TP + FP)*(FP+TN) + (TP +FN)*(FN+TN))
# K-hat or Cohen's Kappa Coefficient
MCC = ((TP*TN)-(FP*FN))/math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
 # phi coefficient / matthews correlation coefficient
FM = math.sqrt(PPV*TPR) # Fowlkes-mallows index
BM = TPR+TNR -1    # informedness / bookmaker informedness
FPR = FP/N   # fall-out / false positive rate
FNR = FN/P  # miss rate / false negative rate
LRP = TPR/FPR      # positive likelihood ratio
LRN = FNR/TNR      # negative likelihood ratio
DOR = LRP/LRN      # Diagnostic odd ratio
NPV = TN/(TN+FN)   # negative predictive value
FDR = FP/(FP+TP)   # false detection / discovery rate
FOR = FN/(FN+TN)   # false omission rate
Prev = P/(P+N)     # Prevelence
PrevT = (math.sqrt(TPR*FPR) - FPR)/(TPR-FPR) # prevelence threshold
TS = TP/(TP+FN+FP) # Threat Score / critical sucess index / jaccard index
MK = PPV+NPV - 1   # markedness / deltaP
AUC_ROC = PPV + NPV -1 # Area under the Curve / Receiver
Operating Characteristic Curve

params = Acc, Error_rate, PPV, TPR, TNR, BA, F1, Gmean, khat, MCC, FM, BM,
```

```
  LRP, LRN, DOR, NPV, FPR, FNR, FDR, FOR, Prev, PrevT, TS, MK, AUC_ROC
  return params
```

Random Signal Generator

```
def coh_signal_gen():
    rand = np.random.RandomState(43)
    """Generate an oscillating signal.

    Returns
    -------
    signal : ndarray
        The generated signal.
    """
    t_rand = 0.001  # Variation in the instantaneous frequency of the signal
    std = 0.1  # Std-dev of the random fluctuations added to the signal
    base_freq = 10.  # Base frequency of the oscillators in Hertz
    #n_times = len(times)
    n_times = len(times) + t_noise * sf
    # Generate an oscillator with varying frequency and phase lag.
    signal =  np.sin(2.0 * np.pi * ((stop_time + t_noise) * sf +
     np.cumsum(t_rand * rand.randn(n_times))))

    # Add some random fluctuations to the signal.
    signal += std * rand.randn(n_times)

    # Scale the signal to be in the right order of magnitude (~100 nAm)
    signal *= 100e-10
    #print("noise generation done \n")
    return signal
```

Epoch generator

```
def psd_signal_gen(EEG_raw, EEG_channel, sf, time_limit,
file_loc, fname):
```

```
#looping to get the wavelet every second (s)
for n in range(0, time_limit):
     stop_time = n + 1
     # setting the stop time
     start_time = n
     # let n as start time
     Tstart, Tstop = EEG_raw.time_as_index([start_time, stop_time])
     # identifying the start and stop time in the EEG RAW data
     data_raw, times = EEG_raw[EEG_channel-1:EEG_channel,
     Tstart : Tstop]
     # extracting from EEG RAW to EEG dataset
  data_raw = data_raw.transpose()
  # transpose the EEG dataset
     samples = int(sf*(stop_time - start_time) )
     # compute the number of samples in each second (s)
     data_eegband  = data_raw.reshape((samples,))
#reshaping the EEG dataset
```

PSD Signal Computation

```
def psd_signal_gen(EEG_raw, EEG_channel, sf, file_loc, fname, z):


  PSD_EEG = pd.DataFrame() # creating a empty dataframe for PSD_EEG


  band_delta= 0.5,4   # setting the delta band
  band_theta = 4,8    # setting the theta band
  band_alpha =  8,12  # setting the alpha band
  band_beta = 12,30   # setting the beta band


#looping to get the PSD every second (s)
  for n in range(0, len(events)+ add_events):
```

```
    if n == 0:
      start_time = 0
    else:
      start_time = events_test[n -1,0]/sf     # let n as start time
    stop_time = events_test[n,0]/sf  # setting the stop time

    Tstart, Tstop = EEG_raw.time_as_index([start_time, stop_time])
    # identifying the start and stop time int he EEG RAW data
    data_raw, times = EEG_raw[EEG_channel-1:EEG_channel, Tstart : Tstop]
    # extracting from EEG RAW to EEG dataset

    data_raw = data_raw.transpose()
    # transpose the EEG dataset
    data_eegband  = data_raw.reshape((len(data_raw),))

    delta = bandpower(data_eegband, sf, band_delta, window_sec=None,
    relative=False)  # getting the PSD delta band
    theta = bandpower(data_eegband, sf, band_theta, window_sec=None,
    relative=False)  # getting the PSD theta band
    alpha = bandpower(data_eegband, sf, band_alpha, window_sec=None,
    relative=False)  # getting the PSD alpha band
    beta = bandpower(data_eegband, sf, band_beta, window_sec=None,
    relative=False)    # getting the PSD beta band

    PSD_EEG = PSD_EEG.append([[delta, theta, alpha, beta]], ignore_index=True)
    # append the bands (delta, theta, alpha, beta)

PSD_EEG.columns = ['DELTA', 'THETA', 'ALPHA', 'BETA']
# labelling the PSD dataset
PSD_EEG.to_csv(file_loc + fname+ "channel_" + str(z+1)+ ".csv", index=False)
# convert PSD dataset to CSV

return PSD_EEG
```

Fourier Transformation Visualization

```
def specgram2d(y, srate=44100, ax=None, title=None):
  if not ax:
    ax = plt.axes()
  ax.set_title("Amplitude vs Frequency")
  spec, freqs, t, im = ax.specgram(y, Fs=fs, scale='dB', vmax=0)
  ax.set_xlabel('time (s)')
  ax.set_ylabel('frequencies (Hz)')
  cbar = plt.colorbar(im, ax=ax)
  cbar.set_label('Amplitude (dB)')
  cbar.minorticks_on()
  return spec, freqs, t, im
```

A.2 Simpson's Estimation Method and Welch PSD Calculation

```
def bandpower(data, sf, band, window_sec=None, relative=False):
    "Compute the average power of the signal x in a
    specific frequency band.

    Parameters
    ----------
    data : 1d-array
        The input signal in the time domain.
    SF : float
        The sampling frequency of the data.
    band : list
        Lower and upper frequencies of the band of interest.
    window_sec : float
        Length of each window in seconds.
        If None, window_sec = (1 / min(band)) * 2
    relative : boolean
        If True, return the relative power (=
        divided by the total power of the signal).
        If False (default), return the absolute power.
```

```
    Return
    ------
    bp : float
        Absolute or relative band power.
    """
    from scipy.signal import welch
    from scipy.integrate import simps
    band = np.asarray(band)
    low, high = band

    # Define window length
    if window_sec is not None:
        nperseg = window_sec * sf
    else:
        nperseg = (2 / low) * sf

    # Compute the modified periodogram (Welch)
    freqs, psd = welch(data, sf, nperseg=nperseg)

    # Frequency resolution
    freq_res = freqs[1] - freqs[0]

    # Find the closest indices of the band in the frequency vector
    idx_band = np.logical_and(freqs >= low, freqs <= high)

    # Integral approximation of the spectrum using Simpson's rule.
    bp = simps(psd[idx_band], dx=freq_res)

    if relative:
        bp /= simps(PSD, dx=freq_res)
    return bp
```

Performance Metrics Graph Generation

```
def graph_visualization_PM1(file_location, data_frame, a, x, y):
```

```
# construct some data like what you have:

ch_name = ['FC5', 'FC3', 'FC1', 'FCz', 'FC2', 'FC4', 'FC6', 'C5', 'C3',
'C1', 'Cz', 'C2', 'C4', 'C6', 'CP5', 'CP3', 'CP1', 'CPz', 'CP2', 'CP4',
'CP6', 'Fp1', 'Fpz', 'Fp2', 'AF7', 'AF3', 'AFz', 'AF4', 'AF8', 'F7', 'F5',
'F3', 'F1', 'Fz', 'F2', 'F4', 'F6', 'F8', 'FT7', 'FT8', 'T7', 'T8', 'T9',
'T10', 'TP7', 'TP8', 'P7', 'P5', 'P3', 'P1', 'Pz', 'P2', 'P4', 'P6', 'P8',
'PO7',  'PO3', 'POz', 'PO4', 'PO8', 'O1', 'Oz', 'O2', 'Iz']

folder_file = file_location + str(task[x]) + "_data/Diagram/
Performance Metrics Test 1/ " + (str(a+1).zfill(2)) + " " +
 str(P_metric[a]) + "_" + str(mental_action[x])


############################

data = pd.DataFrame()
df = pd.DataFrame(x for x in range(1, 101))

data = pd.concat([df, data_frame.iloc[:,0] ], axis = 1)
data.columns = ["x", "y"]
ax1 = data.plot( x="x",  y = "y", kind='scatter', figsize = (12,12),
s = 8, ylabel = (str(P_metric[a])), xlabel = ("increasing Random samples"),
xlim = (0,100), ylim = (0,1),  title = (str(P_metric[a]) +
" VS increasing RANDOMNESS  for MM/I and P300 Oddball Datasets"))


for nchannel in range (1,64):
  data = pd.DataFrame()
  data = pd.concat([df , data_frame.iloc[:, nchannel] ], axis = 1)
  data.columns = ["x", "y"]
  data.plot(x="x",  y = "y", kind = "scatter", ax = ax1, s = 8,
  ylabel = (str(P_metric[a])), xlabel = ("increasing Random samples"),
  c = ('#%02X%02X%02X' % (random.randint(0,255),
  random.randint(0,255),random.randint(0,255))))
```

```
  ############################
  plt.legend(ch_name, ncol = 3)

  # create stacked errorbars:

  plt.savefig( folder_file + ".png")
  plt.show()
  print("file saved")
  return
```

ROC - AUC Graph Generation

```
def graph_visualization_PM5(file_location, data_frame1, data_frame2,   x,  y):

  mental_action = [ "motor action", "motor imagery"]
  noise_random = ["noise", "random"]
  ch_name = ['FC5',  'FC3',  'FC1',  'FCz',  'FC2',  'FC4',  'FC6',  'C5',
  'C3',  'C1',  'Cz',  'C2',  'C4',  'C6',  'CP5',  'CP3',  'CP1',  'CPz',
  'CP2', 'CP4', 'CP6', 'Fp1', 'Fpz', 'Fp2', 'AF7', 'AF3', 'AFz',
  'AF4', 'AF8', 'F7', 'F5', 'F3', 'F1', 'Fz', 'F2', 'F4', 'F6', 'F8', 'FT7',
  'FT8', 'T7', 'T8', 'T9', 'T10', 'TP7', 'TP8', 'P7', 'P5', 'P3', 'P1', 'Pz',
  'P2', 'P4', 'P6', 'P8', 'PO7', 'PO3', 'POz', 'PO4', 'PO8', 'O1', 'Oz',
  'O2', 'Iz', 'MEAN']

  folder_file = file_location  + str(task[x]) + "_data/Diagram/
  Performance Metrics Test 5/ROC_" + str(mental_action[y])

  data = pd.DataFrame()

  data = pd.concat([data_frame1.iloc[:,0],data_frame2.iloc[:,0] ], axis = 1)
  data.columns = ["x", "y"]
  ax1 = data.plot( x = "x",  y = "y", kind='scatter', figsize = (12,12),
  s = 8, xlabel = ("1 - Specificity (%)"), ylabel = ("Sensitivity"),
```

```
xlim = (0,1), ylim = (0,1),  title = ("Receiver Operating Curve (ROC)
of increasing RANDOM samples for " + str(task[x])  + "-" +
 str(mental_action[y]) + " PhysioNET EEG Dataset"))


for nchannel in range (1,64):
  data = pd.DataFrame()
  data = pd.concat([data_frame1.iloc[:,nchannel],data_frame2.
  iloc[:,nchannel] ],
   axis = 1)
  data.columns = ["x", "y"]
  data.plot(x = "x",  y = "y", kind = "scatter", ax = ax1, s = 8,
  xlabel = ("1 - Specificity (%)"), ylabel = ("Sensitivity"),
  c = ('#%02X%02X%02X' % (random.randint(0,255),random.randint(0,255),
  random.randint(0,255))))


data = pd.DataFrame()
data = pd.concat([data_frame1.iloc[:,64],data_frame2.iloc[:,64] ], axis = 1)
data.columns = ["x", "y"]
data.plot(x = "x",  y = "y", kind = "line", ax = ax1,
xlabel = ("1 - Specificity (%)"), ylabel = ("Sensitivity"),
c = 'black', lw = 4)

plt.legend(ch_name, ncol = 3)

plt.savefig( folder_file + "_79%.png")
plt.show()
print("file saved")

return
```

PCA – 3D Scatter Plot Visualisation

```
import plotly.express as px
from sklearn.decomposition import PCA

X = dirtyset[['DELTA', 'THETA', 'ALPHA', 'BETA']]

pca = PCA(n_components=3)
components = pca.fit_transform(X)

total_var = pca.explained_variance_ratio_.sum() * 100

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=dirtyset['TR'],
    title=f'Total Explained Variance: {total_var:.2f}%',
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}, width=1200, height=1000)

fig.show()
```

Parallel Coordinate Plot Visualisation

```
import plotly.express as px

fig = px.parallel_coordinates(dirtyset, color="TR",
labels={"TR",'DELTA', 'THETA', 'ALPHA', 'BETA', },
color_continuous_scale=px.colors.diverging.Tealrose,
color_continuous_midpoint=2)
fig.show()
```

Scatter Matrix Plot Visualisation

```
import plotly.express as px
fig = px.scatter_matrix(dirtyset,
dimensions=['DELTA', 'THETA', 'ALPHA', 'BETA'], color="TR")
fig.show()
```

ANN Model Accuracy Comparison Visualisation

```
df1 = pd.read_csv( file_location +  "MMI_PSD channel_ACC1.csv")
df2 = pd.read_csv( file_location +  "MMI_PSD channel_ACC2.csv" )

sumdf1 = df1.sum(axis=0)
sumdf2 = df2.sum(axis=0)

ch_name = ['FC5', 'FC3', 'FC1', 'FCz', 'FC2', 'FC4', 'FC6', 'C5', 'C3',
'C1', 'Cz', 'C2', 'C4', 'C6', 'CP5', 'CP3', 'CP1', 'CPz', 'CP2', 'CP4', 'CP6',
'Fp1', 'Fpz', 'Fp2', 'AF7', 'AF3', 'AFz', 'AF4', 'AF8', 'F7', 'F5', 'F3', 'F1',
'Fz', 'F2', 'F4', 'F6', 'F8', 'FT7', 'FT8', 'T7', 'T8', 'T9', 'T10', 'TP7',
'TP8', 'P7', 'P5', 'P3', 'P1', 'Pz', 'P2', 'P4', 'P6', 'P8', 'PO7', 'PO3',
'POz', 'PO4', 'PO8', 'O1', 'Oz', 'O2', 'Iz']

dframe = pd.DataFrame({'ANN only': sumdf1,
                    'PCA+ANN': sumdf2 }, index=ch_name)
ax = dframe.plot.bar(rot=0, figsize = (25, 10), ylim = (40, 55),
ylabel = ("Validation Accuracy, %"), xlabel = ("EEG Channels"),
title = ("MM/I EEG Dataset - ANN Validation Accuracy of PCA + ANN
and ANN only"))

ax = inc_accuracy.plot.bar(figsize = (25, 10), x='index',
y=0, rot=0, ylabel = ("increase Validation Accuracy, %"),
xlabel = ("EEG Channels"),
title = ("MM/I EEG Dataset - increase ANN Validation Accuracy
(PCA + ANN vs ANN only)"), color = ['orange'])
```

EEG Channel SC Visualisation

