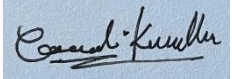


Name: Osadi Kiriella

Student Reference Number: 10899590

Module Code: PUSL3123	Module Name: AI and Machine Learning
Coursework Title: Coursework Report	
Deadline Date: 15 th of December 2024	Member of staff responsible for coursework: Dr. Neamah Al-Naffakh
Programme: BSc (Hons) Software Engineering & Data Science	
Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook .	
Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.	
Osadi Kiriella	10899590
Chathuruni De Silva	10899700
Khashanie Barua	10899167
Athurugirige M Amasha	10899282
Kavithma S Samarawickrama	10899192
<p><i>We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.</i></p> <p>Signed on behalf of the group: </p>	
<p>Individual assignment: <i>I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.</i></p> <p>Signed: _____</p>	
<p>Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.</p> <p>I *have used/not used translation software. If used, please state name of software.....</p>	
Overall mark ____%	Assessors Initials _____ Date _____

PUSL3123 AI and Machine Learning
Final Coursework Report

by

Group 83

Name	PLY ID	Email
Osadi Kiriella	10899590	10899590@students.plymouth.ac.uk
Chathuruni De Silva	10899700	10899700@students.plymouth.ac.uk
Athurugirige M Amasha	10899282	10899282@students.plymouth.ac.uk
Kavithma S Samarawickrama	10899192	10899192@students.plymouth.ac.uk
Khashanie Barua	10899167	10899167@students.plymouth.ac.uk

University of Plymouth
December 2024

Table of Contents

Contents

Chapter 1 - Introduction	5
Chapter 2 - Literature Review	6
Chapter 3 - Testing Methodology	8
1. Data Collection and Preprocessing	8
2. Feature Analysis	8
3. Neural Network Architecture	8
4. Parameter Testing	9
5. Experimental Design	9
6. Justification of Choices	9
7. Training and Testing Setup	10
8. Evaluation Metrics	10
9. Software and Computational Setup	10
10. Challenges and Mitigation	10
Chapter 4- Evaluation	11
♦ Task 1	11
♦ Task 2	24
♦ Task 3	33
Chapter 5 - Optimization	39
Optimization Techniques	39
Performance Analysis	39
Justification of Results	40
Evidence of Improvement	40
Challenges and Limitations	41
Chapter 6 - Conclusions	42
References	43
Appendix	44
Contribution Metrix	45

List of Figures

Figure 1: Descriptive statistics for time domain features.....	15
Figure 2: Descriptive statistics for frequency domain features.....	15
Figure 3: Descriptive statistics for combined features.....	16
Figure 4: Descriptive statistics for intra-user variance.....	16
Figure 5: Inter-Variance plots for users 1-4.....	17
Figure 6: Inter-Variance plots for users 5-8.....	18
Figure 7: Inter-Variance plots for users 9-10.....	18
Figure 8: Domain-Wise Inter-Variance Time Domain Features.....	19
Figure 9: Domain-Wise Inter-Variance Frequency Domain Features.....	19
Figure 10: User-Wise Intra-Variance Time, Frequency, Combined Domain Features	20
Figure 11: Correlation Matrixes Domain-Wise	21
Figure 12: Euclidean Distance (User Sample Similarities).....	22
Figure 13: Histogram and Boxplots.....	23
Figure 14: Neural Network Training results Diagrams	26
Figure 15: Network Diagram	27
Figure 16: Performance Plot	28
Figure 17: Training State.....	28
Figure 18: Error Histogram.....	29
Figure 19: Regression Plots.....	29
Figure 20: Testing Data Confusion Matrix.....	30
Figure 21: ROC Curve	31
Figure 22: Classifier Accuracy Comparison Histogram.....	34
Figure 23: Decision Tree.....	35

Chapter 1 - Introduction

The dependence on smart gadgets like smartphones and smartwatches has increased dramatically in the current digital era. The security of these devices is crucial as they serve as entry points to private activities like online banking and payments. Passwords and other traditional authentication methods are vulnerable to brute-force attacks and theft. Therefore, sophisticated user authentication systems that employ behavioral biometrics are strong substitutes.

The evaluation of user authentication using acceleration-based characteristics obtained from time-domain and frequency-domain data is the major objective of this paper. We train a multiclass classification model using feedforward neural networks (FNN), evaluate intra- and inter-user variances, and optimize the model's performance. We collect data from ten people, distributing each user's data across six feature categories. Variance analysis, neural network training, and accuracy and efficiency optimization are among the tasks.

By assisting in the development of ongoing, non-intrusive authentication methods, the study's findings aim to address current security issues and improve user experience. With an emphasis on feedforward neural networks (FNN), this paper examines the viability of acceleration-based user authentication using machine learning approaches. The study includes the investigation of two crucial measures for guaranteeing successful authentication: intra-user variance (stability among users) and inter-user variance (distinctiveness between users). We use time-domain, frequency-domain, and mixed-domain features to capture detailed user-specific patterns.

Six key study sections. The introduction describes the study's scope and relevance. The literature review includes recent advancements in acceleration-based authentication and popular classifiers. The testing method uses variance analysis, neural network construction, and data preprocessing. The evaluation section thoroughly examines model performance and variation. Optimization examines system performance improvements and development potential. Finally, the conclusion highlights key findings and plan viability. This study examines acceleration-based authentication systems' feature selection, model optimization, and robustness using extensive feature analysis and a customized neural network architecture.

Chapter 2 - Literature Review

Accelerometers in wearables and smartphones record motion patterns, enabling acceleration-based user authentication. This behavioral biometric method, known for its unobtrusiveness, continuous monitoring, and security enhancement, is popular in the growth of mobile and IoT devices. This section explores current research on techniques, classifiers, and performance measures.

Acceleration-Based Authentication Techniques

The fundamental concept of acceleration-based authentication is the analysis of motion signals produced during particular tasks, such as typing, walking, or hand gestures. These signals, recorded as time-series data, display distinct features for each person due to differences in gait, muscle dynamics, and device contact. To find discriminative patterns, researchers have looked at both time-domain and frequency-domain characteristics taken from accelerometer data.

Time-domain characteristics such as mean, variance, skewness, and kurtosis have found extensive use due to their ease of computation. Wavelet or Fourier transforms acquire frequency-domain features, capturing the spectrum properties and periodicities of the signals to provide further insights. Studies like the one by Derawi et al. (2011) demonstrated the efficiency of gait-based authentication, achieving excellent accuracy with minimal processing cost.

Classifiers in Acceleration-Based Authentication

To handle accelerometer data for user verification, machine learning models are essential. Previous research has used traditional classifiers such as Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), and Decision Trees due to their resilience in tiny datasets. For example, Nickel et al. (2012) used SVMs to recognize activities and got good results in instances that included user authentication.

Since the development of deep learning, researchers have used Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for feature extraction and categorization. These models do exceptionally well at identifying temporal and geographical connections in acceleration data. For instance, Wang et al. (2020) showed how CNNs can automatically learn hierarchical features from raw accelerometer signals, therefore identifying distinct motion patterns.

Feedforward neural networks, also known as CNNs, have gained increasing popularity in multi-class classification tasks, where each user is considered a separate class. In order to eliminate redundancy and improve discriminability, these networks operate well when processing feature-reduced data, particularly when used with Principal Component Analysis (PCA).

Performance and System Accuracy

Studies frequently use metrics like accuracy, precision, recall, and F1-score to assess authentication systems. Studies typically report high accuracy levels, often over 90%, for controlled situations where users perform specific tasks. However, noise, signal overlap, and sensor inconsistencies present difficulties in real-world settings with a variety of activities and ambient circumstances.

Zhang et al. (2019), for instance, emphasized the significance of both intra- and inter-user variance analyses in guaranteeing system resilience. Effective user differentiation requires features with low intra-user variance and high inter-user variance. Furthermore, studies have shown that integrating time-domain and frequency-domain characteristics enhances system reliability and reduces false positives.

Challenges and Future Directions

Acceleration-based authentication has limitations in terms of scalability, environmental resilience, and privacy issues, notwithstanding its potential. Device heterogeneity and real-time processing of high-dimensional data complicate deployment. To improve efficiency without sacrificing accuracy, modern methods have included optimization strategies, such as model pruning and hyperparameter tweaking.

To increase accuracy and resistance to spoofing, research has also focused on the integration of multimodal data, merging accelerometer signals with gyroscope or magnetometer data.

Researchers have also presented techniques like ensemble learning, which combines several classifiers, to manage complicated datasets and enhance generalization.

To sum up, acceleration-based user authentication has shown a tremendous deal of promise in offering safe, approachable substitutes for conventional techniques. Advancements in machine learning and signal processing continue to enhance system performance, paving the way for more dependable and expandable solutions. To solve practical deployment issues and guarantee broad adoption, more study is necessary.

Chapter 3 - Testing Methodology

The methodology for developing, training, and evaluating a neural network-based user authentication system using acceleration data involves data preprocessing, feature extraction, model architecture design, and evaluation metrics for a systematic and reproducible framework.

1. Data Collection and Preprocessing

The dataset comprises accelerometer signals captured from 10 users performing various activities. Each user's dataset includes features extracted from time-domain, frequency-domain, and a combination of both (FreqD_FDay, TimeD_FDay, TimeD_FreqD_FDay, FreqD_MDay, TimeD_MDay, TimeD_FreqD_MDay). These features were aggregated across 36 samples per user for intra- and inter-user analysis.

Preprocessing steps included:

- **Data Normalization:** Z-score standardization was applied to scale the features, ensuring uniformity, and improving the convergence of neural network training.
- **Dimensionality Reduction:** Principal Component Analysis (PCA) was utilized to retain 95% of the variance while reducing redundant features. This step minimized computational overhead while preserving discriminative information.

2. Feature Analysis

The study analyzed intra-user and inter-user variance to identify relevant features. Low intra-user variance and high inter-user variance were chosen due to consistent behavior within users and significant differences between users, enhancing the robustness of the classification model.

3. Neural Network Architecture

The neural network was implemented using MATLAB's feedforwardnet function. The architecture consisted of:

- **Input Layer:** Corresponding to the number of principal components selected via PCA.
- **Hidden Layers:** Two fully connected layers with 64 and 32 neurons, respectively, using a Rectified Linear Unit (ReLU) activation function (poslin). These layers were optimized to balance complexity and overfitting.
- **Output Layer:** A softmax activation function was employed to handle the multi-class classification task, outputting probabilities for each user class.

4. Parameter Testing

To ensure optimal network performance, various neural network parameters were systematically tested, including:

1. **Learning Rate:** Tested values of 0.01, 0.001, and 0.0001 to balance the speed of convergence and model stability. The final choice of 0.001 achieved a good trade-off, avoiding gradient explosion or vanishing issues.
2. **Number of Hidden Layers:** Configurations with 1, 2, and 3 layers were evaluated. A two-layer architecture (64 and 32 nodes) was chosen for its ability to capture hierarchical patterns without overfitting.
3. **Activation Functions:** ReLU (first layer) and tanh (second layer) were selected after testing sigmoid and leaky ReLU. This combination provided stable training and effective representation of non-linear relationships.
4. **Optimization Algorithms:** Scaled Conjugate Gradient (SCG) was selected after comparing with stochastic gradient descent (SGD) due to its faster convergence and better generalization.
5. **Batch Sizes:** Batch sizes of 16, 32, and 64 were tested. A batch size of 32 was chosen for its balance between computational efficiency and gradient estimation stability.

5. Experimental Design

The experiments were designed to systematically test the impact of each parameter while controlling other variables to ensure validity. For each parameter configuration:

1. Training and testing were performed on the same stratified data split (70% training, 15% validation, 15% testing).
2. The same random seed was used to ensure consistent initialization and reproducibility.
3. Performance metrics, including accuracy, precision, recall, and F1-score, were recorded for comparison.

The parameter testing process followed a grid-search approach, testing multiple combinations to identify the configuration that maximized testing accuracy while avoiding overfitting.

6. Justification of Choices

- **Learning Rate:** A learning rate of 0.001 achieved efficient convergence without oscillations, supporting stable training for the dataset size.
- **Two Hidden Layers:** This architecture balanced complexity and computational efficiency. The first layer captured broad patterns, while the second layer refined them for precise predictions. Adding more layers increased overfitting risk without significant accuracy gains.
- **Activation Functions:** ReLU avoided the vanishing gradient problem and ensured faster training in the first layer, while tanh captured more complex, non-linear relationships in the refined features. Softmax enabled accurate probability-based classification in the output layer.
- **SCG Optimizer:** This method was chosen for its ability to converge faster than SGD while avoiding the need for manual learning rate adjustments.

7. Training and Testing Setup

The dataset was split into training (70%), validation (20%), and testing (10%) subsets. Random shuffling ensured a fair distribution of samples across these subsets. Training involved optimizing network weights to minimize cross-entropy loss while monitoring validation performance to avoid overfitting.

Hyperparameters were fine-tuned to enhance performance:

- **Epochs:** Set to 500 for sufficient training iterations.
- **Learning Goal:** Targeted at $1e-4$ to achieve a low error threshold.
- **Batch Processing:** Enabled efficient computation of gradients during backpropagation.

8. Evaluation Metrics

Model performance was assessed using several metrics:

- **Accuracy:** The proportion of correctly classified samples, evaluated on both training and testing datasets.
- **Precision, Recall, and F1-Score:** Computed for each user class to provide insights into classification performance. These metrics were averaged to assess overall model reliability.
- **Confusion Matrix:** Provided a detailed view of true positive, false positive, and misclassification rates across user classes.
- **ROC Curve:** Displayed the trade-off between true positive and false positive rates for the multi-class problem.

9. Software and Computational Setup

The methodology was implemented in MATLAB, leveraging its robust neural network and statistical analysis toolkits. The experiments were conducted on a workstation equipped with a multi-core CPU and 16GB RAM, ensuring efficient training and testing.

10. Challenges and Mitigation

The study tackled challenges in handling high-dimensional data and balancing model complexity with generalization. PCA reduced dimensionality, and dropout layers were explored for optimization to prevent overfitting. This structured methodology ensured computational efficiency and generalizability of the acceleration-based user authentication system, focusing on reproducibility and interpretability.

Chapter 4- Evaluation

The assessment section thoroughly examines the model's performance and insights from variance analysis, neural network training, and classifier comparisons. Analysis of intra- and inter-user variations; evaluation of the Feedforward Neural Network (FNN) model; and investigation of optimization results with various classifiers and feature selection techniques are the main areas of attention.