```
ax = summary.plot.bar(figsize = (25, 10), y = 'n_samples removed, %',
x = 'EEG channel', ylim = (5, 20),  ylabel = ("n samples removed, %"),
xlabel = ("EEG Channels"),
title = ("MM/I EEG Dataset - Samples Removed After PCA-SRP
(n = 4927)"), color = 'maroon')
```

```
ax = summary.plot.bar(figsize = (25, 10), y = 'Sc', x = 'EEG channel',
ylim = (0.8, 1.0),  rot=0, ylabel = ("Selectivity (Sc), %"),
xlabel = ("EEG Channels"), title = ("Selectivity per EEG Channel"))
```

MNE Library Import Requirements

```
!pip install mne
import mne.channels
from mne import EpochsArray
from mne.channels import make_standard_montage
from mne.epochs import concatenate_epochs
from mne import create_info, find_events, Epochs
#from mne.viz.topomap import _prepare_topo_plot, plot_topomap
from mne.decoding import CSP
from mne import find_events, Epochs, compute_covariance, make_ad_hoc_cov
from mne.datasets import sample
from mne.simulation import (simulate_sparse_stc, simulate_raw,
add_noise, add_ecg, add_eog)
from numpy.ma.core import shape
```

Motor Movement and Imagery (MM/I) Data Extraction

```
#runs = [5, 9, 13]  # motor action: hands vs feet
#runs = [6, 10, 14]  # motor imagery: hands vs feet
runs = [3, 7, 11]    # motor action: left vs right hand
#runs = [4, 8, 12]    # motor imagery: left vs right hand

n_subjects = 109
subject = 40

raw_fnames = eegbci.load_data(subject, runs)
raw = concatenate_raws([read_raw_edf(f, preload=True) for f in raw_fnames])

eegbci.standardize(raw)  # set channel names
montage = make_standard_montage('standard_1005')
```

```
raw.set_montage(montage)
```

Time Signal Plot Visualisation

```
raw.describe()
raw.plot(duration=60.0, start=0.0, n_channels=64);
```

PSD Plot Visualisation

```
raw.compute_psd(fmax=50).plot()
raw.plot(duration=5, n_channels=64)
print(raw.info)
```

Event - Epoch ID Visualisation

```
events, _ = events_from_annotations(raw, event_id=dict(T1=0, T2=1))
event_ids = {"left": 0, "right": 1}
epochs = mne.Epochs(raw, events, event_id=event_ids)
mne.viz.plot_events(events[:]);print(events)
```

Epoch Time Signal Intensity Visualisation

```
epochs.plot();
epochs['left']
epochs['right']
epochs.info
epochs['left'].plot_image(picks=[29]);
epochs['left'].plot_image(picks=[37]);
epochs_for_tfr = mne.Epochs(raw, events,
 tmin=-.5, tmax=1.5, preload=True)  # need longer data segment
epochs_for_tfr.plot( scalings = 'auto', n_channels=64);
epochs.get_data().shape
left = epochs['left'].average()
right = epochs['right'].average()
epochs.get_data().shape
```

```
epochs['left'].get_data().shape
```

Carnial Topography and Butterfly Plot Visualization

```
left.plot();
left.plot_topomap(times = [0.1, 0.2, 0.3, 0.4, 0.5]);
left.plot_joint(times = [0.1, 0.2, 0.3, 0.4, 0.5]);

right.plot();
right.plot_topomap(times = [0.1, 0.2, 0.3, 0.4, 0.5]);
right.plot_joint(times = [0.1, 0.2, 0.3, 0.4, 0.5]);

diff = mne.combine_evoked((left, -right), weights='equal')
diff.plot_joint(times=[0.1, 0.2, 0.3, 0.37, 0.5]);
diff.plot_image();
```

Region of Interest Visualization

```
rois = mne.channels.make_1020_channel_selections(diff.info, midline="z12")
diff.plot_image(group_by=rois, show=False, show_names="all");
mne.viz.plot_compare_evokeds({"right": right,
                              "left": left}, picks=[29]);
mne.viz.plot_compare_evokeds({"right": right,
                              "left": left}, picks=[37]);
left.plot_topo(color = 'orange');
right.plot_topo();
```

Cranial Localization Sensor Visualization

```
right.plot_sensors(show_names = True);
```

Bootstrapping -Training the Artificial Neural Network

```
start = time.time()
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=30)
```

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dense(16, activation='relu'),
  tf.keras.layers.Dense(1, activation='softmax')])

model.compile(optimizer='adam',  loss='mse',  metrics=['mse'])
model.fit(x_train,y_train, epochs=300, batch_size = 16, verbose = 0)
model.evaluate(x_test,y_test)[1]

end = time.time()
print((end - start)/60.0, "min elapsed.")
```

Bootstrapping of the dataset (either by accuracy or Mean Square Error (MSE))

```
start = time.time()
N = 2000
bootstrap_mse = []
for i in range(N): # 100 Bootstrap samples
    if (i % 200 == 0):
        print("Iteration", i)
    idx = np.random.choice(np.arange(len(x_test)), len(x_test), replace=True)

    x_test = pd.DataFrame(x_test).to_numpy()
    y_test = pd.DataFrame(y_test).to_numpy()

    x_sample = x_test[idx]
    y_sample = y_test[idx]

    bootstrap_mse.append(model.evaluate(x_sample, y_sample, verbose = 0)[1])
end = time.time()
print((end - start)/60.0, "min elapsed.")
```

Bootstrapping - Histograph visualisation comparison of PCA-SRP with ANN and ANN only

```
plt.figure(figsize=(10,6))
plt.xlabel('Accuracy')
plt.ylabel('Frequency')
plt.hist(data1, bins = 80, histtype = 'step', lw = 2,  color = 'blue',
density = True,  label='ANN only')
plt.hist(data2, bins = 80, histtype = 'step', lw = 2,  color = 'orange',
density = True, label='PCA+SRP in ANN')
plt.title("Bootstrapping - Accuracy: Channel "+ str(EEG_channel +1) +
": "+ str(EEG_ch_names[EEG_channel]))

plt.legend()
plt.savefig( "/content/gdrive/MyDrive/Colab Notebooks/MMI/Diagram/
Bootstrapping - Accuracy: Channel "+  str(EEG_channel +1)+ " "
+ str(EEG_ch_names[EEG_channel]) + ".png")
plt.show()
```

Appendix B

Classification Performance Metrics Graph

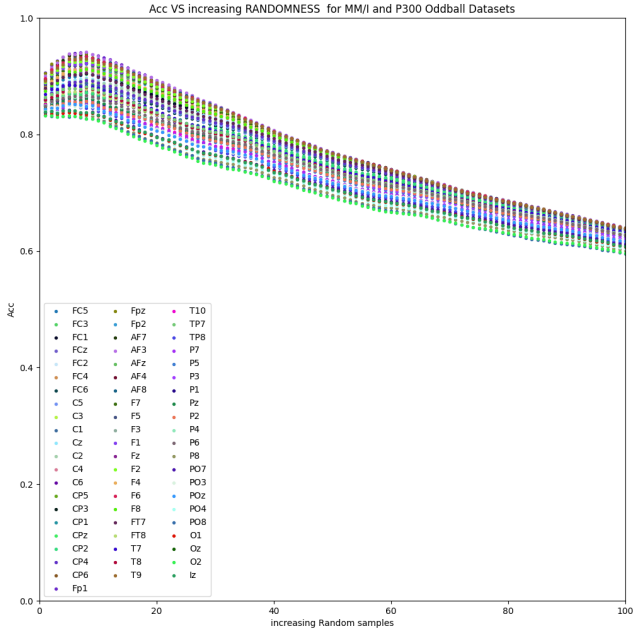


Fig. B.1 Accuracy vs increasing Randomness (Sc = PC_1)

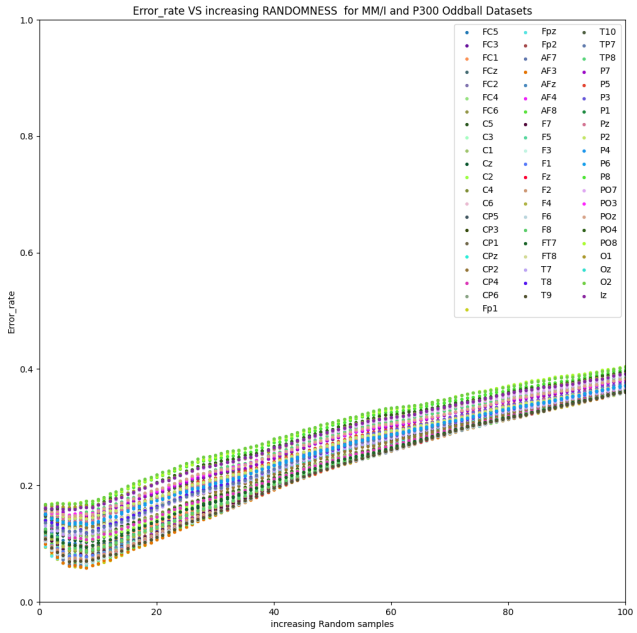


Fig. B.2 Error Rate vs increasing Randomness (Sc = PC_1)

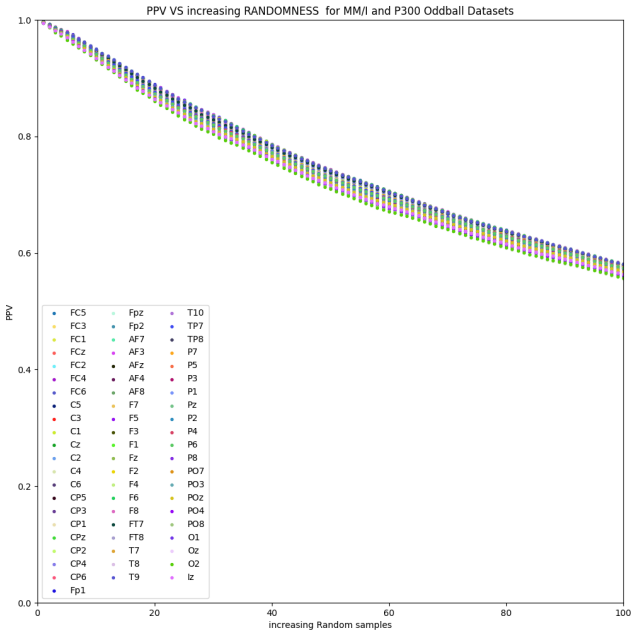


Fig. B.3 PPV vs increasing Randomness (Sc = PC_1)

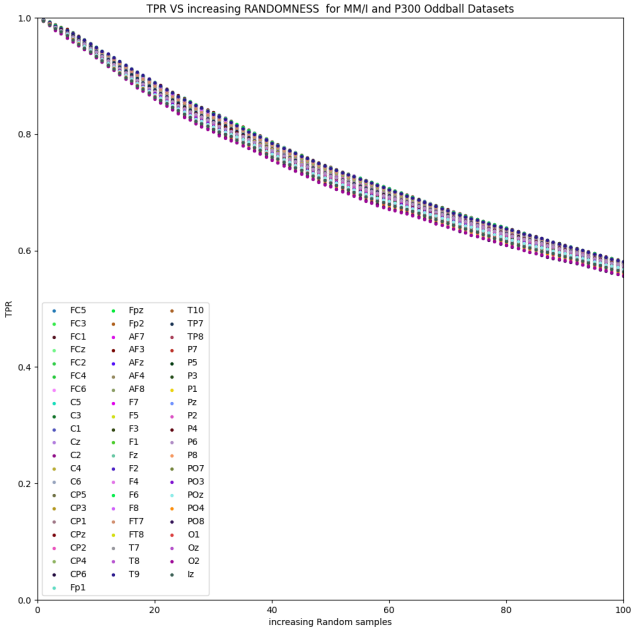


Fig. B.4 TPR vs increasing Randomness (Sc = PC_1)

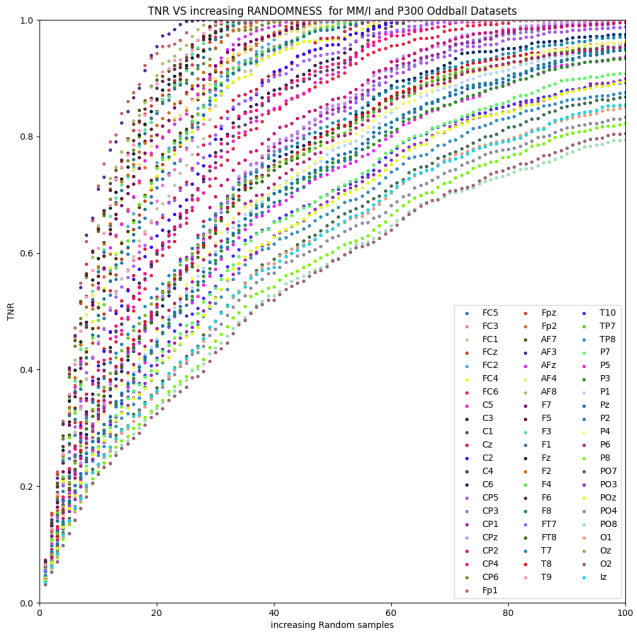


Fig. B.5 True Negative Rate (TNR) vs increasing randomness

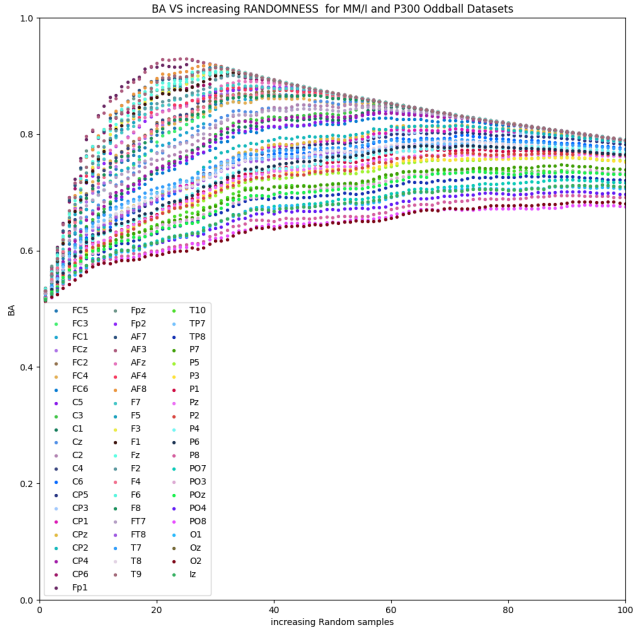


Fig. B.6 Balanced Accuracy (BA) Rate vs increasing Randomness (Sc = PC_1)

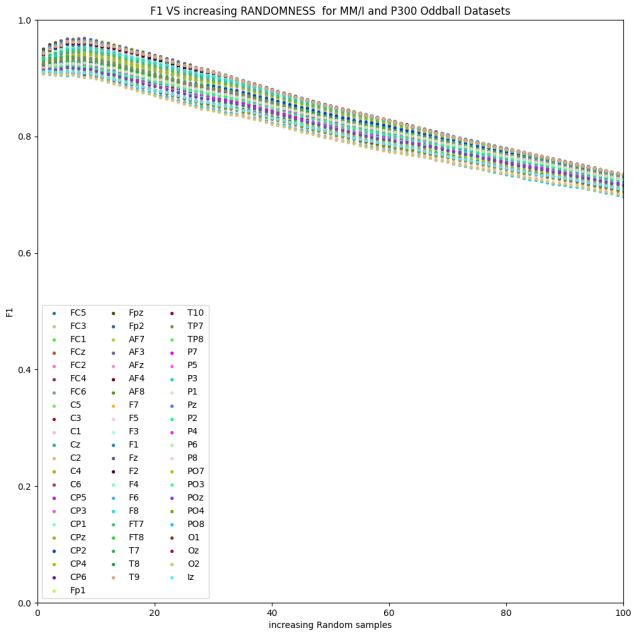


Fig. B.7 F-score vs increasing Randomness (Sc = PC_1)

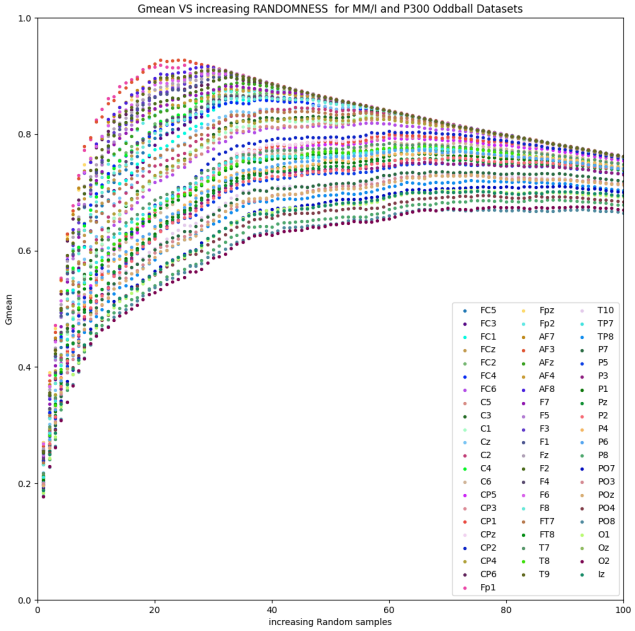


Fig. B.8 G-mean vs increasing Randomness (Sc = PC_1)

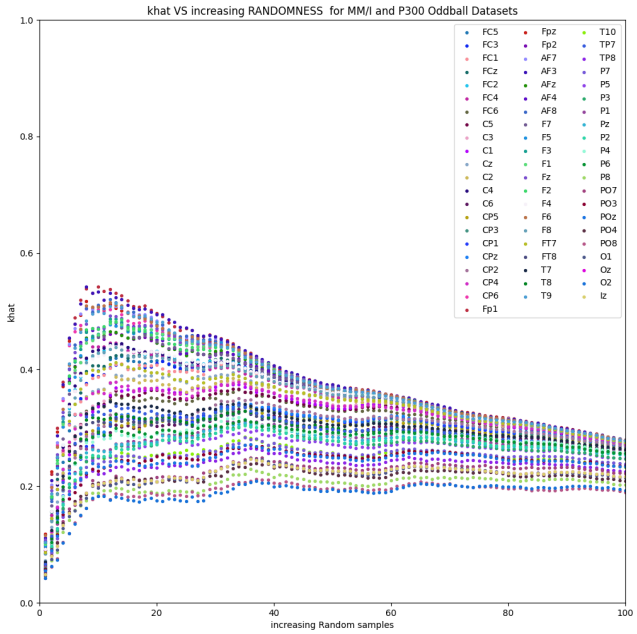


Fig. B.9 Cohen's Kappa Statistics vs increasing Randomness (Sc = PC_1)

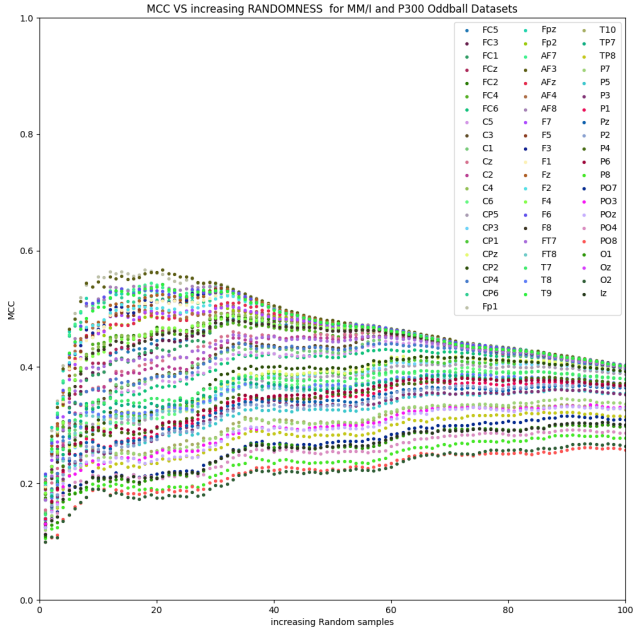


Fig. B.10 Matthew's Correlation Coefficient (MCC) vs increasing Randomness (Sc = PC_1)

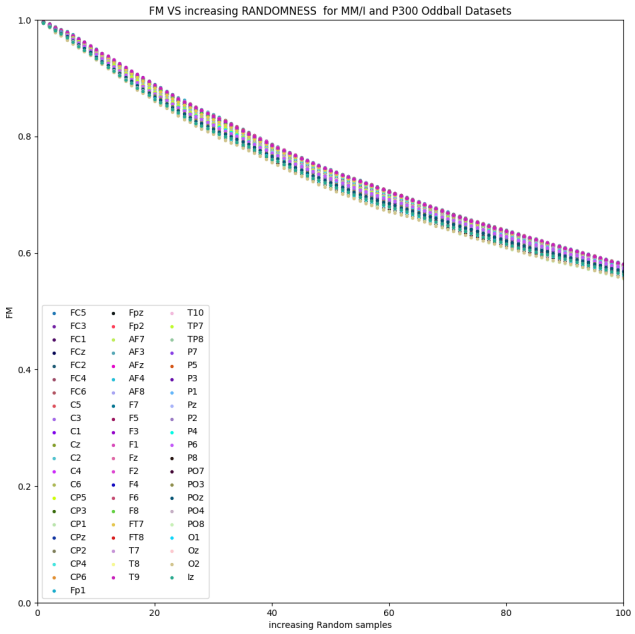


Fig. B.11 Fowles - Mallows (FM) Index vs increasing Randomness (Sc = PC_1)

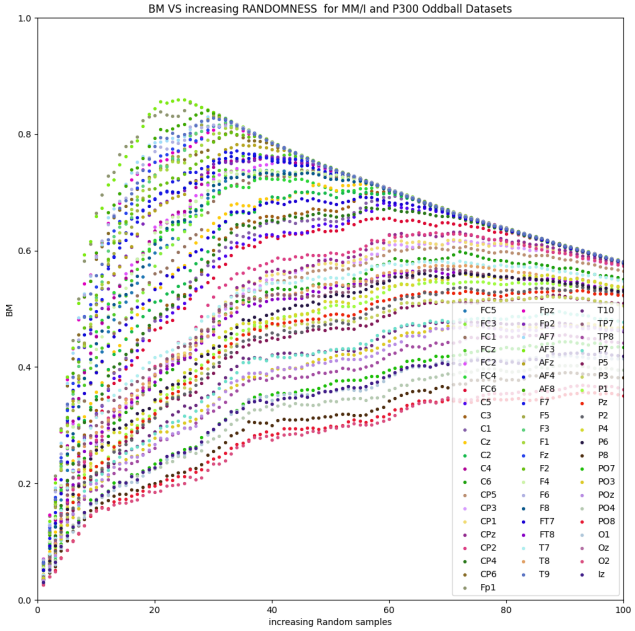


Fig. B.12 Bookmaker (BM) Informedness Rate vs increasing Randomness (Sc = PC_1)

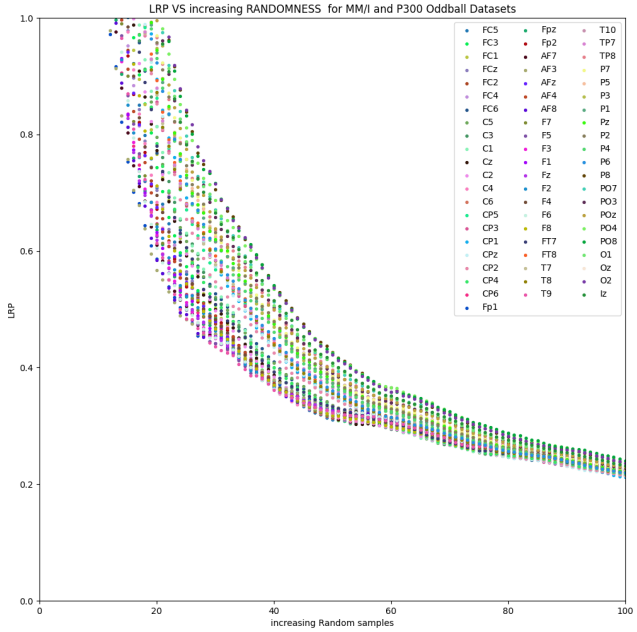


Fig. B.13 Positive Likelihood Ratio vs increasing Randomness (Sc = PC_1)

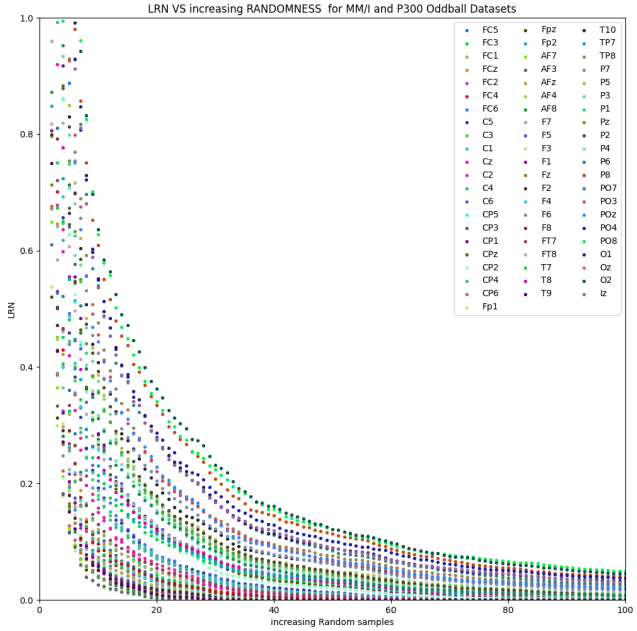


Fig. B.14 Negative Likelihood Ratio vs increasing Randomness (Sc = PC_1)

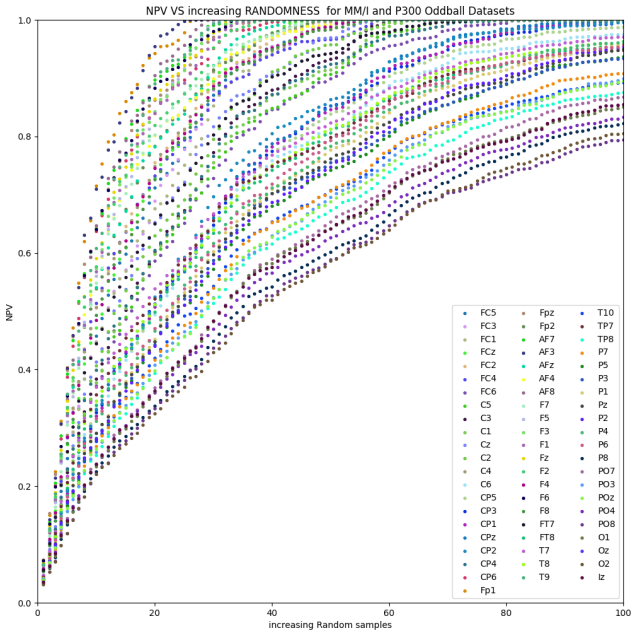


Fig. B.15 Negative Predictive Value (NPV) vs increasing randomness

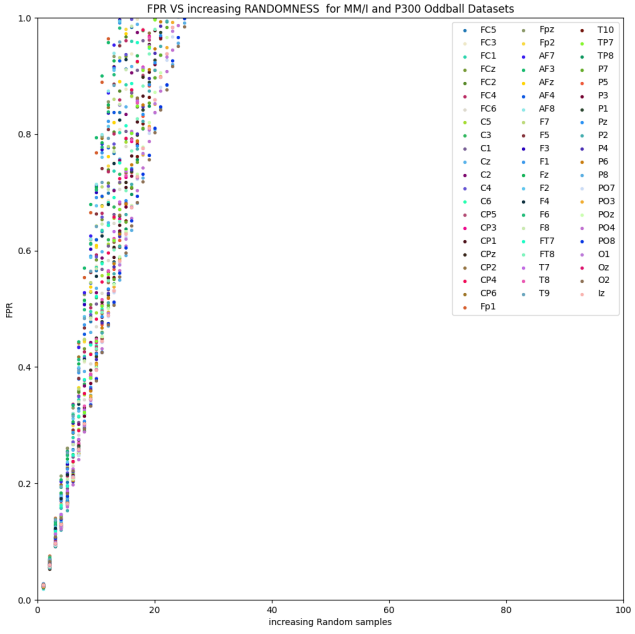


Fig. B.16 False Positive Rate (FPR) vs increasing randomness

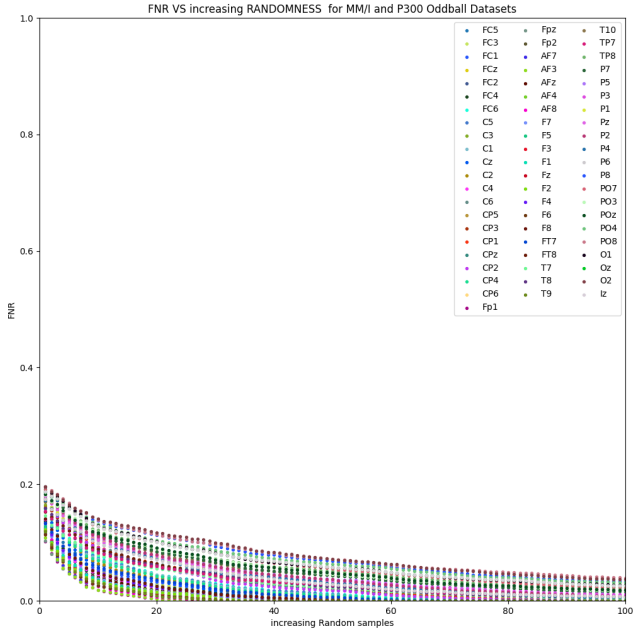


Fig. B.17 False Negative Rate (FNR) vs increasing Randomness (Sc = PC_1)