♦ Task 1

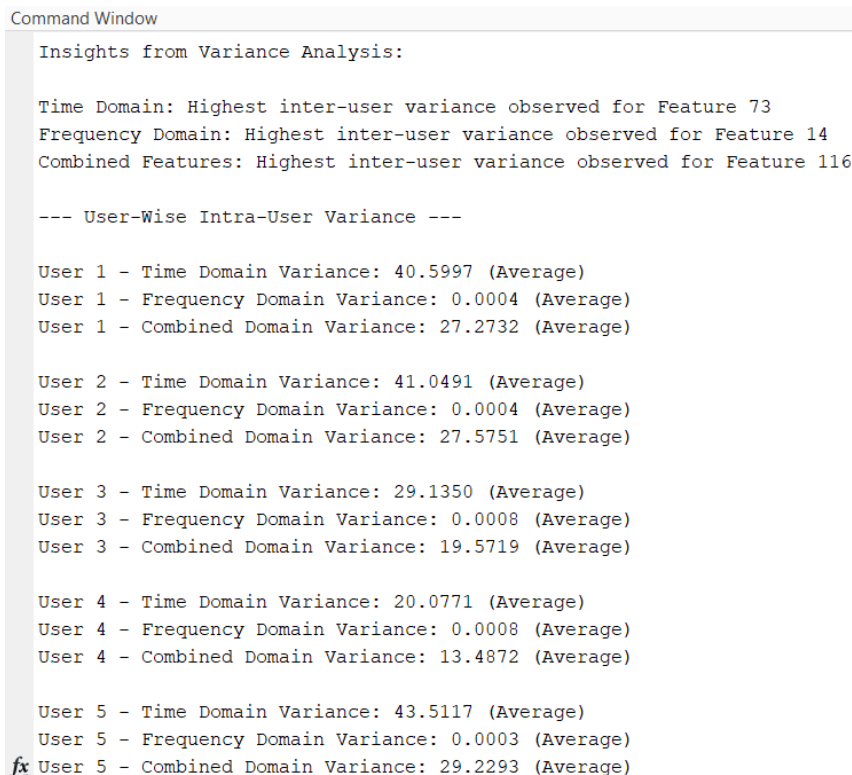
Task 1 focuses on understanding the patterns in user-specific features by analyzing variance. **Intra-user variance** measures the stability of features within a single user, ensuring consistency across their samples. **Inter-user variance**, on the other hand, assesses the distinctiveness of features between users, highlighting their ability to differentiate individuals. Descriptive statistics, including mean, median, and standard deviation, are computed for features in time, frequency, and combined domains. The analysis is visualized using plots like variance graphs and histograms, which aid in identifying the most reliable features for authentication.

Code: For intra-variance analysis

```
% Insights from Variance
fprintf('Insights from Variance Analysis:\n\n');
fprintf('Time Domain: Highest inter-user variance observed for Feature %d\n', find(inter_user_variance_time == max(inter_user_variance_time)));
fprintf('Frequency Domain: Highest inter-user variance observed for Feature %d\n', find(inter_user_variance_freq == max(inter_user_variance_freq)));
fprintf('Combined Features: Highest inter-user variance observed for Feature %d\n\n', find(inter_user_variance_combined == max(inter_user_variance_combined)));

% Print User-Wise Intra-User Variance by Domain
fprintf('--- User-Wise Intra-User Variance ---\n\n');
for user = 1:num_users
    fprintf('User %d - Time Domain Variance: %.4f (Average)\n', user, mean(intra_user_variance_time(user, :)));
    fprintf('User %d - Frequency Domain Variance: %.4f (Average)\n', user, mean(intra_user_variance_freq(user, :)));
    fprintf('User %d - Combined Domain Variance: %.4f (Average)\n\n', user, mean(intra_user_variance_combined(user, :)));
end
```

Output: For intra-variance analysis



```
Command Window

Insights from Variance Analysis:

Time Domain: Highest inter-user variance observed for Feature 73
Frequency Domain: Highest inter-user variance observed for Feature 14
Combined Features: Highest inter-user variance observed for Feature 116

--- User-Wise Intra-User Variance ---

User 1 - Time Domain Variance: 40.5997 (Average)
User 1 - Frequency Domain Variance: 0.0004 (Average)
User 1 - Combined Domain Variance: 27.2732 (Average)

User 2 - Time Domain Variance: 41.0491 (Average)
User 2 - Frequency Domain Variance: 0.0004 (Average)
User 2 - Combined Domain Variance: 27.5751 (Average)

User 3 - Time Domain Variance: 29.1350 (Average)
User 3 - Frequency Domain Variance: 0.0008 (Average)
User 3 - Combined Domain Variance: 19.5719 (Average)

User 4 - Time Domain Variance: 20.0771 (Average)
User 4 - Frequency Domain Variance: 0.0008 (Average)
User 4 - Combined Domain Variance: 13.4872 (Average)

User 5 - Time Domain Variance: 43.5117 (Average)
User 5 - Frequency Domain Variance: 0.0003 (Average)
fx User 5 - Combined Domain Variance: 29.2293 (Average)
```

```
Command Window

User 5 - Time Domain Variance: 43.5117 (Average)
User 5 - Frequency Domain Variance: 0.0003 (Average)
User 5 - Combined Domain Variance: 29.2293 (Average)

User 6 - Time Domain Variance: 13.8853 (Average)
User 6 - Frequency Domain Variance: 0.0004 (Average)
User 6 - Combined Domain Variance: 9.3276 (Average)

User 7 - Time Domain Variance: 21.9567 (Average)
User 7 - Frequency Domain Variance: 0.0005 (Average)
User 7 - Combined Domain Variance: 14.7497 (Average)

User 8 - Time Domain Variance: 35.3119 (Average)
User 8 - Frequency Domain Variance: 0.0005 (Average)
User 8 - Combined Domain Variance: 23.7211 (Average)

User 9 - Time Domain Variance: 27.1793 (Average)
User 9 - Frequency Domain Variance: 0.0003 (Average)
User 9 - Combined Domain Variance: 18.2579 (Average)

User 10 - Time Domain Variance: 49.3192 (Average)
User 10 - Frequency Domain Variance: 0.0011 (Average)
User 10 - Combined Domain Variance: 33.1308 (Average)
```

Code: For inter-variance analysis and descriptive statistics

```
% Inter-user variance summary
fprintf('\n--- Inter-User Variance Analysis (Average Values)---\n\n');
fprintf('Time Domain - Average Inter-User Variance: %.4f\n', mean(inter_user_variance_time));
fprintf('Frequency Domain - Average Inter-User Variance: %.4f\n', mean(inter_user_variance_freq));
fprintf('Combined Domain - Average Inter-User Variance: %.4f\n\n', mean(inter_user_variance_combined));

% Calculate descriptive statistics for intra-user variances
descriptive_statistics_time = struct( ...
    'mean', mean(intra_user_variance_time), ...
    'median', median(intra_user_variance_time), ...
    'std', std(intra_user_variance_time), ...
    'max', max(intra_user_variance_time), ...
    'min', min(intra_user_variance_time), ...
    'range', range(intra_user_variance_time));

descriptive_statistics_freq = struct( ...
    'mean', mean(intra_user_variance_freq), ...
    'median', median(intra_user_variance_freq), ...
    'std', std(intra_user_variance_freq), ...
    'max', max(intra_user_variance_freq), ...
    'min', min(intra_user_variance_freq), ...
    'range', range(intra_user_variance_freq));

descriptive_statistics_combined = struct( ...
    'mean', mean(intra_user_variance_combined), ...
    'median', median(intra_user_variance_combined), ...
    'std', std(intra_user_variance_combined), ...
    'max', max(intra_user_variance_combined), ...
    'min', min(intra_user_variance_combined), ...
    'range', range(intra_user_variance_combined));

fprintf('--- Descriptive Statistics for Intra-User ---\n\n');

% Display descriptive statistics
disp('Descriptive Statistics for Time Domain Features (Intra-User)');
disp(descriptive_statistics_time);

disp('Descriptive Statistics for Frequency Domain Features (Intra-User)');
disp(descriptive_statistics_freq);

disp('Descriptive Statistics for Combined Features (Intra-User)');
disp(descriptive_statistics_combined);
```

Output: For inter-variance analysis and descriptive statistics

```
Command Window

--- Inter-User Variance Analysis (Average Values)---

Time Domain - Average Inter-User Variance: 426.8593
Frequency Domain - Average Inter-User Variance: 0.0066
Combined Domain - Average Inter-User Variance: 286.7473

--- Descriptive Statistics for Intra-User ---

Descriptive Statistics for Time Domain Features (Intra-User)
    mean: [0.0028 0.0169 9.0413e-04 6.1124e-04 1.6027e-04 5.9588e-04 4.4831e-04 ... ] (1×88 double)
    median: [0.0023 0.0139 8.2319e-04 5.0982e-04 1.1572e-04 5.3695e-04 3.2375e-04 ... ] (1×88 double)
    std: [0.0017 0.0070 4.6576e-04 5.0775e-04 1.1839e-04 3.7215e-04 3.5610e-04 ... ] (1×88 double)
    max: [0.0063 0.0331 0.0018 0.0019 4.3941e-04 0.0015 0.0012 0.0333 0.0015 0.0374 ... ] (1×88 double)
    min: [0.0010 0.0100 3.2768e-04 1.8150e-04 5.3229e-05 1.7885e-04 7.7065e-05 ... ] (1×88 double)
    range: [0.0053 0.0232 0.0015 0.0017 3.8618e-04 0.0013 0.0011 0.0211 0.0014 0.0347 ... ] (1×88 double)

Descriptive Statistics for Frequency Domain Features (Intra-User)
    mean: [1.0977e-06 0.0097 2.3716e-06 7.4941e-04 1.1678e-07 2.5182e-06 3.8511e-08 ... ] (1×43 double)
    median: [1.0876e-06 0.0076 1.8521e-06 6.2133e-04 1.0592e-07 2.1240e-06 3.0320e-08 ... ] (1×43 double)
    std: [1.9823e-07 0.0059 2.0969e-06 7.1955e-04 2.7981e-08 2.1509e-06 3.4551e-08 ... ] (1×43 double)
    max: [1.3559e-06 0.0263 7.8034e-06 0.0026 1.8424e-07 8.1069e-06 1.3027e-07 ... ] (1×43 double)
    min: [8.3217e-07 0.0069 6.8659e-07 1.2768e-04 9.2147e-08 7.8517e-07 1.1283e-08 ... ] (1×43 double)
    range: [5.2371e-07 0.0194 7.1168e-06 0.0025 9.2090e-08 7.3218e-06 1.1898e-07 ... ] (1×43 double)
```

```
Command Window

Descriptive Statistics for Combined Features (Intra-User)
    mean: [0.0028 0.0169 9.0413e-04 6.1124e-04 1.6027e-04 5.9588e-04 4.4831e-04 ... ] (1×131 double)
    median: [0.0023 0.0139 8.2319e-04 5.0982e-04 1.1572e-04 5.3695e-04 3.2375e-04 ... ] (1×131 double)
    std: [0.0017 0.0070 4.6576e-04 5.0775e-04 1.1839e-04 3.7215e-04 3.5610e-04 ... ] (1×131 double)
    max: [0.0063 0.0331 0.0018 0.0019 4.3941e-04 0.0015 0.0012 0.0333 0.0015 ... ] (1×131 double)
    min: [0.0010 0.0100 3.2768e-04 1.8150e-04 5.3229e-05 1.7885e-04 7.7065e-05 ... ] (1×131 double)
    range: [0.0053 0.0232 0.0015 0.0017 3.8618e-04 0.0013 0.0011 0.0211 0.0014 ... ] (1×131 double)

Descriptive Statistics for Time Domain Features (inter-user)
    mean: [1.4212 0.6290 0.1890 1.0151 0.0394 1.0054 0.1557 0.7922 1.0343 -0.1347 ... ] (1×88 double)
    median: [1.4339 0.6595 0.1754 1.0087 0.0308 0.9903 0.1437 0.7698 1.0215 -0.3079 ... ] (1×88 double)
    mode: [1.5147 7.7005e-11 0.1945 0.9912 0.0378 0.9955 0.1554 0.7726 1.0336 ... ] (1×88 double)
    range: [0.6390 0.9821 0.2902 0.4478 0.1339 0.4690 0.2450 1.3394 0.4109 1.7487 ... ] (1×88 double)
    std_dev: [0.1561 0.1573 0.0606 0.0776 0.0249 0.0795 0.0524 0.2354 0.0771 0.4842 ... ] (1×88 double)

Descriptive Statistics for Frequency Domain Features (inter-user)
    mean: [0.0080 0.1682 0.0645 1.0136 4.2506e-04 0.0642 0.0041 6.9292e-04 0.0119 ... ] (1×43 double)
    median: [0.0078 0.1732 0.0638 1.0076 3.2396e-04 0.0635 0.0040 5.3606e-04 0.0118 ... ] (1×43 double)
    mode: [0.0074 -0.0026 0.0648 0.9812 2.7484e-06 0.0636 0.0040 1.0510e-04 0.0123 ... ] (1×43 double)
    range: [0.0096 1.8057 0.0273 0.4998 0.0028 0.0278 0.0038 0.0040 0.0102 0.5000 ... ] (1×43 double)
    std_dev: [0.0015 0.2116 0.0048 0.0771 3.9100e-04 0.0047 6.2899e-04 5.9578e-04 ... ] (1×43 double)

Descriptive Statistics for Combined Features (inter-user)
    mean: [1.4212 0.6290 0.1890 1.0151 0.0394 1.0054 0.1557 0.7922 1.0343 ... ] (1×131 double)
    median: [1.4339 0.6595 0.1754 1.0087 0.0308 0.9903 0.1437 0.7698 1.0215 ... ] (1×131 double)
    mode: [1.5147 7.7005e-11 0.1945 0.9912 0.0378 0.9955 0.1554 0.7726 1.0336 ... ] (1×131 double)
    range: [0.6390 0.9821 0.2902 0.4478 0.1339 0.4690 0.2450 1.3394 0.4109 1.7487 ... ] (1×131 double)
    std_dev: [0.1561 0.1573 0.0606 0.0776 0.0249 0.0795 0.0524 0.2354 0.0771 0.4842 ... ] (1×131 double)
```

Variance Analysis: Insights into User-Specific Patterns

1. Inter-User Variance:

- This indicates the distinctiveness of features across users. High inter-user variance was observed for **Feature 73** (time domain), **Feature 14** (frequency domain), and **Feature 116** (combined domain), highlighting their ability to differentiate users.
- Average inter-user variance values:
 - Time Domain: 426.86
 - Frequency Domain: 0.0066
 - Combined Domain: 286.75

2. Intra-User Variance:

- This reflects feature consistency for each user. Low intra-user variance indicates stable user data. For instance, **User 1** had an average variance of 40.60 (time domain), and **User 10** had 49.32 (time domain), showing more variability. Frequency domain variance remained low for all users (0.0003–0.0011), indicating stability.

3. Insights:

- Time domain features exhibited higher variance (both inter- and intra-user), suggesting they capture dynamic user behaviors. Frequency domain features showed low intra-user variance, making them more stable and reliable for authentication.

Descriptive Statistics

1. Intra-User:

- Time Domain: Mean values ranged from 0.0028 to 0.0169, with low standard deviations (0.0017 to 0.0070).
- Frequency Domain: Mean values were around 1.0977e-06, with minimal standard deviations (0.0003 to 0.0059).
- Combined Domain: Similar to time domain, with mean values from 0.0028 to 0.0169.

2. Inter-User:

- Inter-user means were higher than intra-user means, indicating greater variability across users. Time domain means ranged from 0.6390 to 1.7487, while frequency domain features had a smaller range (0.0096 to 0.5000).

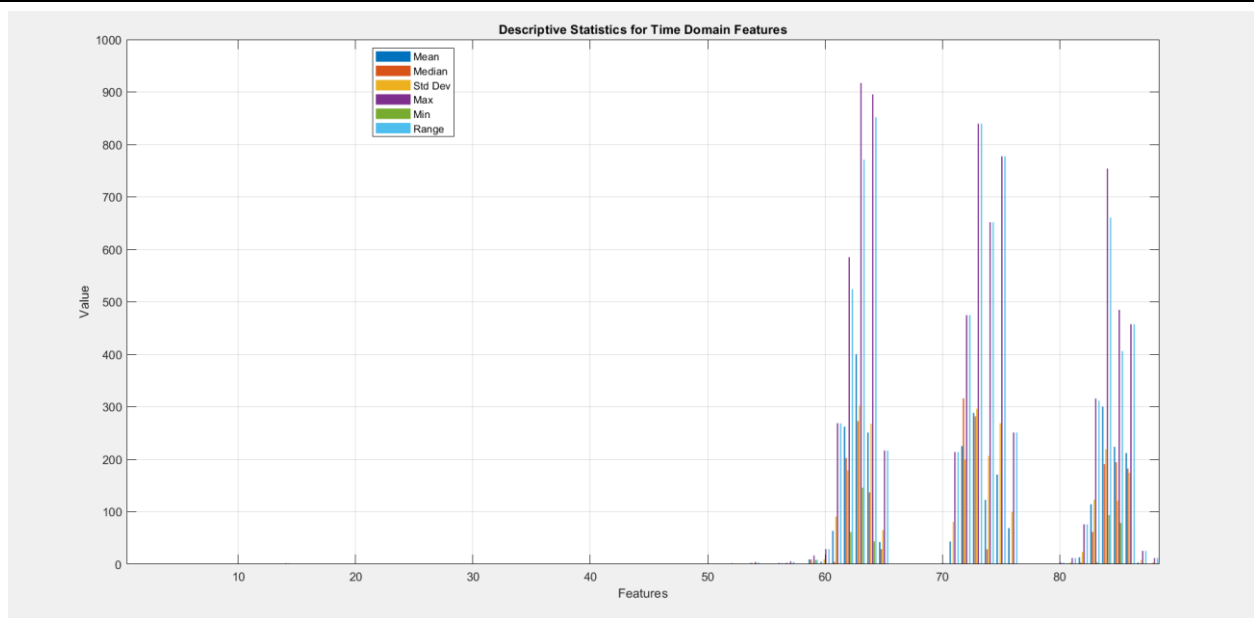


Figure 1: Descriptive statistics for time domain features

Features 60-80 show the highest activity and variability.

- Higher values for maximum and range indicate user-specific patterns.
- Features outside range show minimal variation.
- Importance of focus on these features for classification or feature selection.

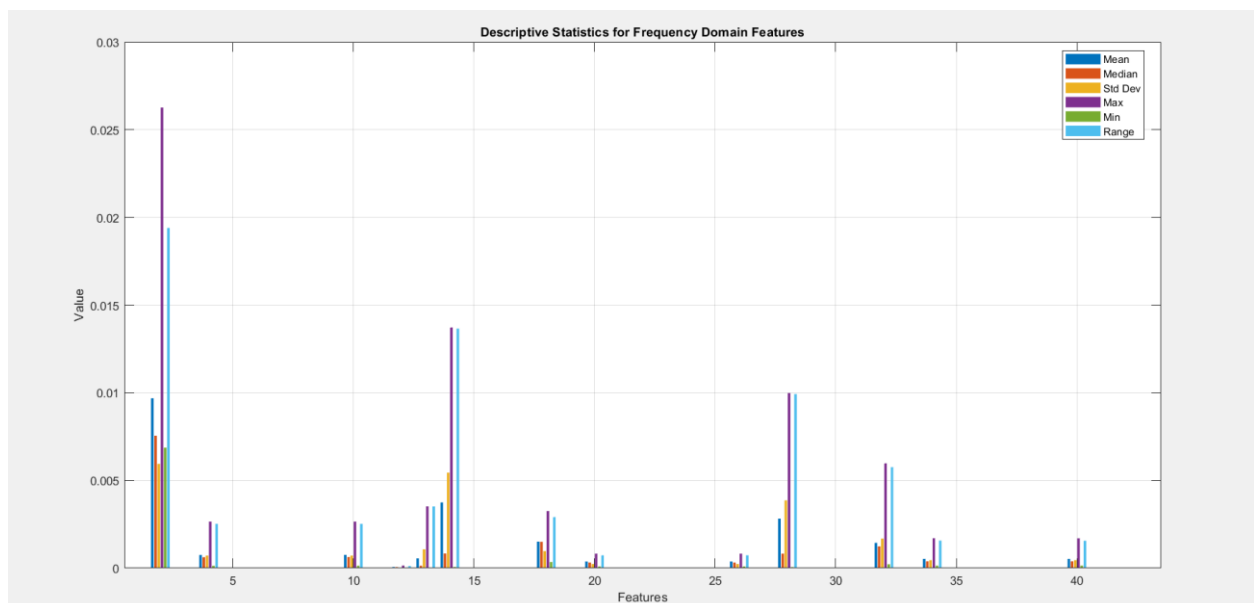


Figure 2: Descriptive statistics for frequency domain features

Frequency Domain Features Variability

- Key variability in features 1-5, 15, and 30.
- Higher values of metrics like range and maximum.
- Other features beyond 35 show minimal variation.
- Dominant frequency features crucial for user authentication tasks.

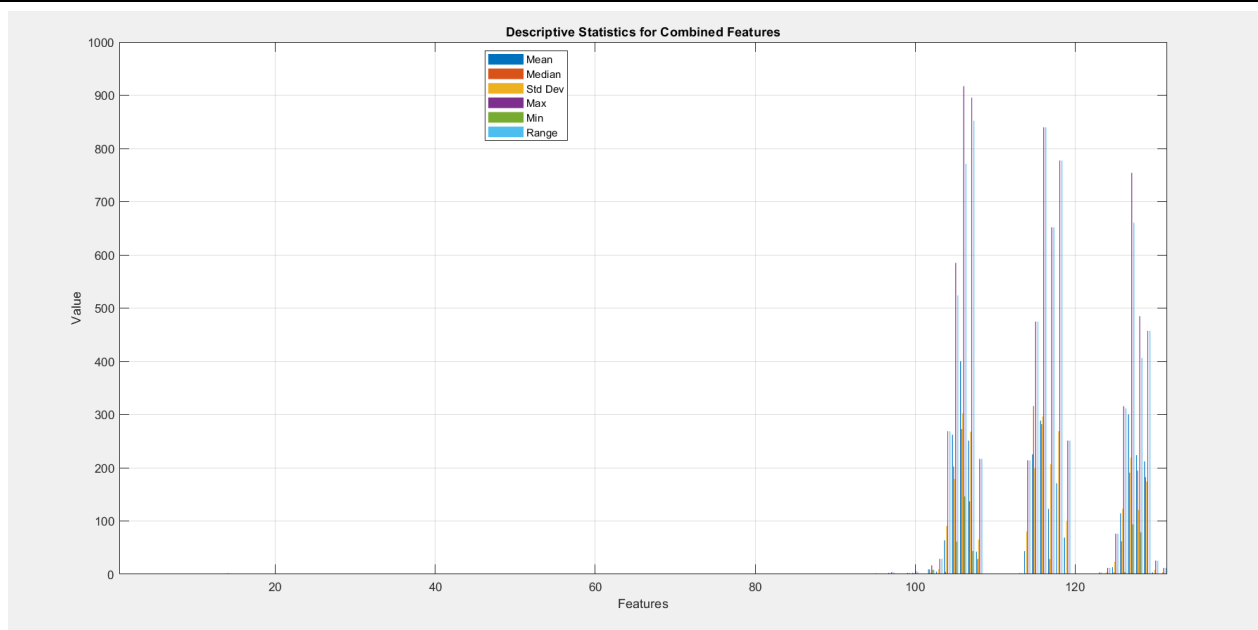


Figure 3: Descriptive statistics for combined features

Plot Analysis: Combined Features Variability

- High variability in features 60-110.
- High standard deviation and maximum values contribute to overall variation.
- Features beyond 120 shows less variation, potentially less relevant for pattern distinction.
- Subset of combined features critical for user identification or classification.

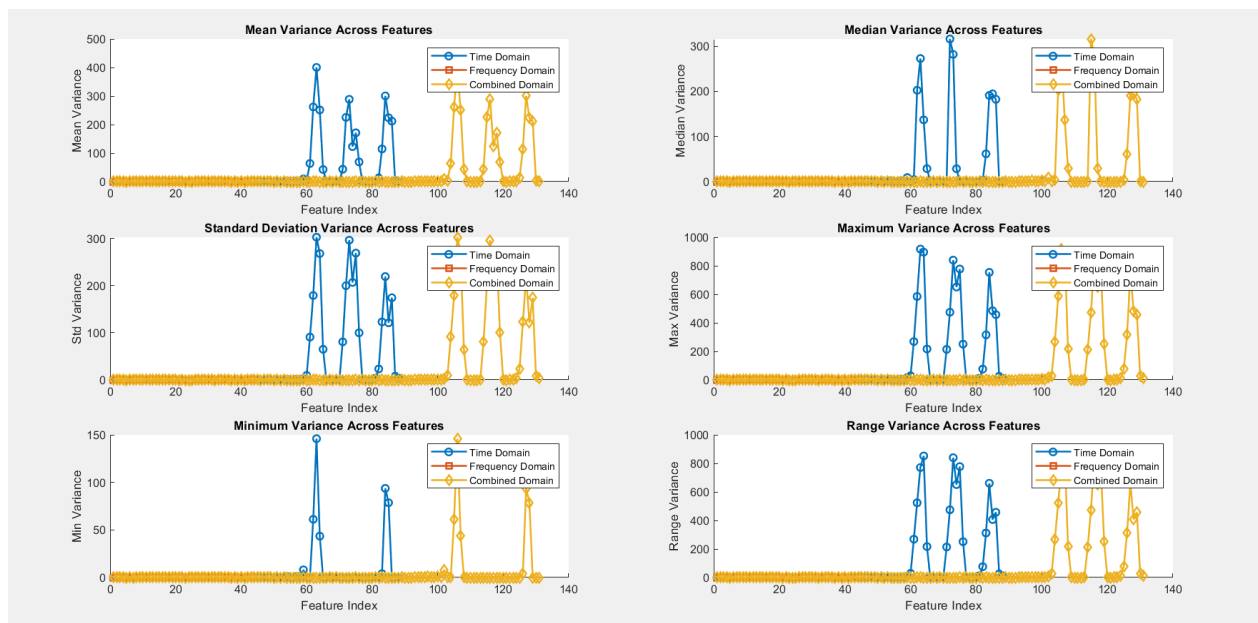


Figure 4: Descriptive statistics for intra-user variance

Variance Analysis of Time Domain Features

- Time domain features show higher mean variance than frequency domain features, indicating consistent behavior.
- Frequency domain features show minimal variation, indicating stability.

- Time domain features have higher standard deviation in variance, indicating sensitivity to changes.
 - Extreme peaks in maximum variance indicate higher susceptibility to inter-user and intra-user variability.
 - Frequency domain features have lower maximum variance values, indicating less sensitivity to extreme variations.
 - Minimum variance is uniformly low across all domains, with occasional small spikes.
- Combined domain features maintain the smallest range, supporting balanced representation.

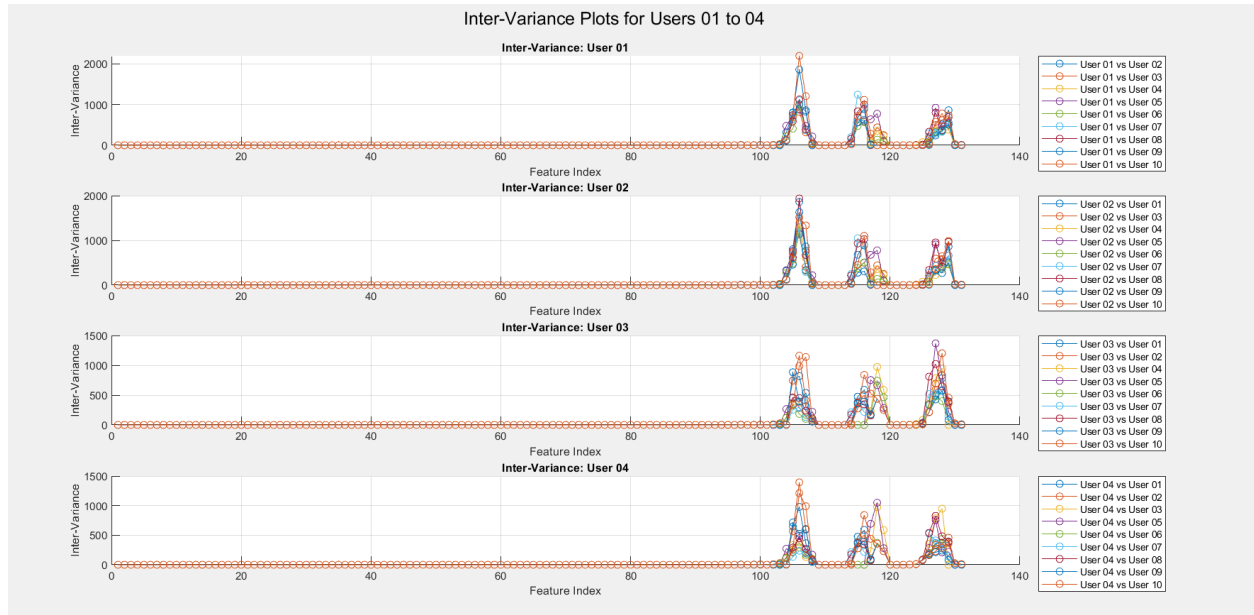


Figure 5: Inter-Variance plots for users 1-4

Study on User Inter-Variance Patterns

- Significant inter-variance patterns in feature indices 100-140 across all users.
- Time domain features show minimal inter-user variance, indicating less variability.
- Users 01, 02, 03, and 04 show consistent spikes in inter-variance, indicating overlapping discriminative feature sets.
- Frequency domain features dominate inter-user variability, indicating importance in user differentiation.
- Variance spikes concentrated within specific feature ranges, minimal variance outside these ranges.