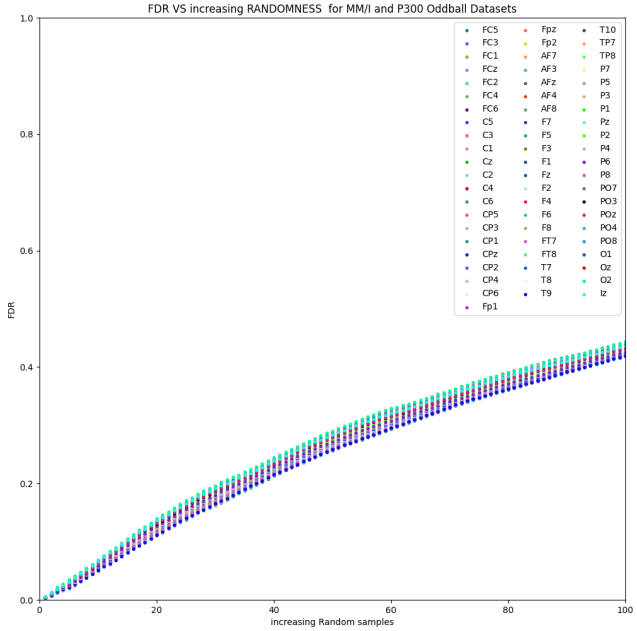


Fig. B.18 False Detection Rate (FDR) vs increasing Randomness (Sc = PC_1)

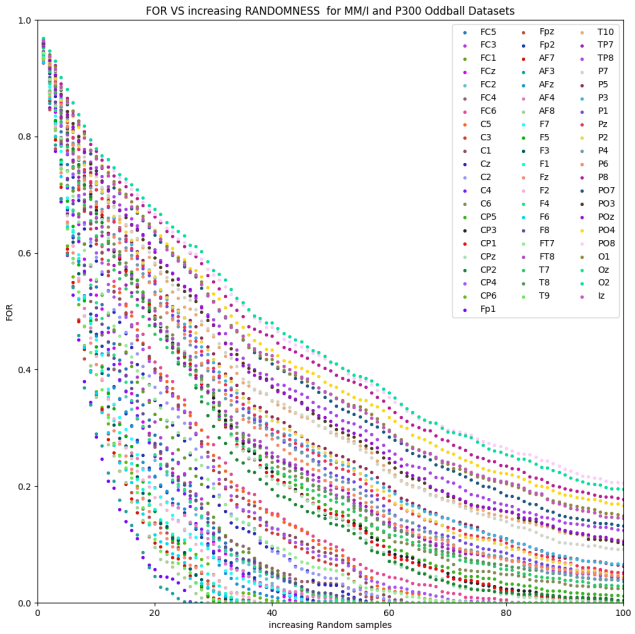


Fig. B.19 False Ommission Rate (FOR) vs increasing Randomness (Sc = PC_1)

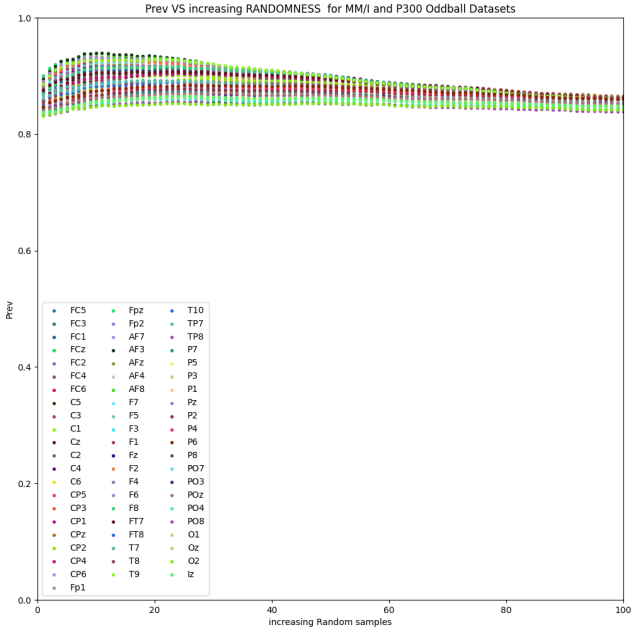


Fig. B.20 Prevalence (Prev) Rate vs increasing Randomness (Sc = PC_1)

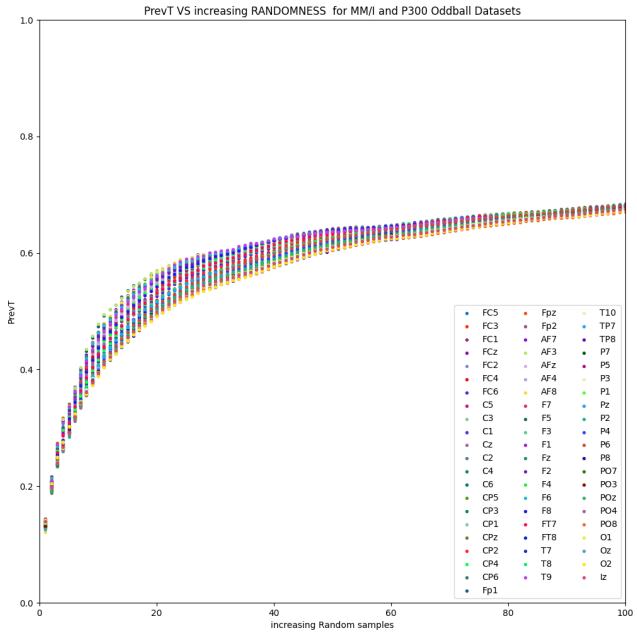


Fig. B.21 Prevalence Threshold (PrevT) Rate vs increasing Randomness (Sc = PC_1)

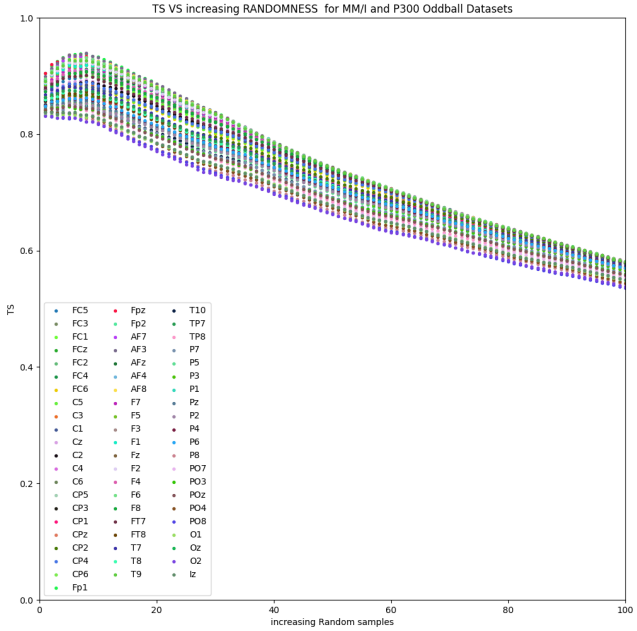


Fig. B.22 Critical Success Rate (CSI) Rate vs increasing Randomness (Sc = PC_1)

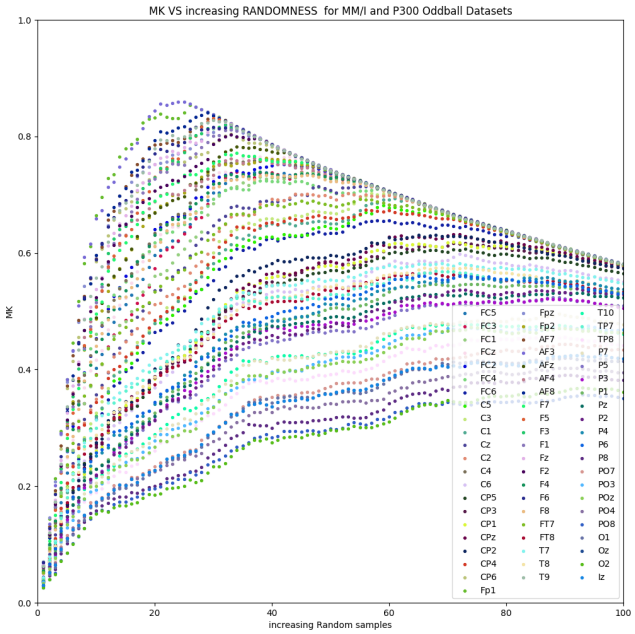


Fig. B.23 Markedness (MK) Rate vs increasing Randomness (Sc = PC_1)

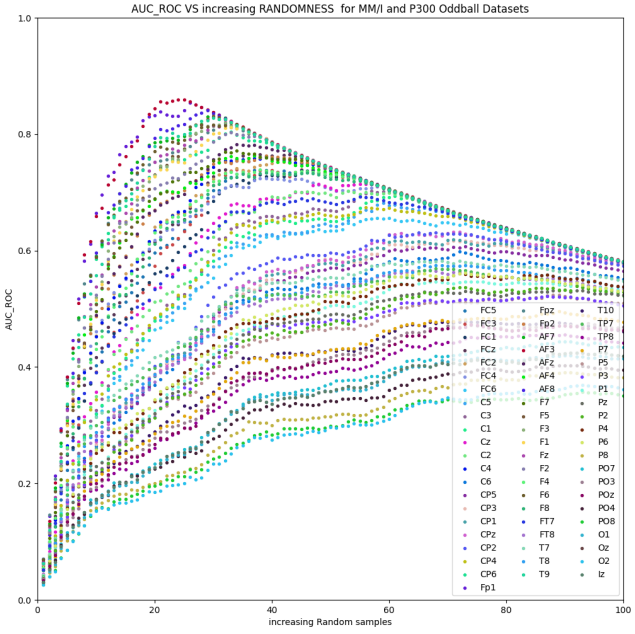


Fig. B.24 AUC-ROC vs increasing Randomness (Sc = PC_1)

Appendix C

PCA and ANN Supplementing Mathematical Discussion

C.1 Mathematics of Principal Component Analysis (PCA)

This chapter presents the mathematical concepts that helped the readers to understand the study quickly. The first section contains the basic idea of Principal Component Analysis (PCA) as dimension reduction and its mathematical concepts, followed by Artificial Neural Networks (ANNs).

Analysing complex real-world data, e.g. multi-dimensional data, takes much work by plotting the information and discovering the different patterns used to train some machine learning models. One way of discerning those patterns in dimensions is to assume that data point and regard this data point as a physical object in such a way that the dimensions are solely viewed where the data is located when it is displayed from a horizontal axis to the vertical axis. It only applies to not more than 3-dimensional data; more than that, and it is not easy to visualise and comprehend human perceptions.

Furthermore, the challenge of analysing trend patterns in multi-dimensional data with several parameters led to high computational costs to improve the understanding of the information dimensions using Principal Component Analysis. So, decreasing an information size, removing redundant dimensions and keeping only the most significant dimensions are the solution to the problem of underfitting the neural network machine learning and avoiding its overfitting.

In understanding the PCA, standard deviation (σ), variance, (var) covariance (σ_{xy}), correlation (ρ), standardization ($\dot{X}$), determinant, and eigenstructures (λ, v) have an important role in this concept.

C.1.1 Standard Deviation (σ or SD)

The standard deviation measures a collection of values' variance or dispersion. A low standard deviation implies that the values are near the set's mean (the expected value), whereas a high standard deviation shows that the values are spread out over a more extensive range.

Equation C.1 is used for the sample standard deviation.

$$\sigma_{sample} = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}} \tag{C.1}$$

C.1.2 Variance ($var(X)$) and Covariance ($cov(xy)$)

Variance is a measure of variability or how to spread the dataset. Mathematically, it is the average squared deviation from the mean score. The following Equation C.2 to compute variance $var(X)$.

$$var(X)_{sample} = \frac{\sum (x_i - \bar{x})^2}{n-1} \tag{C.2}$$

Covariance measures the extent to which corresponding elements from two sets of ordered data move in the same direction. The above formula is denoted by cov(x,y) as the covariance of x and y. Here, x_i is the value of x in i^{th} dimension. $\bar{x}$ and $\bar{y}$ denote the corresponding mean values. One way to observe the covariance ($cov(xy)$) is how interrelated two data sets are using this Equation (C.3).

$$cov(xy)_{sample} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n-1} \tag{C.3}$$

A positive covariance means X and Y are positively related, i.e. as X increases, Y also increases. Negative covariance depicts the exact opposite relation. However, zero covariance means X and Y are not related. Data analysis requires finding patterns among the data sets, so we want the data spread across each dimension. If data has high covariance when represented in some n number of dimensions, then replace those dimensions with a linear combination of those n dimensions. That data will only depend on a linear combination of those related n dimensions. (related = have high covariance)

Table C.1 Covariance vs Correlation.

Covariance	**Correlation**
Indicates the DIRECTION of linear relationship between variables	Indicates both the STRENGTH and DIRECTION of linear relationship between variables
Covariance values are not standard	Correlation values are standardised
Positive number being positive relation Negative number being negative relation	1 being strong positive correlation -1 being strong negative correlation
Values between positive infinity to negative infinity	value is strictly between -1 to 1

C.1.3 Correlation (ρ)

Correlation means association - more precisely, it measures the extent to which two variables are related, including the strength of association or relationship between variables and their direction.

$$\rho = \frac{cov_{xy}}{\sigma_x \sigma_y} \tag{C.4}$$

Table C.1 shows the difference between correlation and covariance.

C.1.4 Standardization ($\dot{X}$)

Standardisation should be done before doing PCA because the data of different scales will get misleading components. Before using any normalising approach, feature visualisation is a crucial step. Features should be shown in Equation C.5 to verify how feature values are distributed. The following characteristics were standardised for this study:

$$\dot{X} = \frac{X - \bar{X}}{\sigma} \tag{C.5}$$

X is the original feature value, the set's mean, and its standard deviation, and $\dot{X}$ is the normalised feature value, respectively. Mathematically, it can be done by subtracting the mean and dividing it by the standard deviation for each variable's value. Where $\overline{X}$ is the mean of matrix X and σ is the standard deviation of the matrix X.

C.1.5 Determinant and Generalized Variance

A determinant is an alternate measure of the total variance suggested to account for correlations between pairs of variables. When there is little connection between the several variables, this metric has a high value. If the variables, whether positive or negative, have a robust association with one another, this measure, however, only takes a little value. The determinant is also called generalised variance, a specific measure of dispersion.