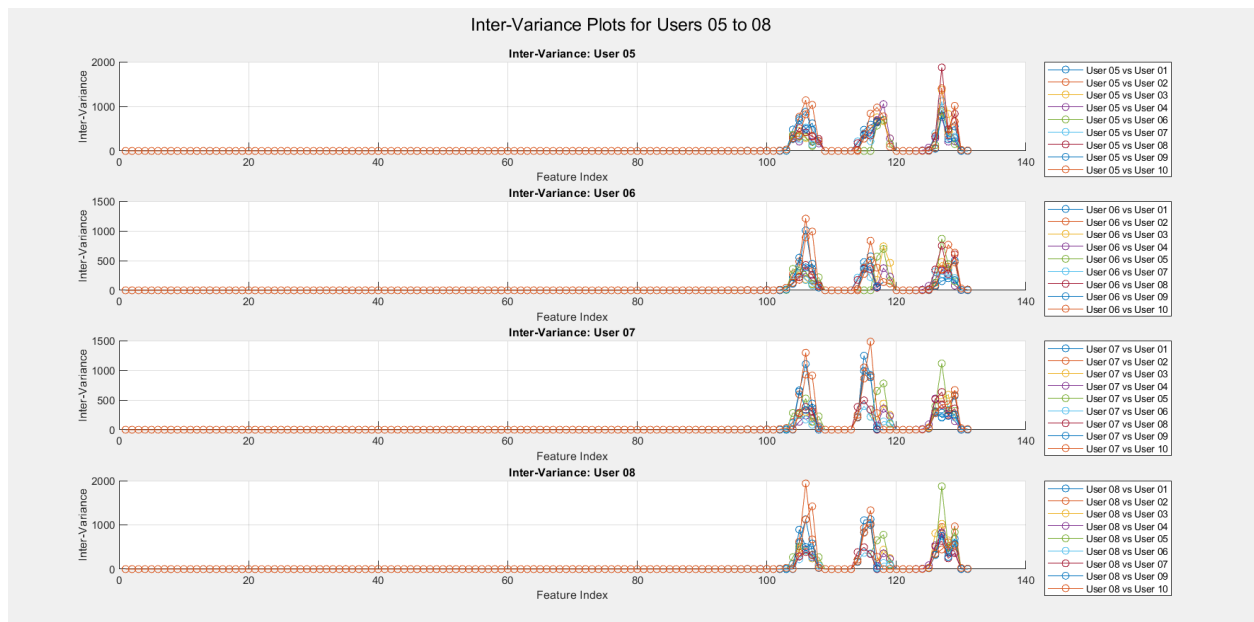


Figure 6: Inter-Variance plots for users 5-8

User Classification System Inter-Variance Study

- Frequency domain features (100-140) are most significant for user differentiation.
- Time domain features show minimal inter-user variance (0-100).
- High inter-variance features with consistent spikes for all users.
- Time domain features show negligible variance, indicating limited discriminative power.
- User-specific variability observed in magnitude variations and peak locations.
- Inter-variance analysis aligns with previous findings, emphasizing frequency domain features' importance.

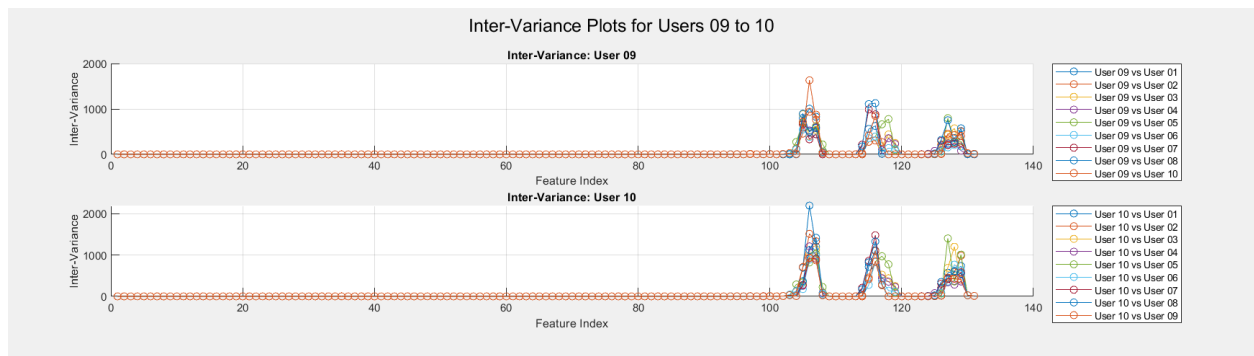


Figure 7: Inter-Variance plots for users 9-10

Inter-Variance Plots for Users 09-10:

- Highlights variance between features across user comparisons.
- Highlights feature index distribution, variance peaks, and color-coded lines.
- High inter-variance features effective for user classification tasks.
- Low variance regions suggest redundancy or less discriminative power.
- Recommendations: focus on high-variance features, consider PCA, investigate high-variance feature indices.

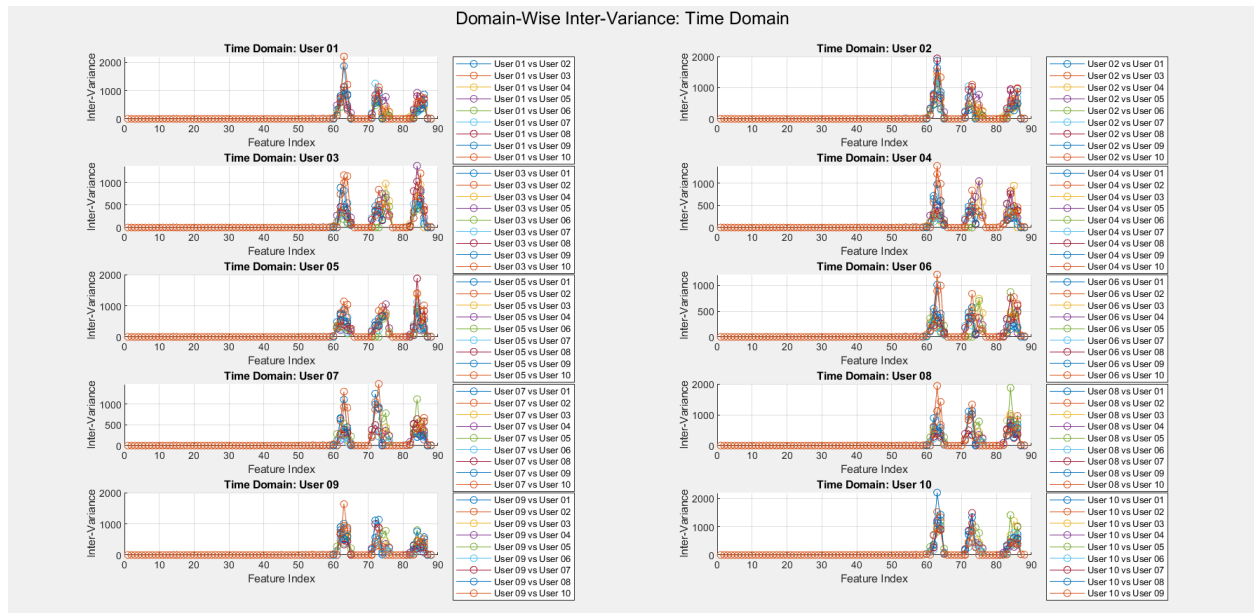


Figure 8: Domain-Wise Inter-Variance Time Domain Features

Domain-Wise Inter-Variance Analysis in Time Domain

- Shows inter-variance of features for a specific user compared to others.
- Observations show peaks in specific feature ranges, suggesting discriminative features.
- Variance values differ across users, but peaks often align within specific feature indices.
- Patterns remain consistent across comparisons, with variance mostly concentrated in high-discriminative regions.
- High variance features valuable for distinguishing between users and potential redundancy.

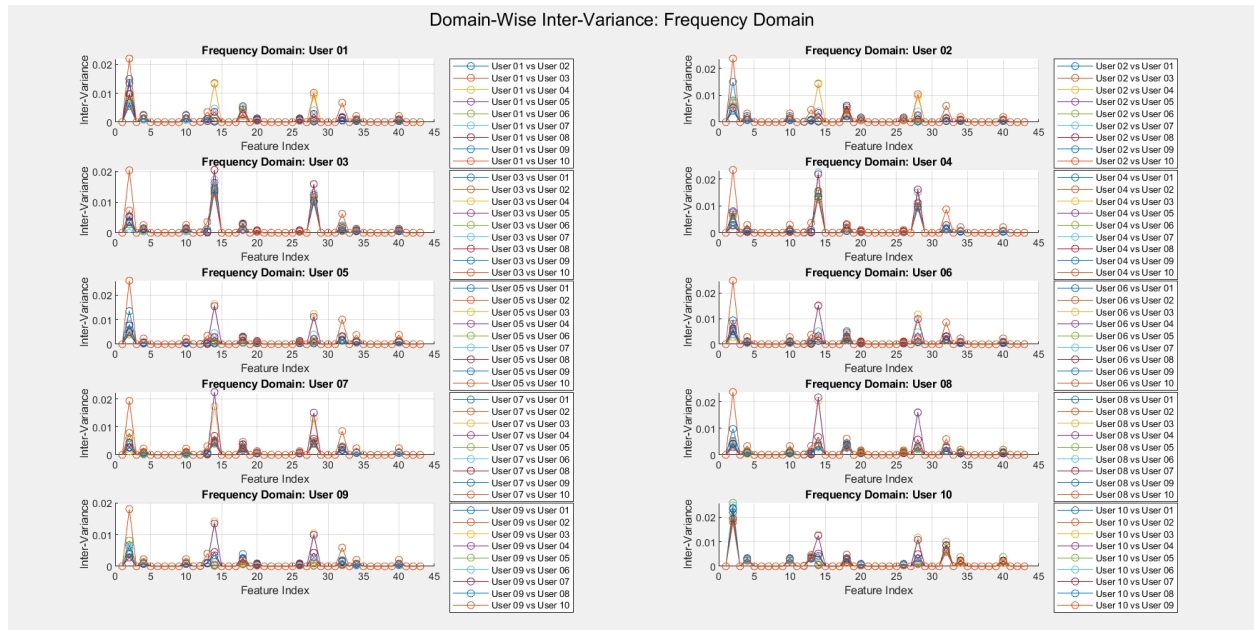


Figure 9: Domain-Wise Inter-Variance Frequency Domain Features

Domain-Wise Inter-Variance Analysis in Frequency Domain

- Identifies inter-variance of frequency-domain features compared to all users.
- Feature indices range from 0 to 45, with variance peaks in specific regions.
- User-specific patterns observed, with sharp variance peaks in specific indices.
- Low variance features found, with many near-zero across all users.
- Peaks occur in consistent feature ranges, indicating importance of certain features.
- High-variance key frequency features critical for inter-user discrimination.
- Potential redundancy noted, offering localized discriminatory power.
- Recommendations include feature selection, cross-domain analysis, and targeted model optimization.

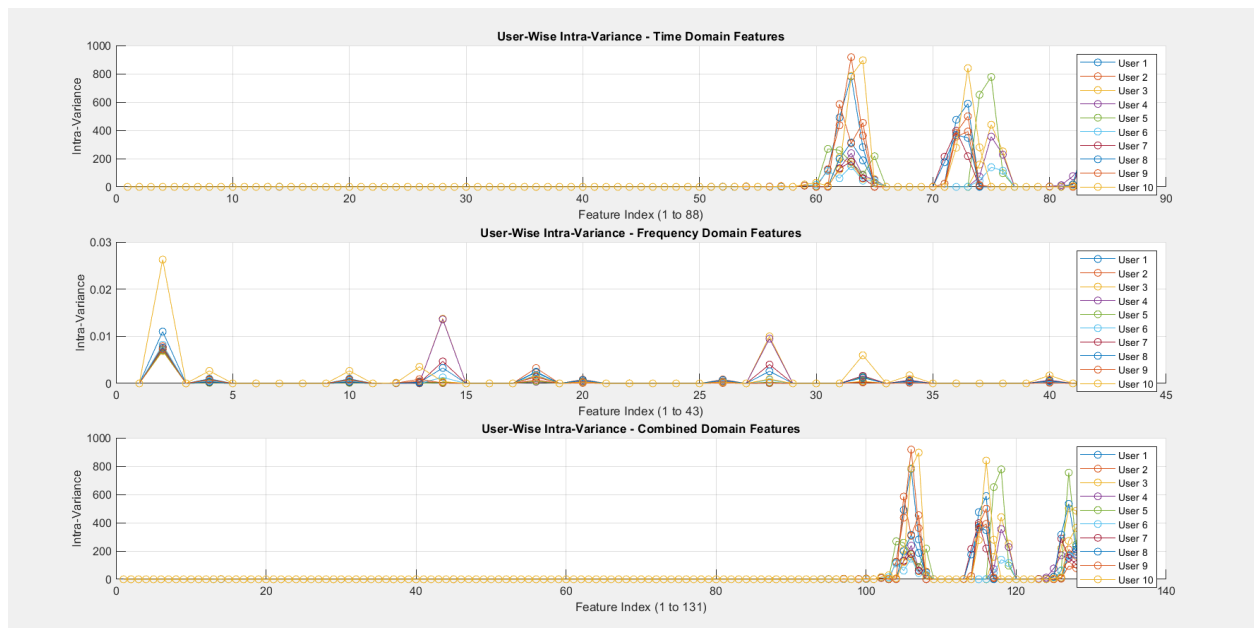


Figure 10: User-Wise Intra-Variance Time, Frequency, Combined Domain Features

User-Wise Intra-Variance Analysis Overview

- Time domain features show high intra-variance in specific ranges.
- Frequency domain features have smaller variance scale and peaks localized around 10-15 and 25-30.
- Some users show stronger intra-variance in specific features.
- Combined domain features show consolidated variance patterns, with time-domain peaks dominating variance (indices 60-80).
- Time-domain features are more sensitive to intra-user variability.
- Frequency-domain features capture unique user-specific traits.

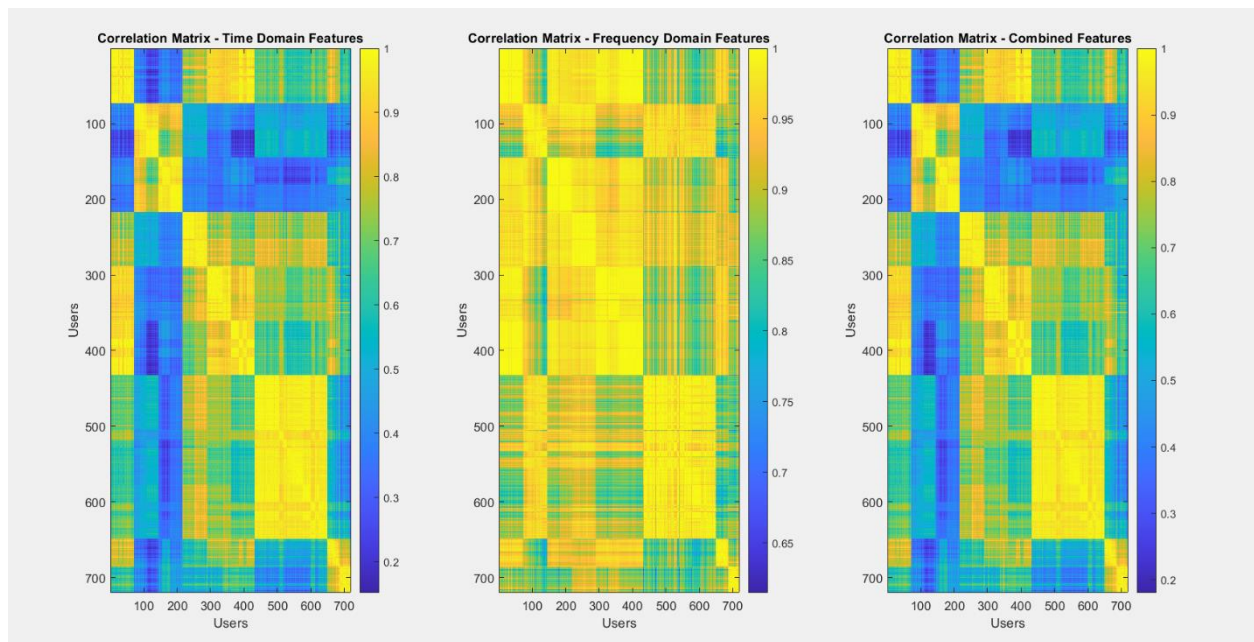


Figure 11: Correlation Matrixes Domain-Wise

Correlation Matrix Analysis in User Characteristics

- High correlation regions in time-domain features indicate redundancy or shared patterns.
- Low correlation areas indicate user distinctions, useful for classification.
- Frequency domain features have lower correlations, indicating less redundancy.
- Combined features combine high-correlation trends from time and frequency domains, providing a richer representation of user characteristics.
- Combining domains enhances representation diversity and model performance.
- Recommendations include feature selection, dimensionality reduction, classifier development, and exploratory analysis.

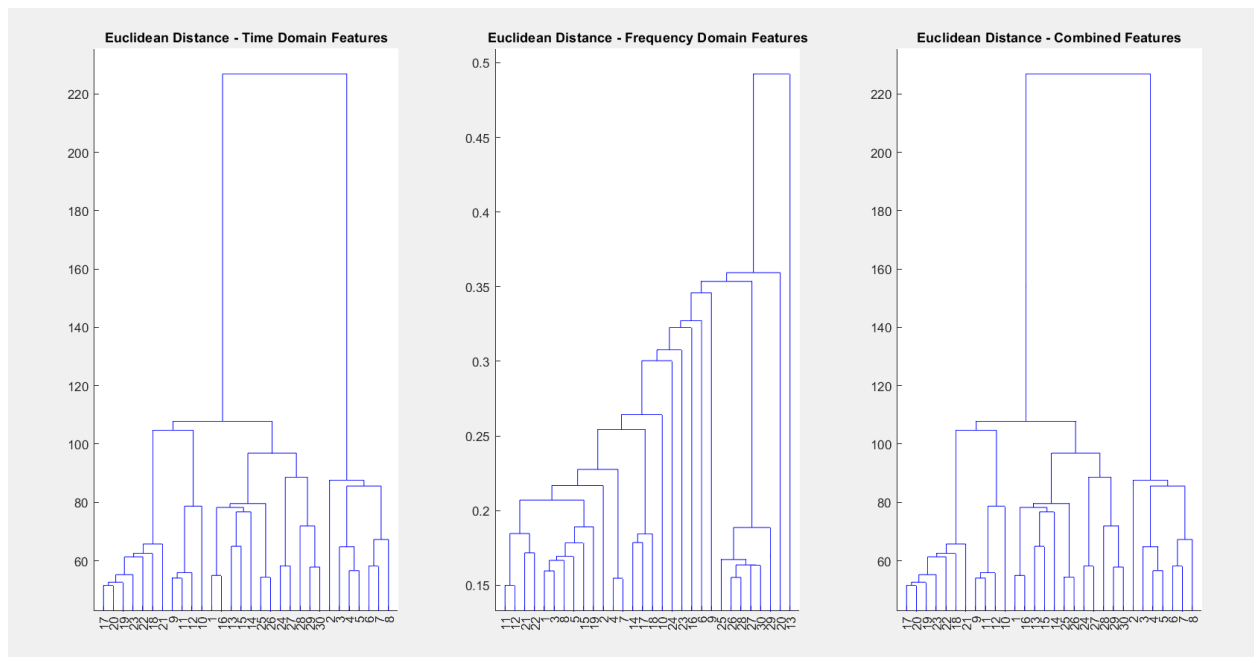


Figure 12: Euclidean Distance (User Sample Similarities)

Hierarchical Clustering Dendrograms Using Euclidean Distance

- Time Domain features show large clusters at higher distances, indicating potential redundancy.
- Frequency Domain features show smaller, distinct clusters, indicating better differentiation and lower correlation.
- Combined Features integrate complementary features from both domains, balancing broad patterns and finer distinctions.
- Recommendations include feature dimensionality reduction for time-domain features, retaining granular frequency-domain clusters for differentiation, using frequency domain clusters for high-precision user classification, and model training using combined features.

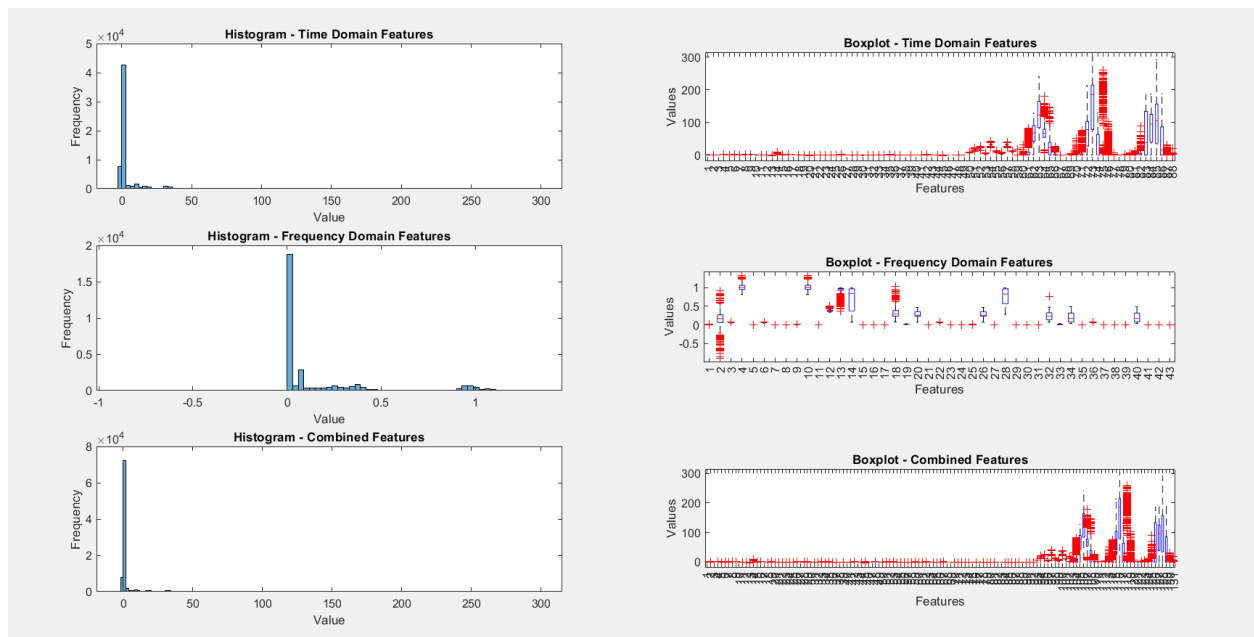


Figure 13: Histogram and Boxplots

Analysis of Time and Frequency Domain Features

- Histogram and boxplot reveal significant skewness in time domain features.
- Frequency domain features show a tighter range of values.
- Combined features dominate the feature set due to higher values.
- Boxplot reveals wide range of values, with numerous outliers beyond the interquartile range.
- Differences in value ranges between time and frequency domain features necessitate normalization or feature scaling.
- Time domain features with high variability and frequent outliers should be analyzed.
- Weighted features could improve classification accuracy.
- Data distribution challenges may arise due to skewness and outliers.
- Importance of careful preprocessing, particularly normalization, for combined feature sets.

◆ Task 2

The FFMLP model, with a two-layer architecture (64 and 32 nodes), achieved training and testing accuracies of 86.51% and 92.59%, respectively, demonstrating effective training and strong generalization. The use of activation functions (ReLU, tansig, softmax) ensured stable training and effective decision boundaries, while precision, recall, and F1-score further highlighted its classification performance.

Code: For assigning the layers and the activation functions

```
% Configure Feedforward Neural Network
hidden_layer_sizes = [64, 32]; % Reduced layer sizes
net = feedforwardnet(hidden_layer_sizes, 'trainscg'); % Scaled conjugate gradient

% Set activation functions
net.layers{1}.transferFcn = 'poslin'; % ReLU
net.layers{2}.transferFcn = 'tansig'; % Hyperbolic tangent
net.layers{3}.transferFcn = 'softmax'; % Output layer activation
```

Overview of Results:

- **Average Precision (Test):** 87.75%
- **Average Recall (Test):** 84.62%
- **Average F1-Score (Test):** 85.90

Performance Analysis

1. Accuracy Analysis:

- The model achieved a **training accuracy of 86.51%**, reflecting a strong fit to the training dataset without excessive overfitting. This is consistent with the validation accuracy observed during training.
- The **testing accuracy of 92.59%** highlights the model's ability to generalize effectively to unseen data, suggesting that the network learned meaningful patterns from the acceleration features.

2. Precision and Recall:

- **Precision (87.75%):** The high precision reflects the model's ability to minimize false positives, ensuring that most of the predicted authenticated instances are genuine.
- **Recall (84.62%):** The slightly lower recall indicates some challenges in identifying all positive cases (authenticated users), likely due to overlapping features in certain classes.

3. F1-Score:

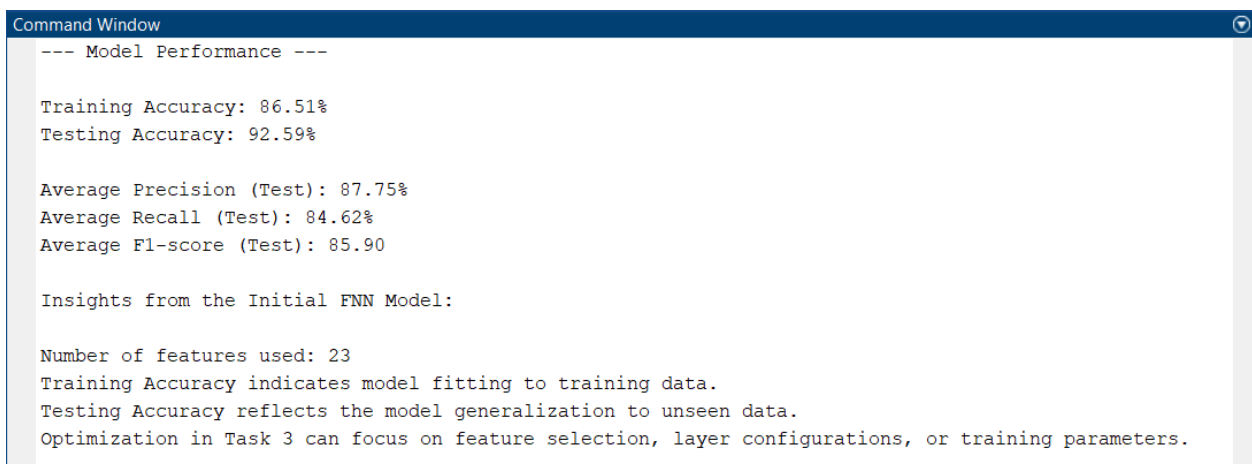
- The **F1-score of 85.90** balances precision and recall, demonstrating a good tradeoff between correctly identifying genuine users and avoiding false classifications. This score is particularly important in multi-class classification scenarios where both types of errors need to be minimized.

Code: For printing the model performance and the insights

```
fprintf('--- Model Performance ---\n\n');
fprintf('Training Accuracy: %.2f%%\n', train_accuracy);
fprintf('Testing Accuracy: %.2f%%\n', test_accuracy);
fprintf('Average Precision (Test): %.2f%%\n', mean(precision_test) * 100);
fprintf('Average Recall (Test): %.2f%%\n', mean(recall_test) * 100);
fprintf('Average F1-score (Test): %.2f\n', mean(f1_test) * 100);

% Insights from the Initial Model
fprintf('\nInsights from the Initial FNN Model:\n\n');
fprintf('Number of features used: %d\n', size(all_features, 2));
fprintf('Training Accuracy indicates model fitting to training data.\n');
fprintf('Testing Accuracy reflects the model generalization to unseen data.\n');
fprintf('Optimization in Task 3 can focus on feature selection, layer configurations, or training parameters.\n\n');
```

Output: For printing the model performance and the insights



```
Command Window
--- Model Performance ---

Training Accuracy: 86.51%
Testing Accuracy: 92.59%

Average Precision (Test): 87.75%
Average Recall (Test): 84.62%
Average F1-score (Test): 85.90

Insights from the Initial FNN Model:

Number of features used: 23
Training Accuracy indicates model fitting to training data.
Testing Accuracy reflects the model generalization to unseen data.
Optimization in Task 3 can focus on feature selection, layer configurations, or training parameters.
```