The determinants are the mathematical method that is very useful in analysing and solving systems of linear equations or matrices. By definition of determinant and eigenstructure, **determinant is the sample of variance-covariance matrix for observation of multivariate vector sometimes called generalised variance**.

C.2 Ideas about Principal Component Analysis

It defines the goal of PCA -

- Find linearly independent dimensions (or basis of views) to represent the data points more clearly.
- Those newly found dimensions should allow us to predict/reconstruct the original dimensions. The reconstruction/projection error should be minimised.

To understand the projection error. Suppose to transform a two-dimensional representation of data points into a one-dimensional representation. We will find a straight line and project data points on them. (A straight line is one-dimensional). There are many possibilities for selecting a straight line. There are two possibilities, as shown in Figure 3.6 - the line traverse line will be our new dimension by rotating to create one dimension, i.e. the perpendicular distance of each data point from the straight line is the projection error. The sum of the error of all data points will be the total projection error. Our new data points will be the projections of those original data points. The transformed two-dimensional data points to one-dimensional data points by Projecting them on one-dimensional Space, i.e. a straight line. That traverse straight line is called the principal axis. Since the Projection to a single dimension only has one principal axis.

The second choice of the straight line is better because -

- The projection error is less than that in the first case.
- Newly projected points are more widely spread than in the first case. i.e. more variance.

The two points mentioned above are related, i.e. to minimise the reconstruction error, the variance will increase. The calculation of the covariance matrix of the original data set matrix A by transforming the original data points such that the covariance matrix of transformed data points is diagonal.

The determinant of a square symmetric matrix $A_{n,n}$ whose diagonal elements D_n (λ) are sample variances and whose off-diagonal elements are sample covariances. Symmetry means that the matrix and its transpose are identical.

A dimension reduction technique finds the variance-maximising directions to project the data as shown in Figure 3.6a and 3.6b.

$$A = \begin{bmatrix} D_a & C_{a,b} & C_{a,c} & C_{a,d} & C_{a,e} \\ C_{a,b} & D_b & C_{b,c} & C_{b,d} & C_{b,e} \\ C_{a,c} & C_{b,c} & D_c & C_{c,d} & C_{c,e} \\ C_{a,d} & C_{b,d} & C_{c,d} & D_d & C_{d,e} \\ C_{a,e} & C_{b,e} & C_{c,e} & C_{d,e} & D_e \end{bmatrix} \tag{C.6}$$

where D is the diagonal; $C_(a,b) \rightarrow$ covariance along dimension a and b

C.2.1 Eigenvalue (λ) and Eigenvector (v)

Given a symmetric matrix $A \in \mathbb{R}^{n \times n}$, a real scalar λ is said to be an eigenvalue of A if there exists a non zero vector $v \in \mathbb{R}^n$ called eigenvector, such that we have :

$$Av = \lambda v \tag{C.7}$$

The vector v is then referred to as an eigenvector associated with the eigenvalue λ. The eigenvector v is said to be normalized if $|v^T v| = 1 = I$, so

$$v^T Av = \lambda v^T v = I\lambda \tag{C.8}$$

The eigenvector v refers to the direction along A in the scalar multiplication of eigenvalue λ. The eigenvalues λ of the matrix A employs characteristic equation, to find the eigenvalues $\lambda_{1,2,3,\ldots,n}$

$$det(\lambda I - A) = 0 \tag{C.9}$$

Note: From the fundamental theorem of algebra, any polynomial of degree n has n (possibly not distinct) complex roots. The eigenvalues are real for symmetric matrices since $\lambda = v^T Av$ when $Av = \lambda v$, and u is normalised.

C.2.2 Spectral Theorem

The spectral theorem, also known as the symmetric eigenvalue decomposition (SED) theorem, is a critical finding in linear algebra that states that for any symmetric matrix, there are precise and possibly not distinct n real eigenvalues that the associated eigenvectors can be chosen to form an orthonormal basis. The outcome provides a straightforward method for decomposing the symmetric matrix into its parts.

Theorem: Let any symmetric $A \in \mathbb{R}^{n\times n}$, then A is diagonalized by a real orthogonal matrix $U \in \mathbb{R}^{n\times n}$ (that is, $U^T U = UU^T = I_n$). Then

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T = U\Lambda U^T, \lambda = diag(\lambda_1, \lambda_2, \lambda_3, ..., \lambda_n) \tag{C.10}$$

Remarks: The eigenvector associated with the largest eigenvalue is the principal eigenvector of matrix A.

By using the principal eigenvector of PCA, it rotates the figure C.1a shows the 3-D example of datasets before the PCA process to figure C.1b, so it is easy to visualise the n^{th} dimension dataset.

C.3 PCA Algorithm

The Principal Component Analysis (PCA) procedure is a dimension reduction technique that projects the data on k dimensions by maximising the variance of the data as follows:

Step 1: Normalise the data with a mean of zero (0) and a standard deviation of 1.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \bar{x}_j}{\sigma_j} \tag{C.11}$$

where

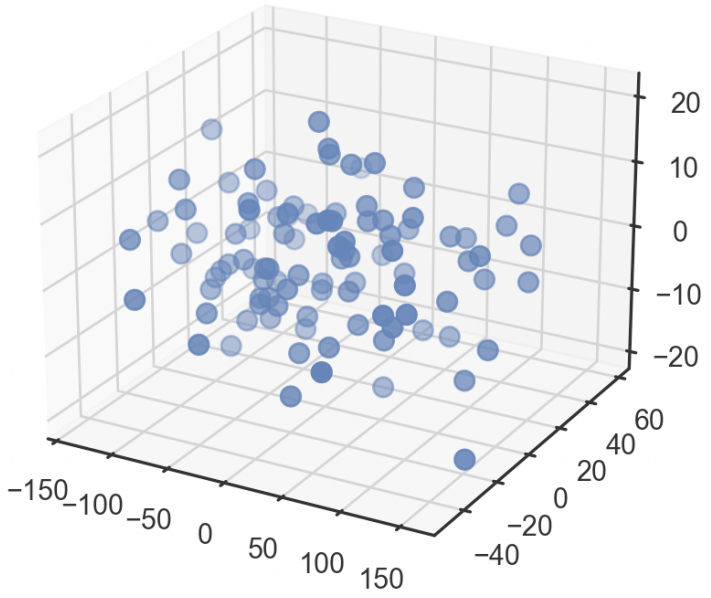
$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \tag{C.12}$$

and

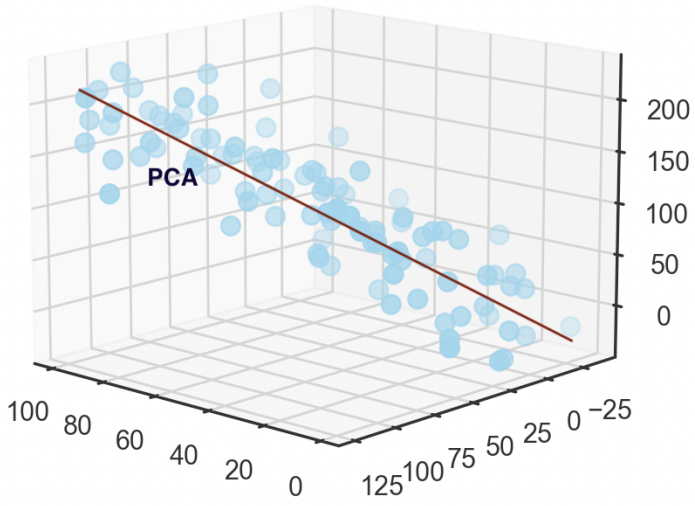
$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \bar{x}_j)^2 \tag{C.13}$$

Step 2: Compute $\sum = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} x_j^{(i)^T} \in \mathbb{R}^{n\times n}$, which is symmetric with real eigenvalues.

Step 3: Compute $v_1, v_2, ..., v_k \in \mathbb{R}^n$ the k orthogonal principal eigenvectors i.e., the k largest eigenvalues.



(a) Projection of data points in 3 dimensions.



(b) Projection of data points in newly constructed 3-dimensions.

Fig. C.1 PCA in 3-Dimensional Space

Step 4: Project the data on span $v_1, v_2, ..., v_k$. It maximises the variance among all k-dimensional spaces.

C.3.1 Procedure in Performing PCA - Example

Principal Component Analysis (PCA) finds a new set of dimensions (or a set of the basis of views) such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them. It means a more important Principal axis occurs first (more important = more variance/more spread out data [165].

1. **Standardization of matrix** A

   Assume the matrix A,

$$A = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \\ 1 & 2 & 3 & 4 \\ 5 & 5 & 6 & 7 \\ 1 & 4 & 2 & 3 \\ 5 & 3 & 2 & 1 \\ 8 & 1 & 2 & 2 \end{bmatrix} \tag{C.14}$$

   It needs to standardise the matrix, using the equation C.5, resulting to mean $\bar{A}$ and standard deviation σ in equation C.15 and C.16, respectively, before standardisation $\dot{A}$.

$$\bar{X} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \\ 4 & 3 & 3 & 3.4 \end{bmatrix} \tag{C.15}$$

$$\sigma = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \\ 3 & 1.58114 & 1.73205 & 2.30217 \end{bmatrix} \tag{C.16}$$

   The matrix A has been transformed as:

$$\dot{X} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \\ -1 & -0.63246 & 0 & 0.26062 \\ 0.3333 & 1.26491 & 1.73205 & 1.56374 \\ -1 & 0.63246 & -0.57735 & -0.17375 \\ 0.3333 & 0 & -0.57735 & -1.04249 \\ 1.3333 & -1.26491 & -0.57735 & -0.60812 \end{bmatrix} \tag{C.17}$$

Calculate the covariance matrix A of data points by equation C.23.

The covariance matrix A ($cov(\dot{A})$) will be calculated as,

$$cov(\dot{X}) = \begin{bmatrix} & y_1 & y_2 & y_3 & y_4 \\ y_1 & var(y_1) & cov(y_1,y_2) & cov(y_1,y_3) & cov(y_1,y_4) \\ y_2 & cov(y_2,y_1) & var(y_2) & cov(y_2,y_3) & cov(y_2,y_4) \\ y_3 & cov(y_3,y_1) & cov(y_3,y_2) & var(y_3) & cov(y_3,y_4) \\ y_4 & cov(y_4,y_1) & cov(y_4,y_2) & cov(y_4,y_3) & var(y_4) \end{bmatrix} \tag{C.18}$$

$$cov(\dot{X}) = \begin{bmatrix} & y_1 & y_2 & y_3 & y_4 \\ y_1 & 0.8 & -0.25298 & 0.03849 & -0.14479 \\ y_2 & -0.25298 & 0.8 & 0.51121 & 0.4945 \\ y_3 & 0.03849 & 0.51121 & 0.8 & 0.75236 \\ y_4 & -0.14479 & 0.4945 & 0.75236 & 0.8 \end{bmatrix} \tag{C.19}$$

2. **Calculate eigenvectors and corresponding eigenvalues.** When that linear transformation is performed to a nonzero vector, an eigenvector is a vector that changes by a scalar amount. The matching eigenvalue determines the eigenvector's scaling factor to satisfy the eigenstructure through the equation C.20 where $cov(\dot{A})$ is a square matrix, λ is an eigenvalue, and v is an eigenvector.

$$[cov(\dot{A})] \cdot [v] = [\lambda] \cdot [v] \tag{C.20}$$

where identity matrix I_n as shown below

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.21}$$

$$(cov(\dot{A}) - \lambda \cdot I)[v] = 0 \tag{C.22}$$

Furthermore, if the determinant of the (square) matrix is precisely zero, the matrix is said to be singular, and it has no inverse as shown in Equation C.23.

$$det(cov(\dot{A}) - \lambda \cdot I) = 0 \tag{C.23}$$

then,

$$0 = \begin{bmatrix} 0.8-\lambda & -0.25298 & 0.03849 & -0.14479 \\ -0.25298 & 0.8-\lambda & 0.51121 & 0.4945 \\ 0.03849 & 0.51121 & 0.8-\lambda & 0.75236 \\ -0.14479 & 0.4945 & 0.75236 & 0.8-\lambda \end{bmatrix} \tag{C.24}$$

we get,

$$\lambda = 2.51579324, 1.0652885, 0.39388704, 0.02503121 \tag{C.25}$$

Equation C.22 is used to find the v with different λ values.

$$\begin{bmatrix} 0.8-\lambda & -0.25298 & 0.03849 & -0.14479 \\ -0.25298 & 0.8-\lambda & 0.51121 & 0.4945 \\ 0.03849 & 0.51121 & 0.8-\lambda & 0.75236 \\ -0.14479 & 0.4945 & 0.75236 & 0.8-\lambda \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = 0 \tag{C.26}$$

Using Kramer's Rule, the highest λ ($\lambda_1 = 2.51579324$):

$$\begin{aligned} v_1 &= 0.16195986 \\ v_2 &= -0.52404813 \\ v_3 &= -0.58589647 \\ v_4 &= -0.59654663 \end{aligned} \tag{C.27}$$

We solved for other λ_n, commonly known as Principal Component (PC_n), and got eigenvectors v and loading scores.

$$\begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 \\ 0.16195986 & -0.917059 & -0.307071 & 0.196162 \\ -0.52404813 & 0.206922 & -0.817319 & 0.120610 \\ -0.58589647 & -0.320539 & 0.188250 & -0.720099 \\ -0.59654663 & -0.115935 & 0.449733 & 0.654547 \end{bmatrix} \tag{C.28}$$

3. **Sort the eigenvectors according to their eigenvalues in decreasing order. Choose the first k eigenvectors, the new k dimensions.**

   Since choosing the highest eigenvalues λ with corresponding its eigenvectors v,

$$\begin{bmatrix} \lambda_1 & \lambda_2 \\ 0.16195986 & -0.917059 \\ -0.52404813 & 0.206922 \\ -0.58589647 & -0.320539 \\ -0.59654663 & -0.115935 \end{bmatrix} \tag{C.29}$$

4. **Transform the original n dimensional data points into k dimensions.**

   Feature matrix * top k eigenvectors = Transformed Data

$$\begin{bmatrix} -1 & -0.6325 & 0 & 0.2606 \\ 0.3333 & 1.2649 & 1.7321 & 1.5637 \\ -1 & 0.6325 & -0.5774 & -0.1738 \\ 0.3333 & 0 & -0.5774 & -1.0425 \\ 1.3333 & -1.264 & -0.5774 & -0.6081 \end{bmatrix} \times \begin{bmatrix} 0.16196 & -0.9171 \\ -0.5240 & 0.2069 \\ -0.5859 & -0.3205 \\ -0.5965 & -0.1159 \end{bmatrix} \tag{C.30}$$

$$= \begin{bmatrix} -0.0140 & 0.7560 \\ 2.5565 & -0.7804 \\ 0.05148 & 1.2531 \\ -1.01415 & 0.00024 \\ -1.5799 & -1.2289 \end{bmatrix} \tag{C.31}$$

   Therefore, the principal Component is the eigenvectors $|v_{PC}|$ with the highest eigenvalue λ_{PC}, which is the loading scores.

$$|v_{PC}| = \begin{bmatrix} 0.16196 \\ 0.52409 \\ 0.5859 \\ 0.5965 \end{bmatrix} \tag{C.32}$$

C.3.2 Explained Variance Ratio and F-Distribution

Given the definition of variance (S^2) in the Equation C.2 and F- Distribution (F) formula, PC is the distribution of the samples in PC spaces, so $F = PC$.

$$F = \frac{S_1^2}{S_2^2} \tag{C.33}$$

By the definition of eigenvalues (λ),

$$S^2 = \lambda \tag{C.34}$$

Then, from the Equation C.9.

$$det(\lambda I - A_{mxn}) = 0 \tag{C.35}$$

So,

$$\sum_{1}^{n} \lambda_k = \lambda_1 + \lambda_2 + ... + \lambda_n \tag{C.36}$$

Furthermore,

$$1 = \frac{\lambda = \lambda_1 + \lambda_2 + ... + \lambda_n}{\sum_1^n \lambda_k} \tag{C.37}$$

$$s_T^2 = \sum_{1}^{n} \lambda_k; s_1^2 = \lambda_1; s_2^2 = \lambda_2; ....; s_n^2 = \lambda_n \tag{C.38}$$

Moreover,

$$1 = \frac{\lambda_1}{\sum_1^n \lambda_k} + \frac{\lambda_2}{\sum_1^n \lambda_k} + ... + \frac{\lambda_n}{\sum_1^n \lambda_k} \tag{C.39}$$

$$1 = \frac{S_1^2}{S_T^2} + \frac{S_2^2}{S_T^2} + ... + \frac{S_n^2}{S_T^2} \tag{C.40}$$

Therefore,

$$1 = F_1 + F_2 + ... + F_{n-1} + F_n \tag{C.41}$$

which F_1 serves as the first explained variance ratio, F_2 serves as the second explained variance ratio, then so on and so forth. By definition, F-distribution is the number of frequencies or samples in the different distribution of dimension.