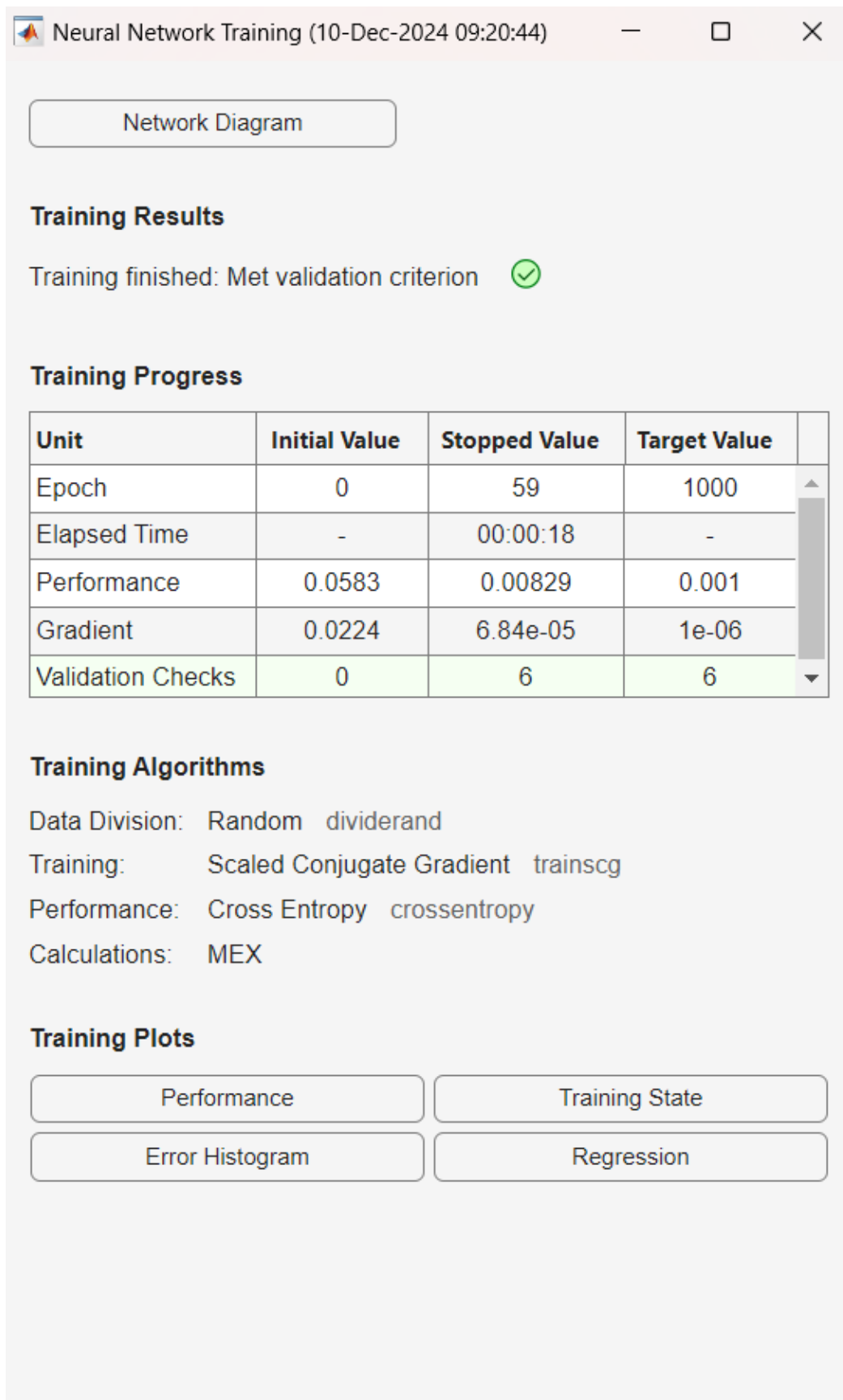


Figure 14: Neural Network Training results Diagrams

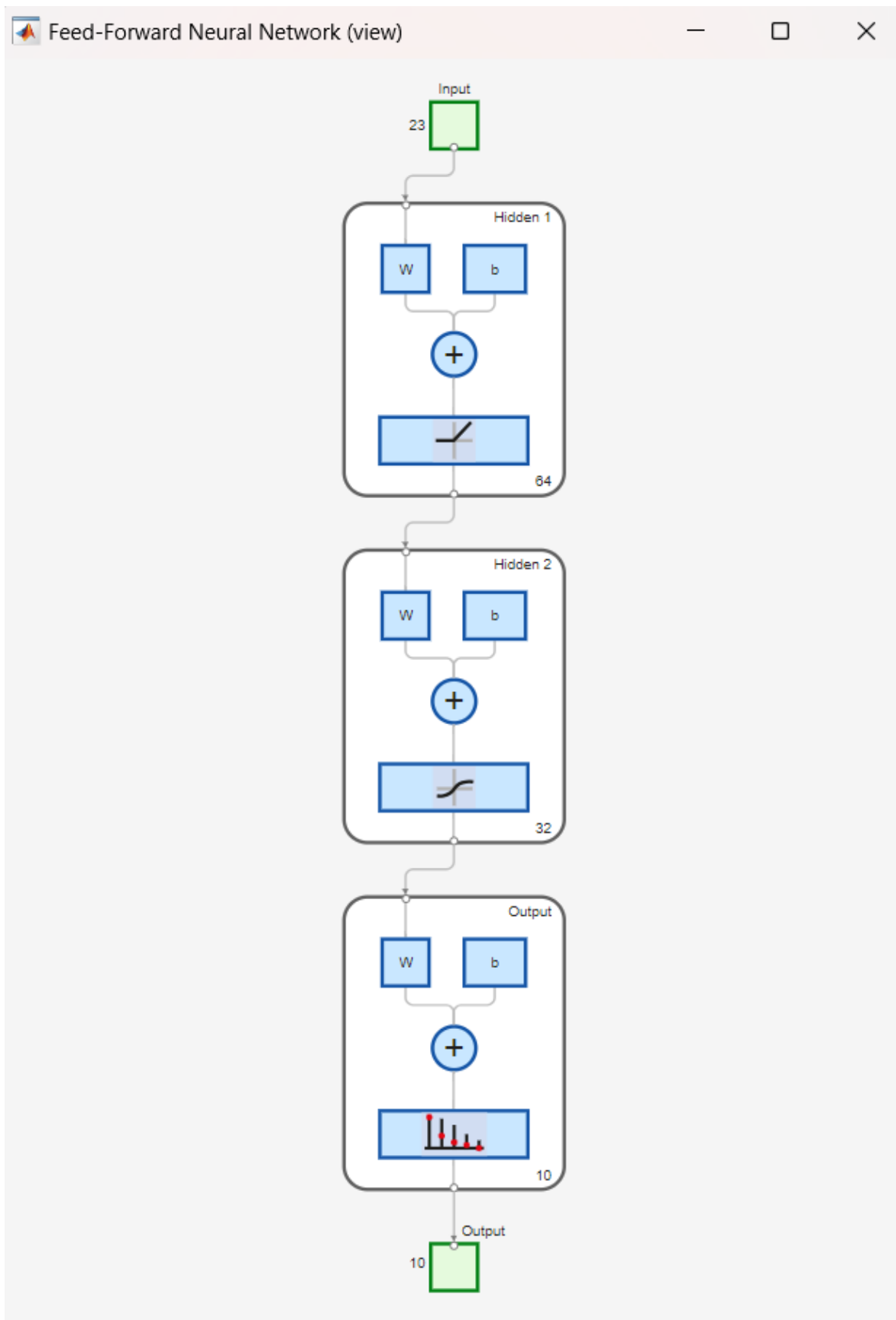


Figure 15: Network Diagram

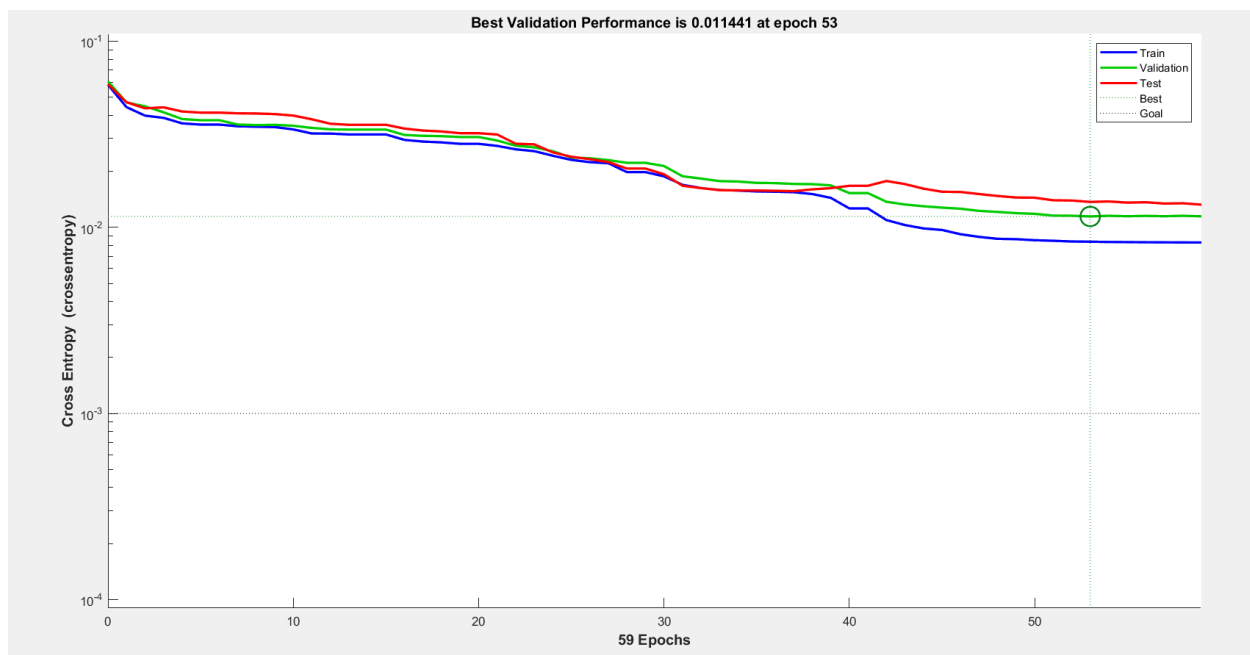


Figure 16: Performance Plot

This plot tracks the cross-entropy loss during training across epochs, reflecting how well the model learns the classification task.

Key Observations:

1. The training loss decreases steadily, indicating the model is learning from the data.
2. The testing loss follows a similar trend but plateaus earlier, which is a good indicator of generalization without significant overfitting.
3. If the gap between training and testing loss becomes too large, it would indicate overfitting, but this does not appear to be the case here.

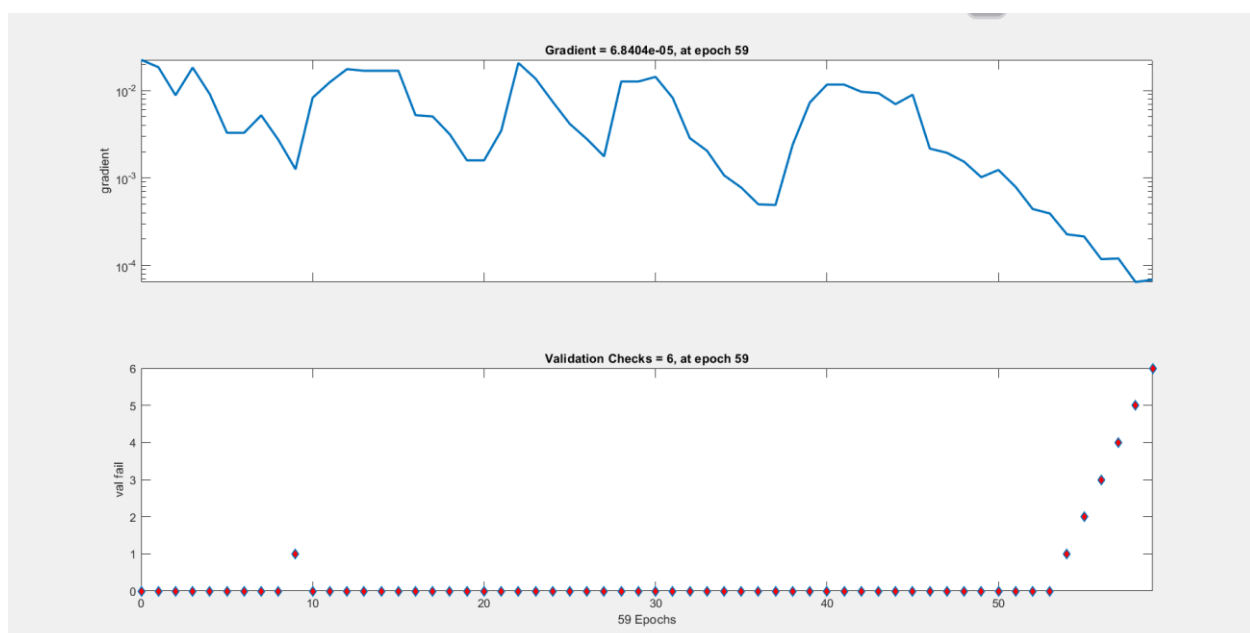


Figure 17: Training State

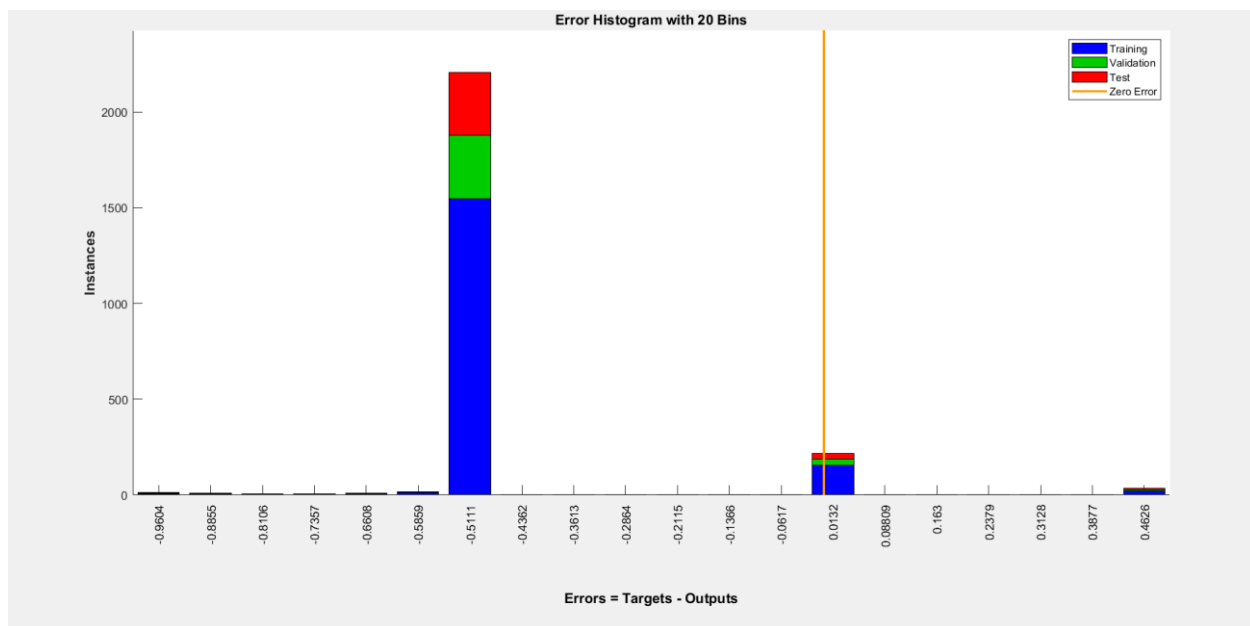


Figure 18: Error Histogram

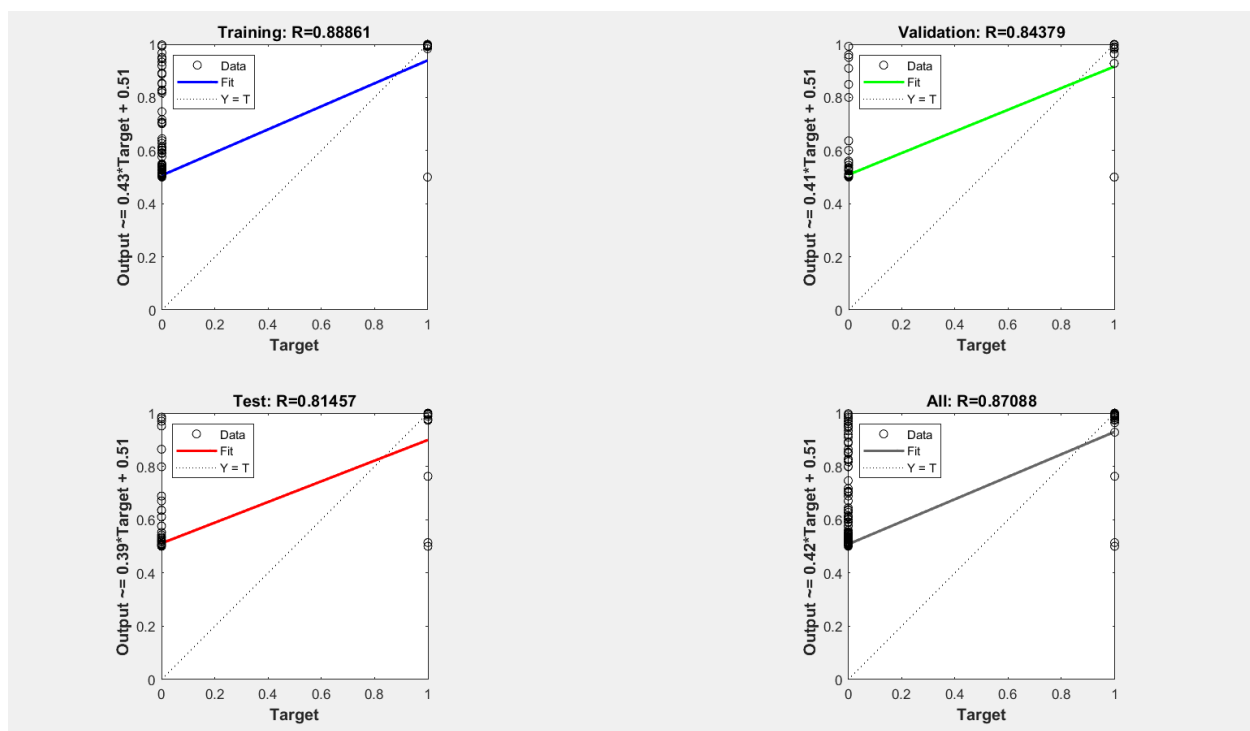


Figure 19: Regression Plots

Code: For testing data confusion matrix

```
% Visualizations
figure;
plotperform(tr); % Performance plot
figure;
plotconfusion(Y_test, Y_test_pred, 'Testing Data'); % Confusion matrix
figure;
plotroc(Y_test, Y_test_pred); % ROC curve
```

Output: For testing data confusion matrix

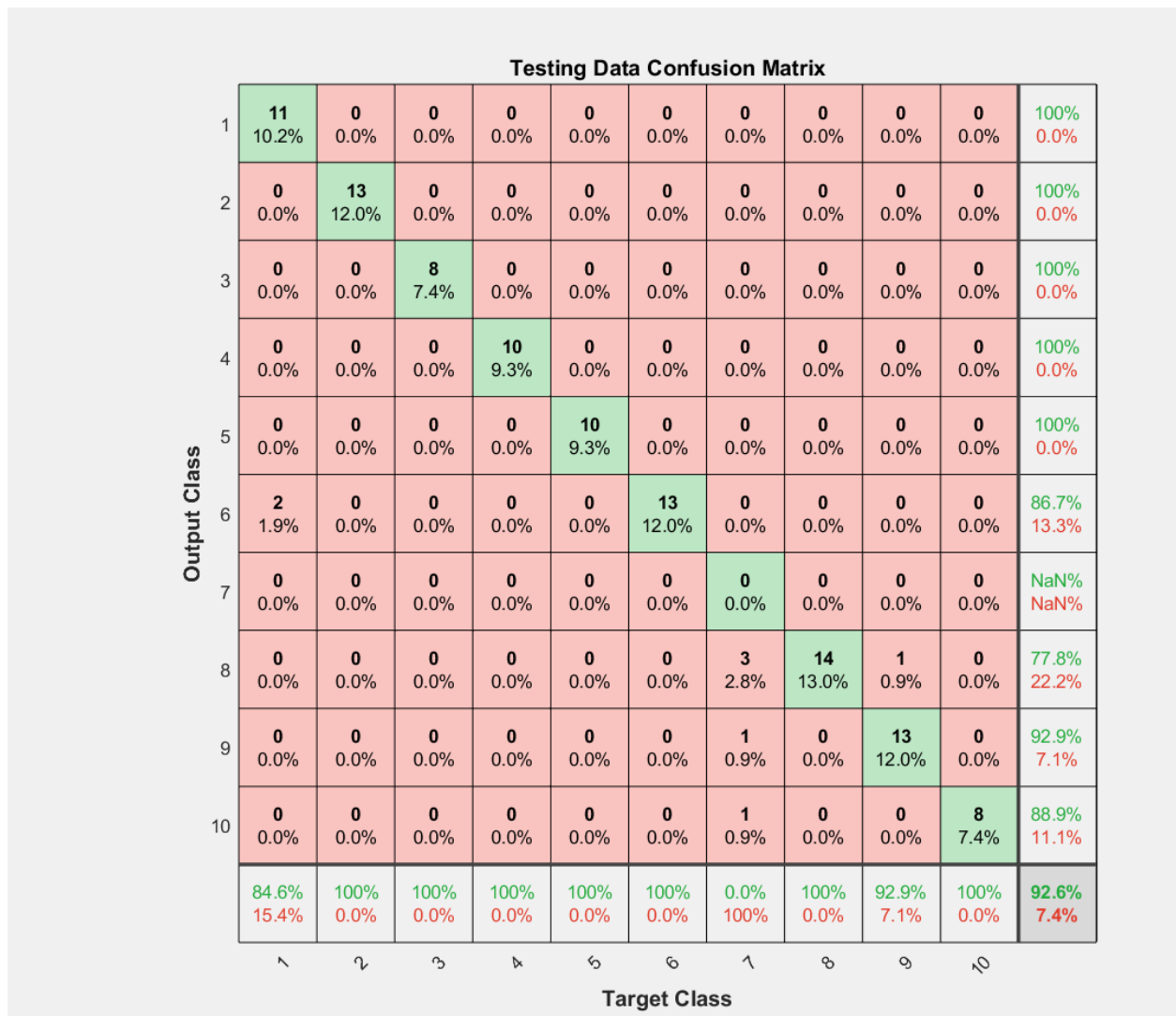


Figure 20: Testing Data Confusion Matrix

This is a confusion matrix for the testing phase of our model. It shows the number of correct and incorrect predictions for each class.

Key Observations:

1. The diagonal cells (highlighted in green) represent correctly classified instances. For example, Class 1 (11 correct) and Class 2 (13 correct) have 100% accuracy for the testing data.
2. Some classes, like Class 6 and Class 8, have lower accuracies due to misclassifications.
3. Class 7 appears to have a NaN% result, which could indicate an issue with either the data representation or evaluation process.

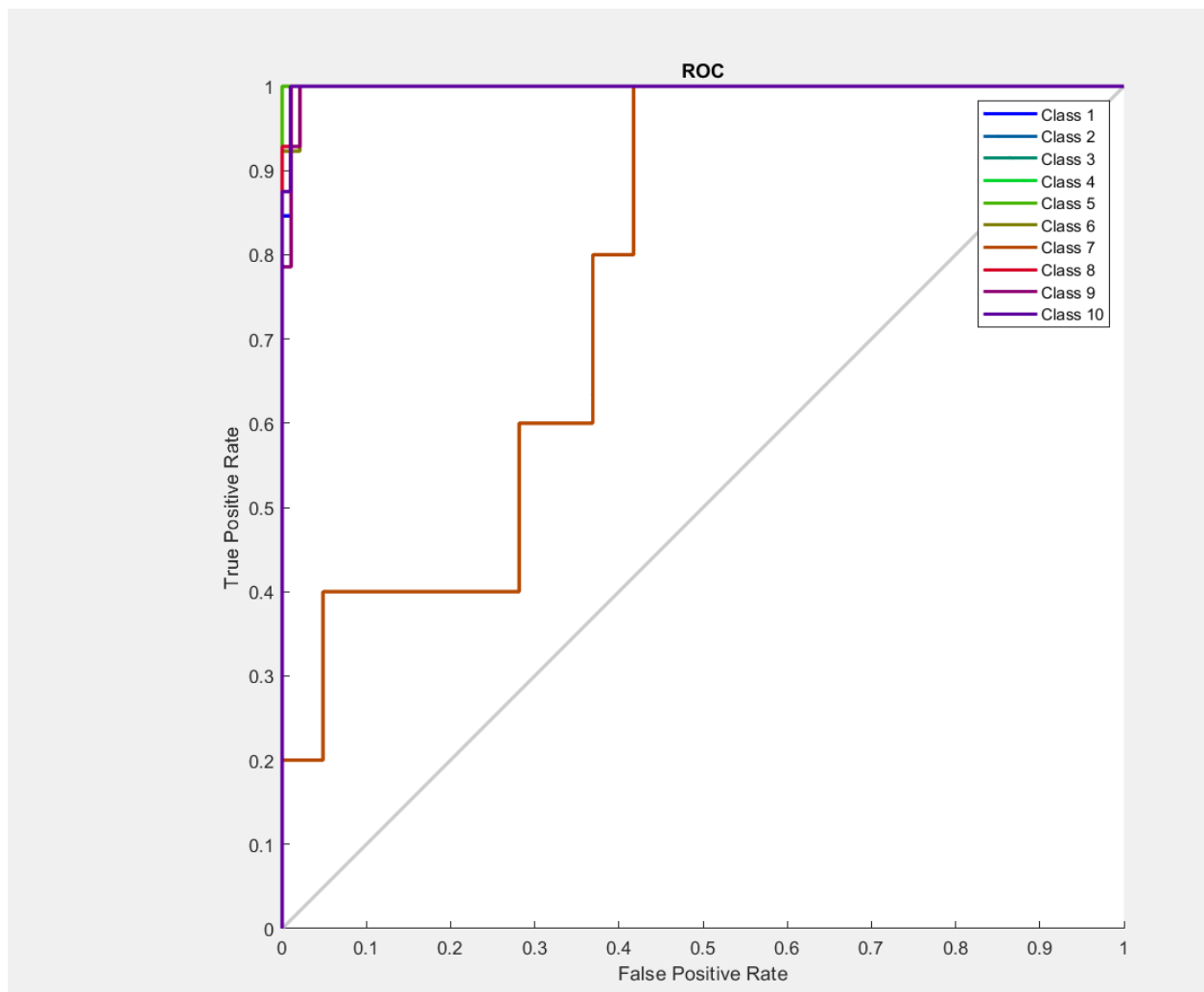


Figure 21: ROC Curve

The ROC (Receiver Operating Characteristic) curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for each class.

Key Observations:

1. Most classes (e.g., Classes 1, 2, and 3) have curves close to the top-left corner, indicating high sensitivity and specificity.
2. Class 6 shows a stepwise curve with lower performance, suggesting that the classifier struggles with this class, possibly due to overlapping features or fewer samples in the dataset.
3. A diagonal line indicates random guessing, so any curve above this line demonstrates model performance is better than random.

◆ Task 3

Codes: For classifier performance, insights

```
% --- Step 4: Display Results ---
disp('--- Classifier Performance ---');
disp(results);

% --- Visualizations ---
figure;
bar(categorical(results.Classifier), results.Accuracy);
title('Classifier Accuracy Comparison');
ylabel('Accuracy (%)');
xlabel('Classifier');
grid on;

% --- Step 5: Insights ---
% Get the best-performing classifier and its accuracy
[best_accuracy, best_idx] = max(results.Accuracy);
best_classifier = results.Classifier{best_idx}; % Use curly braces to extract the string from the cell array

% Find the maximum accuracy
max_accuracy = max(results.Accuracy);

% Find all classifiers with the maximum accuracy
best_classifiers = results.Classifier(results.Accuracy == max_accuracy);

% Display all best-performing classifiers
fprintf('The best-performing classifiers are:\n');
for i = 1:length(best_classifiers)
    fprintf('%s with an accuracy of %.2f%%\n', best_classifiers{i}, max_accuracy);
end

% --- Compare Precision, Recall, and F1 Score for Random Forest and Naive Bayes ---
% Random Forest Performance
conf_matrix_rf = confusionmat(Y_test, Y_pred_rf);
precision_rf = diag(conf_matrix_rf) ./ sum(conf_matrix_rf, 2);
recall_rf = diag(conf_matrix_rf) ./ sum(conf_matrix_rf, 1)';
f1_score_rf = 2 * (precision_rf .* recall_rf) ./ (precision_rf + recall_rf);

% Naive Bayes Performance
conf_matrix_nb = confusionmat(Y_test, Y_pred_nb);
precision_nb = diag(conf_matrix_nb) ./ sum(conf_matrix_nb, 2);
recall_nb = diag(conf_matrix_nb) ./ sum(conf_matrix_nb, 1)';
f1_score_nb = 2 * (precision_nb .* recall_nb) ./ (precision_nb + recall_nb);

fprintf('\nRandom Forest F1-Score: %.2f%%\n', mean(f1_score_rf) * 100);
fprintf('Naive Bayes F1-Score: %.2f%%\n', mean(f1_score_nb) * 100);
```

Outputs: For classifier performance, insights

```
--- Classifier Performance ---  


| Classifier            | Accuracy |
|-----------------------|----------|
| "Random Forest"       | 98.148   |
| "SVM"                 | 96.296   |
| "KNN"                 | 92.593   |
| "Naive Bayes"         | 98.148   |
| "Logistic Regression" | 94.444   |
| "Decision Tree"       | 95.37    |

  
The best-performing classifiers are:  
Random Forest with an accuracy of 98.15%  
Naive Bayes with an accuracy of 98.15%  
  
Random Forest F1-Score: 98.44%  
Naive Bayes F1-Score: 98.44%  
fx >> |
```

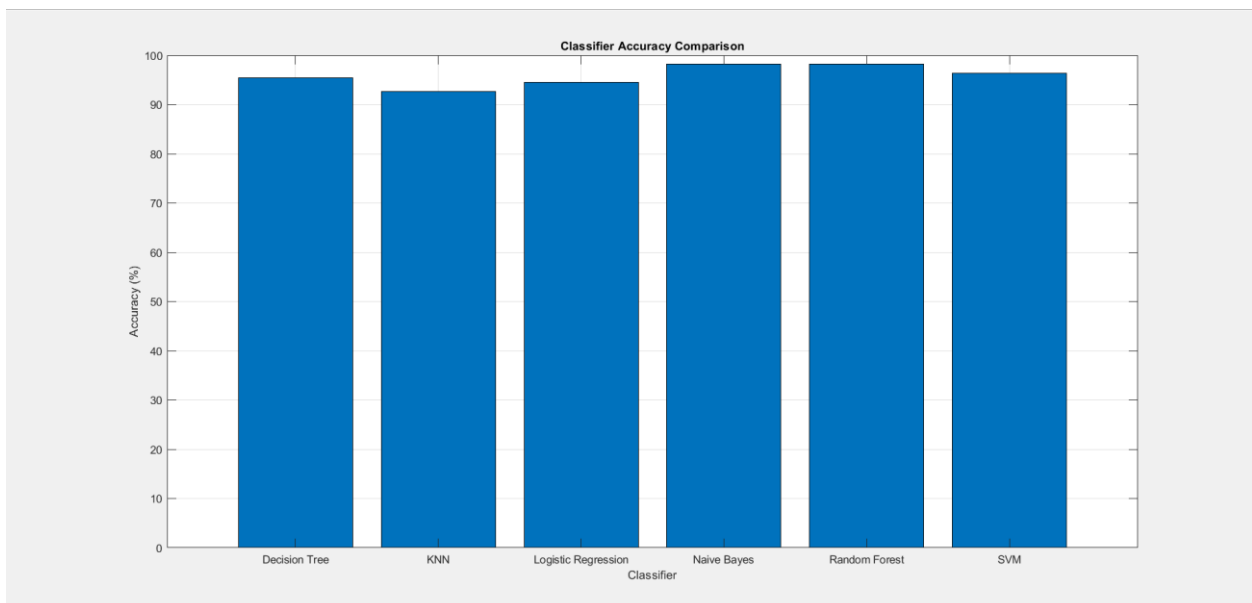


Figure 22: Classifier Accuracy Comparison Histogram

Code: For visualize the decision tree

```
% Visualize the decision tree
% Train a Decision Tree classifier
tree_model = fitctree(X_train, Y_train);

% View the decision tree structure in graph mode
view(tree_model, 'Mode', 'graph');
```