So,

$$1 = PC_1 + PC_2 + ... + PC_{n-1} + PC_n \tag{C.42}$$

As described in Figure C.2.

$$1 \leq F_1 \leq F_2 \leq ... \leq F_{n-1} \leq F_n \leq 0 \tag{C.43}$$

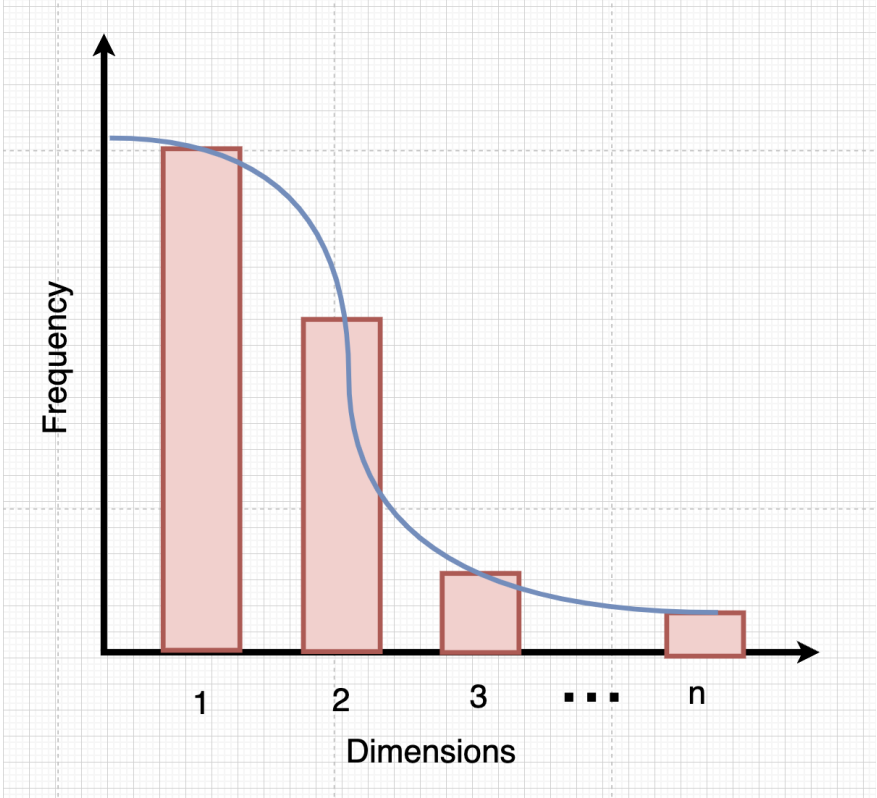


Fig. C.2 Frequency Distribution Diagram

C.3.3 PCA Typical Python Implementation

Here is the Python implementation of typical PCA using the sklearn library by solving for transformed data and its loading scores v_{PC}. Moreover, Figure C.3 shows the implementation output.

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA

A = np.matrix ([[1,  2, 3, 4],
[5, 5, 6, 7],
[1, 4, 3, 2],
[5, 3, 2, 1],
[8, 1, 2, 2]])
df = pd.DataFrame(A, columns = [ 'f1',  'f2', 'f3',  'f4'])
df_std = (df - df.mean()) / (df.std())
n_components = 2
pca = PCA(n_components = n_components)
principalComponents = pca.fit_transform (df_std)
principal = pd.DataFrame(data = principalComponents,
columns = [ 'nf' + str (i+1) for i in range (n_components)])
print(principalDF)

loading_scores = pd.Series(pca.components_[0]).abs()
print(loading_scores )
```

The following shows the generation of explained variance ratio.

```
print('list of PC variance: \n ',  per_var)
per_var_passingrate = per_var[0]
passrate = per_var_passingrate
```

Also shows the generation of the PCA Scree Plot.

```
per_var = np.round(pca.explained_variance_ratio_ *  100,  decimals = 5)
labels = ['PC' + str(x) for x in range(1, len(per_var)+1)]
plt.bar(x = range (1,  len(per_var) + 1),
```

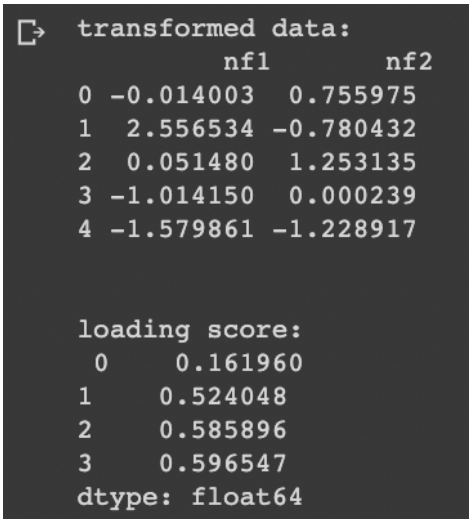


Fig. C.3 Example output of the PCA Implemenattion

```
list of PC variance:
  [7.067842e+01 2.925465e+01 6.693000e-02 0.000000e+00]
```

Fig. C.4 Example Explained Variance Ratio

```
height = per_var,  tick_label = labels)
plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Component')
plt.title('Scree Plot')
plt.show()
```

And the following code makes a fancy-looking plot using PC1 and PC2.

```
pca_df = pd.DataFrame(pca_data,  columns = labels)
plt.scatter(pca_df.PC1,  pca_df.PC2)
plt.title('My PCA Graph')
plt.xlabel('PC1 - {0}%'.format(per_var[0]))
plt.ylabel('PC2 - {0}%'.format(per_var[1]))

for sample in pca_df.index:
    plt.annotate(sample, (pca_df.PC1.loc[sample],  pca_df.PC2.loc[sample]))
plt.show()


#generating parameters
```

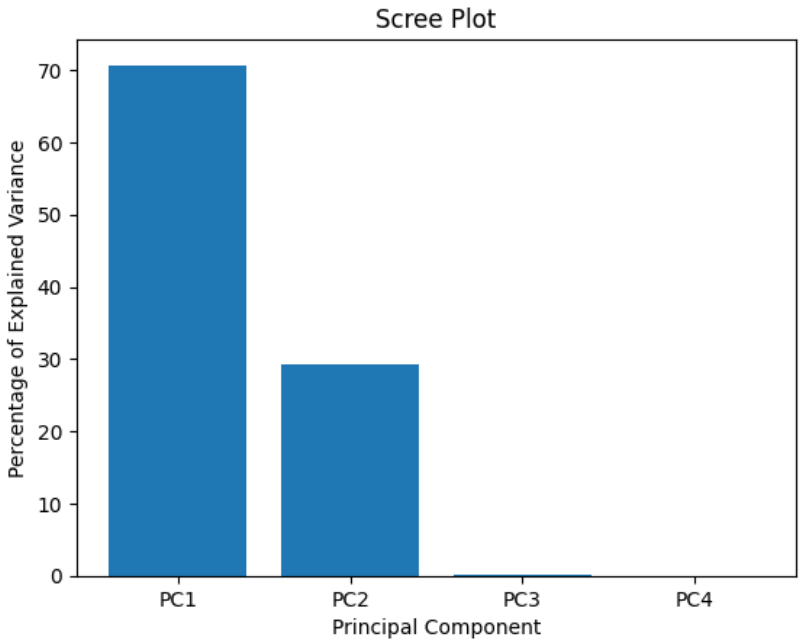


Fig. C.5 Example Scree Plot

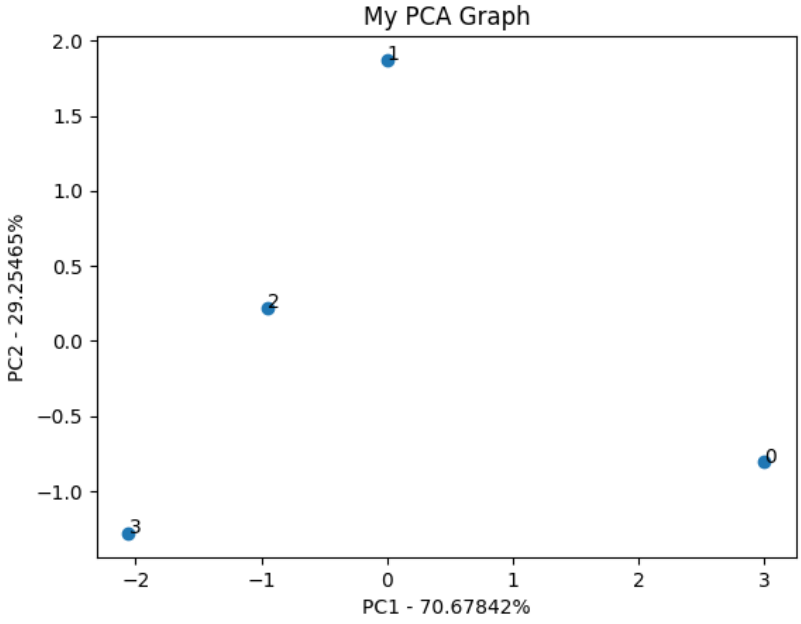


Fig. C.6 Example PCA 2D Space Graph

```
print('list of PC variance: \n ',per_var)
per_var_passingrate = per_var[0]
#per_var_passingrate = per_var[0]
passrate = per_var_passingrate
print('recommended Sc: ',  per_var_passingrate, " %")
```

C.4 Artificial Neural Network (ANN).

Artificial Neural Networks (ANNs) are machine learning models built with layers. An ANN learning algorithm is an attribute that adjusts ANN weights to achieve the required output(s) for the specified input(s) with adaptivity and capability to cope with deviations in the situation. It relates to ANN retraining when an available new information set has been introduced.

A graph that cannot learn (or be corrected by its weights) cannot be listed as an ANN, as it cannot hold the adaptation property. Moreover, when ANN sets in a change in the environment, there is no need to construct a new ANN model with the improved new data once an ANN model is created.

Training data is essential for neural networks to develop and enhance their accuracy over time. Nonetheless, these learning algorithms become practical tools in computer science and artificial intelligence if they are adjusted for accuracy, enabling us to categorise and cluster data quickly.

Artificial Neural Networks are used in a wide range of applications, including image recognition, machine translation, medical diagnosis, and most typical random function approximation, which is an ability to work with incomplete knowledge or information.

ANN is said to be if it has:

- Start Element (SE), A input node in a directed graph in Start Element (SE) k is, which gets an input I_{ij} from the input matrix $I = I_{ij}$; i = 1, 2, 3, . . . ,n; j = 1, 2, 3, . . . ,m of n features of m independent accounts, and beginning of the flow in the graph.

- End Element (EE), A output node in a directed graph in End Element (EE) i is which produces an output O_{ij} from the output matrix $O = O_{ij}$; i = 1, 2, 3, . . . ,n; j = 1, 2, 3, . . . ,m of n desired outputs of m independent input accounts and ends a flow in the graph.

- Processing Element (PE), namely weights, summing function, activation function, output and bias,

- Nodes use as Processing Elements (PEs), excluding the start and end nodes,

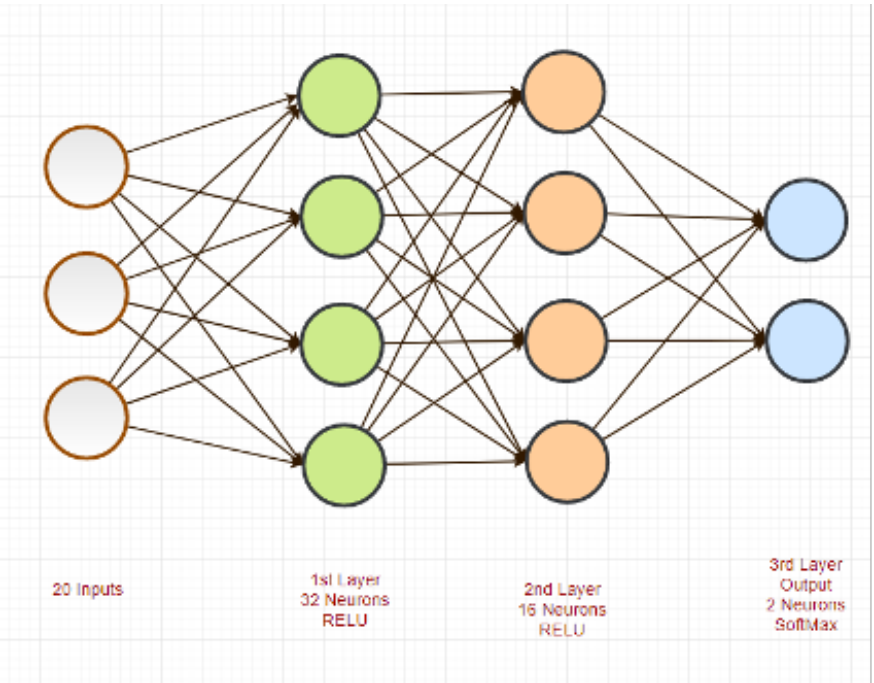


Fig. C.7 Artificial Neural Network

- A state variable n_i connected with each node i,
- weight w_{ki} is a real-valued connected with each link (ki) to node i from node k
- bias b_i is a real-valued linked with each node i,
- a learning algorithm that supports to model of the anticipated output for the given input,
- a flow on each link (ki) to node i from node k, that transmits precisely the same flow which the same to n_k by the output of node k,
- Each start node is linked to a minimum of one end node, and each end node is associated with a minimum of one start node.
- no parallel edges (each link (ki) to node i from node k is unique)

C.4.1 Mathematical Derivation of Artificial Neural Network

To generate a neural network needs the following:

- input set $(1,2,...,n-1,n)$ $(i_1, i_2, i_3, \ldots, i_{n-1}, i_n)$

- output set $(1,2,...,m-1,m)$ (o_1,o_2,o_3, ... , o_{n-1}, o_n)
- at least one hidden layers
- at least one neuron in each hidden layer

Optional properties:

- Each neuron has a bias in the hidden layer(s) (b_1,b_2) and output layer (b_3 , b_4)
- Bias's weight in the hidden layer(s) (v_{11}, v_{12}) and in output layer (v_{21}, v_{22})
- Weights connecting neurons (w_1 w_2 - for input to hidden, w_3 - for hidden to output)

In a neural network, the weights are informed to minimise the error between the inputs and outputs as it is trained. Therefore, the weights are repetitively updated by specifying the starting weights regardless, providing a starting point for final weight values. It can be noted that the bias (b_i) and also its weight/s are non-compulsory; due to differentiation, it does not hinge on anything and is evaluated to 0.

C.4.2 Feed Forward Neural Network

Given the network N, it has two (2) inputs (i_1, i_2), one (1) hidden layer with one (1) neuron (h_1), one (1) bias (b_1) and 1 output (o_1).

Calculate the value of h_1.

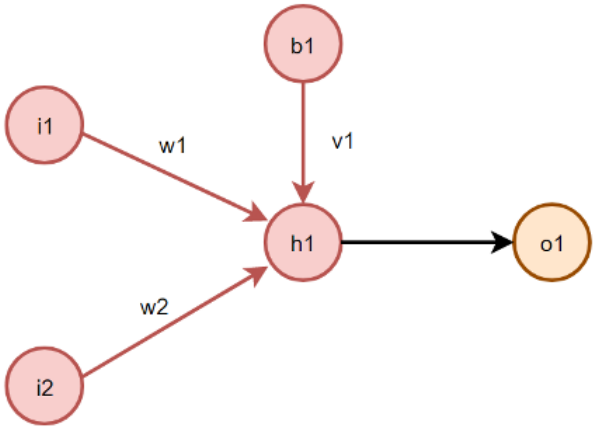


Fig. C.8 One Layer Neural Network.

In Figure 1, h_1 hinge on i_1, with weight w_1, i_2 is with weight w_2, and b_1 with weight v_1. To compute the net value of h_1. Multiply the cost of the neuron and its weight. It can be expressed by,

$$\sum_{m}^{n} i_m w_m + b_k v_k \tag{C.44}$$

Therefore,

$$h_1 = i_1 w_1 + i_2 w_2 + b_1 v_1 \tag{C.45}$$

The value of h_1 can be activated with functions like sigmoid, RELU (Rectified Linear Unit), tanh, SoftMax, etc., shown in Figure C.9. However, in this example, Sigmoid Function is used.

A sigmoid neuron outputs a continuous plane range of values between 0 and 1. As exponential functions are like to handle mathematically and since learning algorithms comprise a great deal of differentiation, selecting a correct function that is computationally inexpensive to handle is excellent. The function is well-defined to be:

$$sig(x) = \frac{1}{1+e^{-x}} \tag{C.46}$$

Therefore, the output of h_1 is

$$sig(h_1) = \frac{1}{1+e^{-h_1}} \tag{C.47}$$

Work out for $sig(x)$ for o_1, the results for o_1 are as follows:

$$sig(o_1) = \frac{1}{1+e^{-o_1}} \tag{C.48}$$

It can be noted that with the use of the sigmoid value of $o_1 \rightarrow sig(o_1)$, we can use the Euclidean Norm in working out the total error, that is

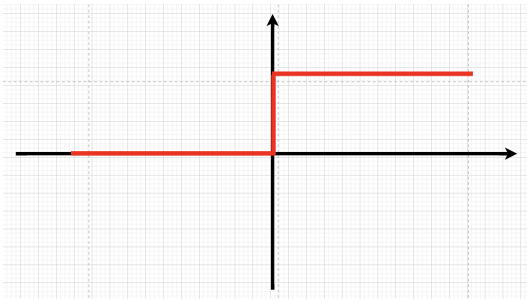
$$E_{total} = \frac{1}{2}\sum_{1}^{n}(target(o_k) - sig(o_k))^2 \tag{C.49}$$

Therefore, our Total Error is:

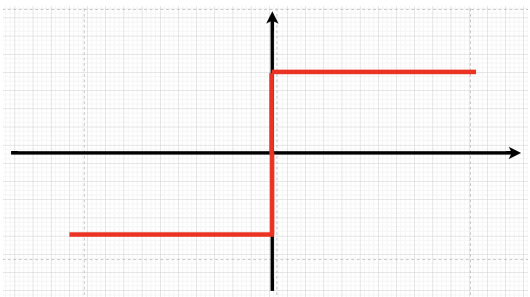
$$E_{total} = \frac{1}{2}(target(o_k) - sig(o_k))^2 \tag{C.50}$$

The effect of $sig(o_1)$ on the total error using partial differentiation. From E_{total}, it depends on 2 arguments, target (o_1) and $sig(o_1)$. Compute for the partial derivative means differentiation on only one variable, but not all.

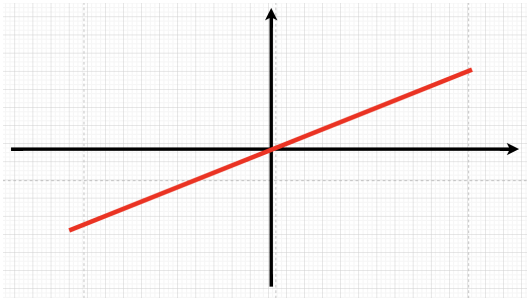
The $sig(o_1)$ affects the total error on the equation for E_{total}.



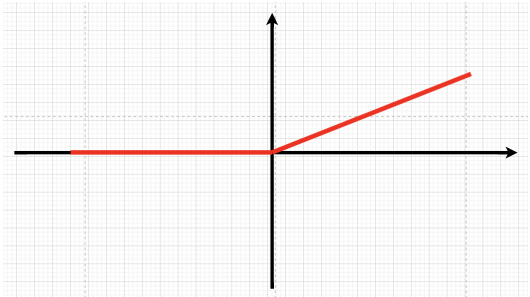
(a) Step Function



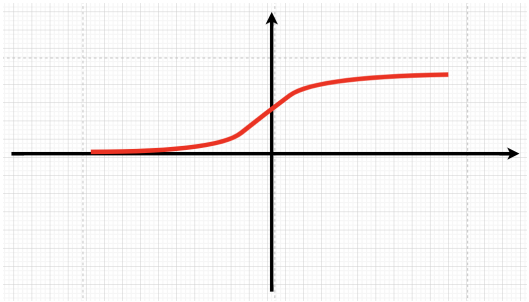
(b) Signum Function



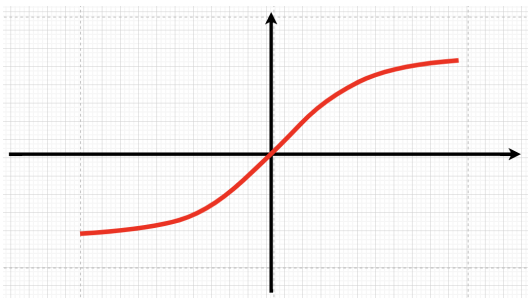
(c) Linear Function



(d) RELU Function



(e) Sigmoid / Softmax Function



(f) Hyperbolic Tangent Funtion

Fig. C.9 Artificial Neural Networks (ANNs) Activation Functions

$$\frac{\partial E_{total}}{\partial sig(o_1)} = sig(o_1) - target(o_1) \tag{C.51}$$

Calculate using partial differentiation and the chain rule to get from

$$\frac{\partial E_{\text{total}}}{\partial \operatorname{sig}(o_1)} = \frac{\partial \frac{1}{2}(\operatorname{target}(o_1) - \operatorname{sig}(o_1))^2}{\partial \operatorname{sig}(o_1)} = \operatorname{sig}(o_1) - \operatorname{target}(o_1) \tag{C.52}$$

$$\begin{aligned} &\frac{\partial \frac{1}{2}(\operatorname{target}(o_1) - \operatorname{sig}(o_1))^2}{\partial \operatorname{sig}(o_1)} \\ &\qquad \rightarrow 2 * \frac{1}{2}(\operatorname{target}(o_1) - \operatorname{sig}(o_1))^{2-1} * \frac{\partial(\operatorname{target}(o_1) - \operatorname{sig}(o_1))}{\partial \operatorname{sig}(o_1)} \\ &\qquad \rightarrow 1 * (\operatorname{target}(o_1) - \operatorname{sig}(o_1)) * (-1) \\ &\qquad \rightarrow \operatorname{sig}(o_1) - \operatorname{target}(o_1) \end{aligned} \tag{C.53}$$

C.4.3 Back Propagation

Backpropagation is a method to update the weights in the ANNs by considering the actual and desired outputs. The derivative concerning top weights w is computed using Chain Rule.

The aim is to know how the weights affect the total error. In w_1, to calculate

$$\frac{\partial \mathrm{E}_{\text{total}}}{\partial \mathrm{w}_1} \tag{C.54}$$

E_{total} depends on $sig(o_1)$, $sig(o_1)$ depends on $net(o_1)$, $net(o_1)$ depends on w_1. So E_{total} does depend on w_1.

Therefore, in getting E_{total} depends on w_1 the partial derivative can be expressed in:

$$\begin{aligned} \frac{\partial \mathrm{E}_{\text{total}}}{\partial \mathrm{w}_1} &= \frac{\partial \mathrm{E}_{\text{total}}}{\partial \operatorname{sig}(o_1)} * \frac{\partial \operatorname{sig}(\mathrm{o}_1)}{\partial \operatorname{net}(\mathrm{o}_1)} * \frac{\partial \operatorname{net}(\mathrm{o}_1)}{\partial \mathrm{w}_1} \\ \frac{\partial E_{\text{total}}}{\partial \operatorname{sig}(o_1)} &= \operatorname{sig}(o_1) - \operatorname{target}(o_1) \end{aligned} \tag{C.55}$$

To know how $sig(o_1)$ depends on $net(o_1)$, express the partial derivative in sigmoid function

$$\operatorname{sig}(o_1) = \frac{1}{1 + e^{-\operatorname{net}(o_1)}} \tag{C.56}$$

Then,

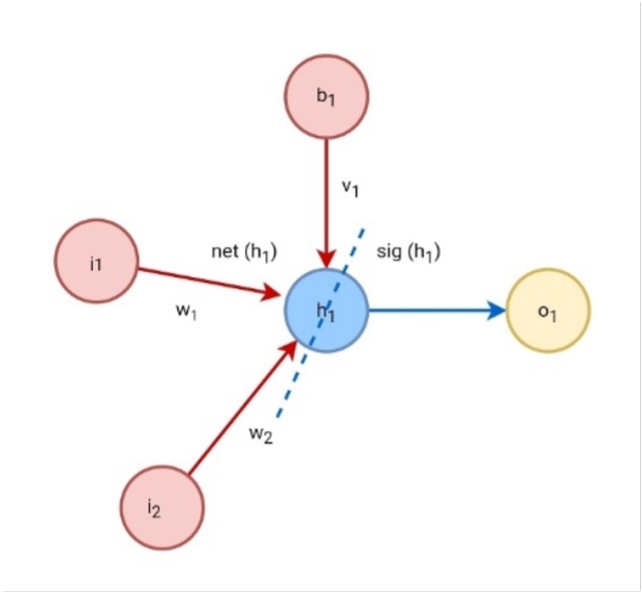


Fig. C.10 Deriving the Total error

$$\frac{\partial \operatorname{sig}(o_1)}{\partial \operatorname{net}(o_1)} = \operatorname{sig}(o_1) * (1 - \operatorname{sig}(o_1)) \tag{C.57}$$

To find how $net(o_1)$ depends on w_1 and b_1 using the partial derivative from

$$\begin{aligned} \operatorname{net}(o_1) &= i_1 w_1 + i_2 w_2 + b_1 v_1 \\ \frac{\partial \operatorname{net}(o_1)}{\partial w_1} &= i_1 \\ \frac{\partial \operatorname{net}(o_1)}{\partial b_1} &= v_1 \end{aligned} \tag{C.58}$$

where $v_1 = 1$ because it is said to be a biased line weight.

Looking back at the definition of $net(o_1)$.

$$\begin{aligned} \frac{\partial \mathrm{E}_{\text{total}}}{\partial \mathrm{w}_1} &= \frac{\partial \mathrm{E}_{\text{total}}}{\partial \operatorname{sig}(o_1)} * \frac{\partial \operatorname{sig}(o_1)}{\partial \operatorname{net}(o_1)} * \frac{\partial \operatorname{net}(o_1)}{\partial \mathrm{w}_1} \\ \frac{\partial \mathrm{E}_{\text{total}}}{\partial \mathrm{w}_1} &= (\operatorname{sig}(o_1) - \operatorname{target}(o_1)) * \operatorname{sig}(o_1) * (1 - \operatorname{sig}(o_1)) * \mathrm{i}_1 \end{aligned} \tag{C.59}$$

Observe that the calculated w_1 affects the total error in the network.

The w_1 successfully linked the sigmoid value of neuron h_1 to neuron o_1. The new value of w_1, w_{1_1}, is now (w_{1_0} is the old value)

$$\mathrm{w}_{1_1} = \mathrm{w}_{1_0} - \eta \frac{\partial \mathrm{E}_{\text{total}}}{\partial \mathrm{w}_{1_0}} \tag{C.60}$$

The η is an "eta" signifying the learning rate, indicating how the weights update. It can be fixed or adaptively changed. The most popular method is called "Adam Optimizer", which is a method that adapts the learning rate. The larger the learning rate, the quicker the neural network will lessen the accuracy of the error to get close to the output. Though, the neural network will need to be more accurate.

For the calculation of ANN accuracy, it uses the formula:

$$Accuracy = \frac{numberofcorrectpredictions}{totalnumberofpredictions} \tag{C.61}$$

Cross-entrophy Loss is also known as loss function. It is a measure from the field of information theory, building upon entropy and generally calculating the difference between two probability distributions and the performance of a classification model whose output is a probability value between 0 and 1. It can be computed through the:

In binary classification, where the number of classes ($M = 2$)

$$L = -[y\log(z) + (1-y)\log(1-z) \tag{C.62}$$

If $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class (c) label per observation (o) and sum the result.

$$L = -\sum_{c=1}^{M} y_{(o,c)}\log(p_{(o,c)}) \tag{C.63}$$

where:
M - number of classes
y - binary indicator (0 or 1) if class label c is the correct classification for observation o
p - predicted probability observation o is of class c

C.4.4 Artificial Neural Network - Python Implementation

Given the Artificial Neural Network (ANN) concept above Section 2, it is the typical example of 2 hidden layers - Artificial Neural Network shown in Table C.2.