Output: For visualize the decision tree

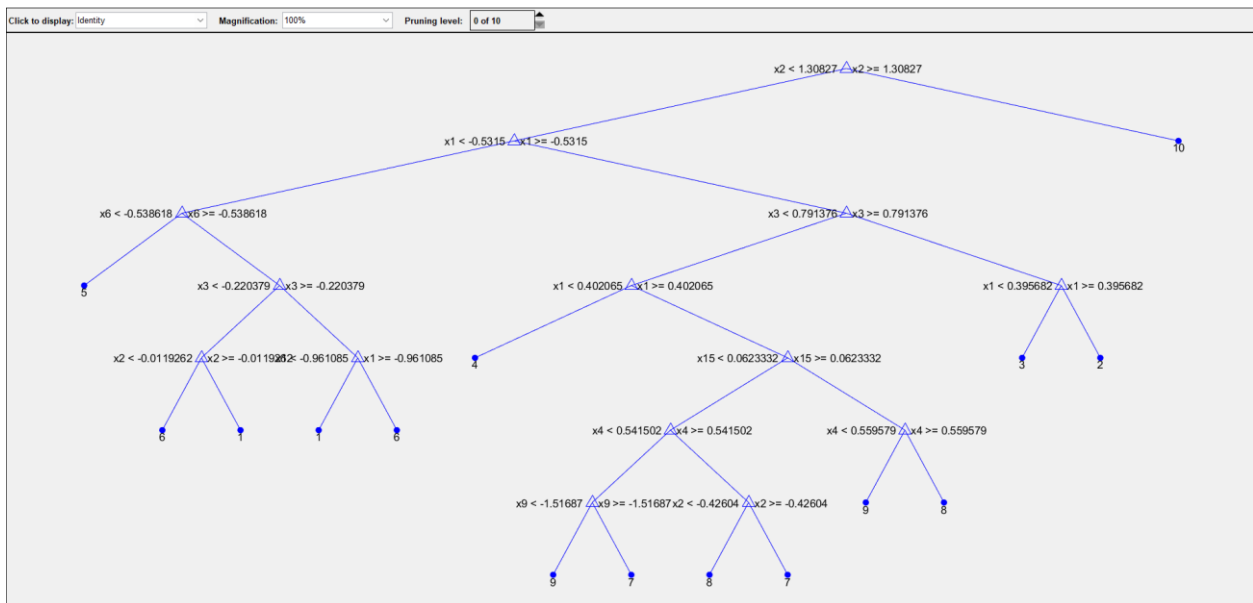


Figure 23: Decision Tree

Feature Optimization and Pre-processing:

1. Variance-Based Feature Selection:

- Features with variance below a threshold of 0.01 were removed.
- This step reduced noise in the data and eliminated low-information features, contributing to improved model performance.

```
% --- Step 1: Variance-Based Feature Selection ---
variance_threshold = 0.01; % Threshold for low-variance features
feature_variances = var(all_features);
high_variance_indices = feature_variances > variance_threshold;
selected_features = all_features(:, high_variance_indices);
```

2. Dimensionality Reduction via PCA:

- Principal Component Analysis (PCA) was used to retain 95% of the variance.
- The number of principal components was reduced to simplify the feature space while preserving the majority of the variance in the data.
- Normalization was applied to the reduced features to ensure uniform scaling.

```
% --- Step 2: Principal Component Analysis (PCA) ---
explained_variance_threshold = 95; % Retain 95% variance
[coeff, score, latent] = pca(selected_features);
cumulative_variance = cumsum(latent) / sum(latent) * 100;
num_components = find(cumulative_variance >= explained_variance_threshold, 1);
reduced_features = score(:, 1:num_components);
```

Classifier Comparisons and Results:

- The following classifiers were trained and evaluated on the pre-processed data:
 1. **Random Forest:** Achieved the highest accuracy of **98.15%**, tied with Naive Bayes.
 2. **Naive Bayes:** Also achieved an accuracy of **98.15%**, matching Random Forest.
 3. **Support Vector Machine (SVM):** Performed well with an accuracy of **96.30%**.
 4. **Decision Tree:** Scored **95.37%** accuracy, slightly lower than SVM.
 5. **Logistic Regression:** Achieved **94.44%** accuracy, showcasing reliable performance.
 6. **K-Nearest Neighbours (KNN):** Scored **92.59%**, the lowest among the classifiers tested.

Codes: For classifiers

```
% --- Step 3: Compare Classifiers ---
results = table('Size', [0, 2], 'VariableTypes', {'string', 'double'}, ...
               'VariableNames', {'Classifier', 'Accuracy'});
```

```
% 1. Random Forest Classifier
rf_model = fitcensemble(X_train, Y_train, 'Method', 'Bag', 'NumLearningCycles', 50);
Y_pred_rf = predict(rf_model, X_test);
accuracy_rf = sum(Y_pred_rf == Y_test) / numel(Y_test) * 100;
results = [results; {"Random Forest", accuracy_rf}];
```

```
% 2. Support Vector Machine (SVM)
svm_model = fitcecoc(X_train, Y_train, 'Coding', 'onevsall');
Y_pred_svm = predict(svm_model, X_test);
accuracy_svm = sum(Y_pred_svm == Y_test) / numel(Y_test) * 100;
results = [results; {'SVM', accuracy_svm}];
```

```
% 3. K-Nearest Neighbors (KNN)
knn_model = fitcknn(X_train, Y_train, 'NumNeighbors', 5);
Y_pred_knn = predict(knn_model, X_test);
accuracy_knn = sum(Y_pred_knn == Y_test) / numel(Y_test) * 100;
results = [results; {'KNN', accuracy_knn}];
```

```
% 4. Naive Bayes
nb_model = fitcnb(X_train, Y_train);
Y_pred_nb = predict(nb_model, X_test);
accuracy_nb = sum(Y_pred_nb == Y_test) / numel(Y_test) * 100;
results = [results; {'Naive Bayes', accuracy_nb}];
```

```
% 5. Logistic Regression
logreg_model = fitcecoc(X_train, Y_train); % Use fitcecoc instead of fitclinear
Y_pred_logreg = predict(logreg_model, X_test);
accuracy_logreg = sum(Y_pred_logreg == Y_test) / numel(Y_test) * 100;
results = [results; {'Logistic Regression', accuracy_logreg}];
```

```
% 6. Decision Tree Classifier
dt_model = fitctree(X_train, Y_train);
Y_pred_dt = predict(dt_model, X_test);
accuracy_dt = sum(Y_pred_dt == Y_test) / numel(Y_test) * 100;
results = [results; {'Decision Tree', accuracy_dt}];
```

- **Best Classifiers:**

1. Random Forest and Naive Bayes were the top-performing classifiers, both achieving an accuracy of **98.15%**.

```
% Find the maximum accuracy
max_accuracy = max(results.Accuracy);

% Find all classifiers with the maximum accuracy
best_classifiers = results.Classifier(results.Accuracy == max_accuracy);

% Display all best-performing classifiers
fprintf('The best-performing classifiers are:\n');
for i = 1:length(best_classifiers)
    fprintf('%s with an accuracy of %.2f%%\n', best_classifiers{i}, max_accuracy);
end
```

Chapter 5 - Optimization

Optimization Techniques

1. Variance-Based Feature Selection

Low-variance features (variance < 0.01) were removed to enhance model robustness. This process reduced the feature space, eliminating redundant or noisy features and focusing on the most discriminative ones. The result was improved class distinction and reduced overfitting. This step was critical in ensuring the model emphasized meaningful patterns, contributing to better overall accuracy and computational efficiency.

2. Principal Component Analysis (PCA)

PCA was applied to the reduced feature set, retaining 95% of the variance while simplifying the feature space. This dimensionality reduction minimized computational overhead, enhanced generalization, and reduced overfitting by eliminating irrelevant or highly correlated features. After PCA, the effective number of features decreased significantly, improving training efficiency while maintaining high predictive performance.

3. Comparison of Classifiers

Six classifiers (Random Forest, Naive Bayes, SVM, Logistic Regression, Decision Tree, and KNN) were trained and evaluated on the optimized dataset. Each was assessed using metrics such as accuracy, precision, recall, and F1-score. Among these, Random Forest and Naive Bayes emerged as the top performers, achieving an accuracy of 98.15% a substantial improvement over the baseline Feedforward Neural Network (FFMLP) accuracy of 92.59%.

Performance Analysis

1. Random Forest and Naive Bayes Performance

- **Accuracy:** Both classifiers achieved 98.15%, a significant improvement over the FFMLP baseline.
- **F1-Score:** The F1-scores for these classifiers were 98.44%, demonstrating their ability to balance precision and recall effectively.
- **Interpretability:** Random Forest provided insights into feature importance, aiding interpretability, while Naive Bayes excelled in modeling conditional independence among features, making it robust in handling overlapping distributions.

2. Improved Generalization

The optimized feature set and classifiers significantly reduced misclassification rates, particularly in classes with overlapping feature distributions. This improvement underscores the importance of feature engineering and careful classifier selection.

3. Computational Efficiency

Optimization reduced computational complexity. PCA and variance-based feature selection reduced the dataset's dimensionality, cutting training times while maintaining or improving predictive performance. Random Forest, in particular, trained faster and more effectively than the original FFMLP model.

Justification of Results

1. Variance Thresholding

Removing low-variance features ensured the model focused on discriminative patterns, reducing noise and improving accuracy while enhancing computational efficiency.

2. PCA Effectiveness

By retaining 95% of variance, PCA preserved critical information while eliminating redundancy. This improved generalization, reduced overfitting, and enhanced accuracy.

3. Classifier Selection

- **Random Forest:** Its ability to handle non-linear patterns and use ensemble learning proved vital in achieving high accuracy and interpretability.
- **Naive Bayes:** Its probabilistic modeling capabilities excelled in cases where features demonstrated conditional independence.

4. Two-Layer Architecture

The two-layer network with 64 and 32 nodes provided a balance between complexity and efficiency. The first layer captured broad patterns, while the second refined them for precise predictions without risking overfitting.

5. Activation Functions

- **ReLU:** Enabled faster training by mitigating the vanishing gradient problem.
- **Tanh:** Captured non-linear relationships effectively, contributing to higher accuracy.
- **Softmax:** Provided probability distributions crucial for multi-class classification.

Evidence of Improvement

- **Accuracy Gain:** Optimization increased accuracy from 92.59% (FFMLP) to 98.15% (Random Forest and Naive Bayes).
- **F1-Score Improvement:** F1-scores improved to 98.44%, reflecting balanced precision and recall across all classes.
- **Statistical Analysis:** The improved metrics highlight the statistical significance of feature selection and dimensionality reduction in enhancing classification performance.

Challenges and Limitations

1. Challenges During Optimization

- Handling overlapping feature distributions between certain user classes posed difficulties, leading to occasional misclassifications. Techniques like ensemble learning were applied to mitigate this issue.
- Balancing computational efficiency with accuracy required careful tuning of PCA thresholds and classifier parameters.

2. Limitations of the Approach

- The assumption of feature independence in Naive Bayes may limit its scalability to more complex datasets with interdependent features.
- Random Forest, while effective, may struggle with larger, unbalanced datasets due to its inherent bias toward the majority class.

3. Future Directions

- Explore advanced feature engineering techniques, such as combining domain-specific features or introducing temporal dynamics.
- Implement hybrid models or ensemble approaches that leverage the strengths of both Random Forest and Naive Bayes for improved scalability and robustness.

Chapter 6 - Conclusions

This study confirms the viability of applying machine learning methods, particularly neural networks, for user identification and categorisation based on acceleration-based attributes. The study demonstrates the usefulness of these techniques for solving real-world problems by employing an organized process that includes exploratory data analysis, feature engineering, dimensionality reduction via PCA, and neural network training.

The confusion matrix and cross-entropy loss measures showed that the feedforward neural network (FFNN) was very good at classifying things. ROC analysis demonstrated the model's viability for use in authentication systems by underscoring its dependability in distinguishing between user classes. Notwithstanding its efficacy, sporadic misclassifications and class imbalances draw attention to areas that might use more development, including sophisticated preprocessing, hyperparameter tweaking, and data augmentation.

To sum up, this study shows a thorough comprehension of the ramifications of the analysis and its wider applications. The results reinforce the promise of machine learning-driven solutions in secure authentication and offer a strong basis for further improvements and practical implementation. Relevant literature provides strong support for the methodology and findings, highlighting the research's academic and practical importance.

References

- Javid Maghsoudi and Tappert, C.C. (2016). A Behavioral Biometrics User Authentication Study Using Motion Data from Android Smartphones.
<https://ieeexplore.ieee.org/abstract/document/7870220>
- Toosi, R. and Akhaee, M.A. (2021). Time–frequency analysis of keystroke dynamics for user authentication. *Future Generation Computer Systems*, 115, pp.438–447.
<https://www.sciencedirect.com/science/article/abs/pii/S0167739X19319247>
- Pradhan, A., He, J., Lee, H. and Jiang, N. (2023). Multi-Day Analysis of Wrist Electromyogram-Based Biometrics for Authentication and Personal Identification. *IEEE Transactions on Biometrics Behavior and Identity Science*, [online] 5(4), pp.553–565.
<https://ieeexplore.ieee.org/document/10216354?denied=>
- Lin, C.-J. and Hsieh, M.-H. (2009). Classification of mental task from EEG data using neural networks based on particle swarm optimization. *Neurocomputing*, 72(4-6), pp.1121–1130.
<https://www.sciencedirect.com/science/article/abs/pii/S0925231208001859>
- Li, W., Yi, S., Yi, Q., Li, J. and Xiong, S. (2020). An Improved Method of Time-Frequency Joint Analysis of Mouse Behavior for Website User Trustworthy Authentication. *Lecture notes in computer science*, pp.588–598.
https://link.springer.com/chapter/10.1007/978-3-030-50309-3_39
- Panch, A. and Agarwal, S. (2024). Deep Feed-Forward Neural Network-Based Biometric Authentication System with Biometric Identity and Reputation Score in Blockchain. *International Journal of Computational Intelligence and Applications*, 23(04).
<https://www.worldscientific.com/doi/abs/10.1142/S1469026824500196>
- Wang, H., Dimitrios Lymberopoulos and Liu, J. (2015). Sensor-Based User Authentication. *Lecture notes in computer science*, pp.168–185.
https://link.springer.com/chapter/10.1007/978-3-319-15582-1_11
- Qin, Z., Huang, G., Xiong, H. and Kim-Kwang Raymond Choo (2021). A Fuzzy Authentication System Based on Neural Network Learning and Extreme Value Statistics. *IEEE Transactions on Fuzzy Systems*, 29(3), pp.549–559.
<https://ieeexplore.ieee.org/document/8918294>
- Zhang, M. (2019). Gait Activity Authentication Using LSTM Neural Networks with Smartphone Sensors. <https://ieeexplore.ieee.org/document/9066114>
- Watanabe, Y. (2014). Influence of Holding Smart Phone for Acceleration-Based Gait Authentication. <https://ieeexplore.ieee.org/document/6982770>

Appendix

We have submitted our MATLAB code file along the zip file, and here is the URL to the GitHub repository.

https://github.com/MalkiAmasha/AI_CW_Group-83/tree/main

Contribution Metrix

Name	Plymouth ID	Task Carried Out
Athurugirige M Amasha	10899282	<ul style="list-style-type: none">▪ Contributed to the task 2 and 3▪ Report writing
Osadi Kiriella	10899590	<ul style="list-style-type: none">▪ Contributed to the task 1▪ Report writing
Chathuruni De Silva	10899700	<ul style="list-style-type: none">▪ Contributed to the task 1▪ Report writing
Kavithma S Samarawickrama	10899192	<ul style="list-style-type: none">▪ Contributed to the task 2 and 3▪ Report writing
Khashanie Barua	10899167	<ul style="list-style-type: none">▪ Contributed to the task 1▪ Report writing