```
#Import packages
import os
import math
import pandas as pd
import numpy as np
```

Table C.2 ANN parameters.

PCA- SRP and ANN Parameters	**Values**
Learning rate (η)	0.1%
Epochs	100
Number of ANN neurons	2 hidden layers (32 and 16, respectively)
Activation functions used	ReLU in hidden layers SoftMax in final layer

```
from sklearn.decomposition import PCA
from sklearn import pre-processing
import matplotlib.pyplot as plt
import tensorflow as tf
optim = tf.keras. optimisers.Adam()
from pandas.plotting import scatter_matrix
from keras.layers import Dense
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
#from tensorflow.python.keras. optimisers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
accuracy_score
from sklearn. pre-processing import MinMaxScaler
from scipy.interpolate import UnivariateSpline
import timeit
```

Setting the time

```
start = time it.default_timer()
```

Setting ANN hyperparameters

```
target = 'target'
class_name = target
learning_rate = 0.1
epoch = 100
size_test = 0.2
```

Creating a data frame

```
data = pd.DataFrame()
```

Setting Data and File Directory

```
file_loc = "/content/gdrive/MyDrive/Colab Notebooks/
Neural Network Program Python/"
filename = "16 cancer patient"
other = " random"
```

Getting the CSV file

```
file_loc_input =  ( file_loc + filename +  ".csv")
data = pd.read_csv( file_loc_input)
data_old = data
```

ANN Part

The number of input columns and output classification

```
input_column = len( data_old.columns) - 1
output_class = len( data_old.groupby( class_name).size() )
```

We now have a data frame containing all of the data frame .csv data. However, we need to separate them into [X, Y].

Where our target labels are 'Y', and 'X' is our training data.

```
Y = data_old.target.values
X = data_old.drop([class_name],  axis = 1)
```

Now split to train/test

```
X_train,  X_test,  Y_train,  Y_test = train_test_split
( X,  Y,  test_size = size_test,  random_state = 42 )
```

Define a Neural Network Model

```
def NN_model( learning_rate ):
    model = Sequential()
    model.add(Dense( 32,  input_dim = input_column,
     kernel_initializer = 'normal', activation = 'relu'))
    model.add( Dense( 16,  kernel_initializer = 'normal',
     activation = 'relu'))
```

```
    model.add( Dense( output_class,  activation = 'softmax'))
    #Adam( lr = learning_rate)
    model.compile( loss = 'sparse_categorical_crossentropy',
     optimizer = 'Adam', metrics = ['accuracy'] )
    return model
```

Build an NN model, and start training

```
model = NN_model( learning_rate )
print(model.summary() )
history = model.fit( X_train,  Y_train,  validation_data =
( X_test,  Y_test),  epochs = epoch,  batch_size = 16,
verbose = 0)
```

Calculate the accuracy and classification report

```
predictions = np.argmax( model.predict( X_test ),  axis = 1 )
model_accuracy = accuracy_score( Y_test,  predictions )
* 100
print( "Model Accuracy:  ",  model_accuracy )
```

Appending the accuracy

```
ModAcc = ModAcc.append( { 'Model Accuracy' :
model_accuracy },  ignore_index = True )
```

Appendix D

Relevant Tables and Figures

Table D.1 Performance Metrics Table

		Predicted			
	Population **(P + N)**	**Positive** **(P)**	**Negative** **(N)**	**Informedness** **Bookmaker Informedness (BM)** **TPR +TNR -1**	**Prevalence Threshold** **(PrevT)** **(sqrt(TPR * FPR) - FPR)** **/ (TPR - FPR)**
Actual	**Positive** **(P)**	**True Positive** **(TP)**	**False Negative** **(FN)**	**True Positive Rate** **(TPR)** **Recall, Sensitivity** **TP / P**	**False Negative Rate** **(FNR)** **Miss Rate** **FN / P**
	Negative **(N)**	**False Positive** **(FP)**	**True Negative** **(TN)**	**False Positive Rate** **(FPR)** **Fall - out** **FP / N**	**True Negative Rate** **(TNR)** **Specificity** **TN / N**
	Prevalence **(Prev)** **P / (P + N)** **Accuracy** **(Acc)** **(TP + TN) / (P +N)** **Balanced Accuracy** **(BA)** **(TPR + TNR) / 2**	**Positive Predictive Value** **(PPV), Precision** **1 - FDR** **False Discovery Rate** **(FDR)** **1 - PPV** **F1 Score** **(F1)** **2 * TP /** **(2 * TP + FN + FP)**	**False Omission Rate** **(FOR)** **FN / PN** **Negative Predictive Value** **(NPV)** **TN / PN** **Fowles - Mallows Index** **(FM)** **sqrt(PPV * TPR)**	**Positive Likelihood Ratio** **(LRP, LR+)** **TPR / FPR** **Markedness (MK , δp)** **MK = PPV + NPV - 1** **Matthews Correlation** **Coefficient (MCC)** **sqrt(TPR * TNR * PPV * NPV) -** **sqrt(FNR * FPR * FOR * FDR)**	**Negative Likelihood Ratio** **(LRN, LR-)** **FNR / TNR** **Diagnostic Odds Ratio** **(DOR)** **LRP / LRN** **Threat Score** **Critical Success Index** **(TS, CSI)** **Jaccard index** **TP / (TP + FN + FP)**

D.0.1 Performance Metric Table Summary

D.0.2 Performance Metric Details

Table D.2 Performance Metrics Summary

Definition	Details	Formula
Error Rate	It represents the complement of accuracy. It could vary between 0 and 1. Being 0 is the best and one the worst. Error Rate is not suitable if the sample size is minimal and only focuses on false classification.	Error_rate = (FP + FN) / (P + N)
Balanced Accuracy	This metric is beneficial when the number of observations across classes is imbalanced. It is only used if there are two or more classes, normally if the classification has more target classes.	BA = (TPR + TNR) / 2
F-score	F1-score, F-measure is the harmonic mean of Precision and Recall. It is useful because as the Precision increases and Recall decreases, and vice versa. It can handle imbalanced data, and like ROC and kappa coefficient, it measures the classifier performance across different significance levels. It does not generally take into account true negatives. True negatives can change without affecting the F-measure.	F1 = 2 * TP / (2 * TP + FP + FN)
G-mean	The Geometric Mean (gmean) is a measure that considers a balance between the performance of both majority and minority classes. The higher the value, the lower the risk of over-fitting of negative and under-fitting of positive classes	Gmean = math.sqrt (TPR * TNR)
Matthews Correlation Coefficient	Also known as phi-coefficient. It is advantageous when the number of observations belonging to each class is uneven. It varies between 0-1, being 0 the worst and one the best. The mcc estimation is only available for binary cases (two classes).	MCC = ((TP * TN)- (FP * FN))/ math.sqrt ((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))

Fowlkes-Mallows Index	The fmi is a metric that measures the similarity between two clusters (predicted and observed). It is equivalent to the square root of the product between Precision (PPV) and Recall (TPR). It varies between 0-1, being 0 the worst and one the best.	FM = math.sqrt (PPV * TPR)
Informedness	Also known as the Bookmaker Informedness, or as the Youden's J-index. It is a suitable metric when the number of cases for each class is uneven. It varies between 0 to 1, and it is the distance between 1 and the sum of Specificity and Sensitivity.	BM = TPR + TNR - 1
Positive Likelihood Ratio	The posLr, also known as LR(+) represents the odds of obtaining a positive prediction for actual positives.	LRP = TPR / FPR
Negative Likelihood Ratio	The negLr, also known as LR(-) indicates the odds of obtaining a negative prediction for actual positives (or non-negatives in multiclass) relative to the probability of actual negatives of obtaining a negative prediction	LRN = FNR / TNR
Diagnostic Odds Ratio	The dor is a metric summarising the effectiveness of classification. It represents the odds of a positive case obtaining a positive prediction result concerning the odds of actual negatives obtaining a positive result	DOR = LRP / LRN
False Positive Rate	It represents the complement of specificity. It could vary between 0 and 1 - the lower the better.	FPR = FP / N
False Negative Rate	It represents the complement of recall. It could vary between 0 and 1 - the lower, the better.	FNR = FN / P
False Detection Rate	It represents the complement of Precision (or positive predictive value -PPV-). It could vary between 0 and 1, with 0 being the best and one the worst.	FDR = FP / (FP + TP)
False Omission Rate	It represents the complement of the npv. It could vary between 0 and 1, being 0 the best and one the worst	FOR = FN / (FN + TN)

Prevalence	It represents the complement of accuracy. It could vary between 0 and 1. Being 0 the best and 1 the worst	Prev = P / (P + N)
Prevalence Threshold	It represents the complement of accuracy. It could vary between 0 and 1. Being 0 the best and 1 the worst	PrevT = (Actual (TPR * FPR) - FPR) / (TPR - FPR)
Critical Success Index	The csi is also known as the threat score (TS). It could vary between 0 and 1, being 0 the worst and one the best	TS = TP /(TP + FN + FP)
Markedness or deltap	The delta is a metric that quantifies the probability that a condition is marked by the predictor for a random chance; it is the distance of the sum of Precision and NPV to 1	MK = PPV + NPV - 1

D.0.3 MM dataset - PCA-SRP Summary (Sc = PC_1 , n = 4927)

Table D.3 MM dataset - PCA-SRP Summary (Sc = PC_1 , n = 4927)

EEG channel	Sc	max loading score	threshold	n_PCA_samples	n_samples removed, %
FC5	0.926615	0.01479983716	0.01371375111	4389	10.91942358
FC3	0.9325801	0.01475247084	0.01375786073	4425	10.18875584
FC1	0.930692	0.01476741612	0.01374391604	4392	10.85853461
FCz	0.9332523	0.01474718112	0.0137628407	4391	10.87883093
FC2	0.9367157	0.01471984541	0.0137883103	4400	10.69616399
FC4	0.9352056	0.01473174079	0.01377720648	4403	10.63527502
FC6	0.932667	0.0147517493	0.01375846976	4391	10.87883093
C5	0.9148108	0.01489507106	0.01362617187	4309	12.54312969
C3	0.9156646	0.01488814111	0.01363254378	4336	11.99512888
C1	0.9203725	0.01484999314	0.01366752531	4340	11.91394358
Cz	0.9179583	0.0148695273	0.013649606	4331	12.09661051
C2	0.924493	0.01481687681	0.01369809889	4350	11.71098031
C4	0.9164246	0.01488195373	0.0136381885	4339	11.9342399
C6	0.9398372	0.01469542234	0.01381130458	4413	10.43231175
CP5	0.8906524	0.01509574065	0.01344505764	4223	14.28861376
CP3	0.8949061	0.01505980678	0.01347711295	4235	14.04505784
CP1	0.8973749	0.0150390474	0.01349566365	4248	13.7812056
CPz	0.8974655	0.01503830898	0.01349636349	4257	13.59853866
CP2	0.8982369	0.01503186685	0.01350217748	4257	13.59853866
CP4	0.9028034	0.01499379783	0.01353645166	4271	13.3143901
CP6	0.9166222	0.01488036235	0.01363967048	4304	12.64461133

Fp1	0.9420526	0.01467814222	0.01382758204	4492	8.828901969
Fpz	0.9560938	0.01456995216	0.01393024092	4541	7.834381977
Fp2	0.945395	0.01465217368	0.01385209174	4514	8.382382789
AF7	0.9398288	0.0146954948	0.01381124925	4478	9.113050538
AF3	0.9510294	0.01460868576	0.01389328966	4505	8.565049726
AFz	0.9588517	0.01454899405	0.01395032767	4530	8.057641567
AF4	0.9419454	0.01467896609	0.01382678459	4479	9.092754211
AF8	0.9380152	0.01470969595	0.01379791838	4465	9.376902781
F7	0.954474	0.01458228957	0.01391841625	4533	7.996752588
F5	0.9395444	0.01469771194	0.01380915295	4490	8.869494621
F3	0.94864	0.01462707605	0.01387582942	4472	9.234828496
F1	0.9467427	0.01464170568	0.01386192797	4485	8.970976253
Fz	0.9462075	0.01464585327	0.01385801621	4473	9.21453217
F2	0.9491149	0.01462339808	0.01387928501	4476	9.153643191
F4	0.9446887	0.01465764699	0.01384691348	4449	9.701644002
F6	0.9351537	0.01473215417	0.01377682848	4425	10.18875584
F8	0.9516677	0.01460378363	0.01389794917	4468	9.316013802
FT7	0.9341631	0.0147399822	0.01376954747	4408	10.53379338
FT8	0.9251443	0.01481166001	0.01370292283	4377	11.1629795
T7	0.8816652	0.0151722951	0.01337688459	4197	14.81631825
T8	0.9051667	0.01497420155	0.01355414861	4288	12.96935255
T9	0.8913071	0.01509015885	0.01344996572	4190	14.95839253
T10	0.9543023	0.01458363183	0.01391719339	4466	9.356606454
TP7	0.864828	0.01531945819	0.01324869639	4115	16.48061701

TP8	0.8907132	0.01509517675	0.01344547319	4235	14.04505784
P7	0.8496028	0.01545612411	0.01313156632	4079	17.21128476
P5	0.8626011	0.01533918467	0.01323159757	4136	16.05439415
P3	0.8693408	0.01527960771	0.01328318639	4145	15.87172722
P1	0.8677898	0.01529325703	0.01327133246	4138	16.0138015
Pz	0.8734056	0.0152440557	0.01331424362	4165	15.46580069
P2	0.8739702	0.01523913104	0.0133185464	4158	15.60787497
P4	0.8750667	0.01522956602	0.01332688608	4177	15.22224477
P6	0.8827496	0.01516315288	0.01338526714	4198	14.79602192
P8	0.8853872	0.01514044624	0.01340515731	4202	14.71483661
PO7	0.8301149	0.01563645721	0.01298005612	4016	18.48995332
PO3	0.8414308	0.01553095821	0.01306822659	4046	17.88106353
POz	0.8538675	0.01541744688	0.01316445682	4085	17.0895068
PO4	0.8485243	0.01546594659	0.0131232315	4075	17.29247006
PO8	0.8289752	0.01564722604	0.01297116233	3977	19.28151005
O1	0.8096398	0.01583285791	0.01281891191	3956	19.7077329
Oz	0.8318901	0.01561975614	0.0129939205	4019	18.42906434
O2	0.8336977	0.0156026869	0.01300792418	4017	18.46965699
Iz	0.8123043	0.01580698255	0.0128400799	3950	19.82951086