IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: Hapuarachchige Jayawardhana
Student Reference Number: 10818147

Module Code: PUSL3123	Module Name: AI and Machine Learning
Coursework Title: 'Review on Pattern Recognition: Keystroke Biometric'	
Deadline Date: 3rd January 2024	Member of staff responsible for coursework: Dr Neamah Al-Naffakh
Programme: Software Engineering	

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.

Hapuarachchige Jayawardhana 10818147
Shehan Sumudhitha 10749923
Hemachandra Wijesinghe 10819487

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.

Signed on behalf of the group: Hapuarachchige Jayawardhana

Individual assignment: ***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.***

Signed :

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.

I *have ~~used/~~not used translation software.

If used, please state name of software……

Overall mark _____% Assessors Initials _____ Date_____

*Please delete as appropriate Sci/ps/d:/students/cwkfrontcover/2013/14

Table of Contents

Table of Figures

Keystroke Biometric Pattern Recognition

Hapuarachchige Jayawardhana, Shehan Sumudhitha, Hemachandra Wijesinghe

CHAPTER 01: INTRODUCTION

Technology is developing at an ever faster pace, and security of authentication has become increasingly important. These factors are spurring researchers to investigate novel biometric modalities in order to come up with solutions for the problems posed by mobile technology today. **Keystroke** dynamics, that is characteristic and intrinsic typing behavior patterns existing in one individual alone may be considered to have received much attention. As one of the newer types out there, keystroke biometric pattern recognition is sure to see greater use in next-generation authentication systems. This literature review is designed to bring together keystroke biometrics research papers, Together with the highlights of these major landmarks in thought and work that are presented here, we hope it will provide a comprehensive overview of the various methodologies used as well as current advances or limitations surrounding this emerging type of front line.

1.1. Significance & Rationale for choosing Keystroke Dynamics

Typing on a keyboard is one of those personal activities that are determined by nature; muscle memory acquired through training in previous years or even incarnations; and typing habits. Keystroke dynamics aids in analyzing keystrokes, working from the separation between each character such as unique two-character rhythm. Keystrokes provide an unobtrusive means to authenticate users for their security and convenience based on timing and speed each key is pressed by users. Our choice of keystroke dynamics as a biometric modality is inspired by its special qualities and the fact that it can overcome shortcomings posed for traditional authentication methods. Additionally, because keystroke dynamics provides seamless means of user authentication, it should reduce reliance on easily compromised passwords regardless of its complexity and length. Hence, an additional layer of security needs to be implemented.

CHAPTER 02: LITERATURE REVIEW: KEYSTROKE BIOMETRIC PATTERN RECOGNITION

Keystroke dynamics is an area of deep biometric pattern recognition in which the characteristics used for user identification are typing patterns. This is a careful review of nine papers on Keystroke Biometric Pattern Recognition. The chosen articles offer detailed observations on wide-spread phenomenon like data collection methods, feature extraction and selection techniques; classifiers used in pattern recognition systems such as the Naive Bayes Classifier. The objective of this entire examination is to discover the manners and strategies in which keystroke dynamics holds promise for biometric pattern recognition.

2.1. Data Collection Methodologies

To understand what keystroke biometrics is, it first involves the collection of typing data in an accurate and efficient manner. According to studies by (Kasprowski, Borowska and Harezlak, 2022), a new approach that employs mobile devices is proposed which highlights the combination of accelerometer data in order to improve keystroke dynamics. This technique not only catches the existentialities of typing, but also includes mobility-spatial nuances introduced by mobile keyboards. At the other extreme, (Whiskerd et al., 2020) uses a standard keyboard but concentrates on identifying users through long-term automatic authentication by means of an analysis of dwell time and flight time. Such a juxtaposition of methodologies shows the range in approaches to extracting keystroke data, with some more convenient for users and others providing greater accuracy from the part seen system.

Using mobile devices and accelerometer data

Innovative data collection method Incorporating mobile devices into the keystroke dynamics framework introduced by (Kasprowski, Borowska and Harezlak, 2022). By pushing the applicability of keystroke biometrics beyond traditional keyboards, mobile device applications can take advantage Despite the diversity that exists in user interfaces. The method is not only capable of capturing typing patterns but also integrates accelerometer data, giving a multi-perspective insight into user behavior. With mobile devices in the mix, a bit more of that realism can be injected into data collection. People interact with various form factors-and it is this ecological validity which makes keystroke data collected from actual sources so important to begin with.

(Whiskerd et al., 2020): user authentication continued in dwell time and Flight Time

On the other hand, (Whiskerd et al., 2020) considers persistent authentication and introduces a collection model based on stay time and airtime testing. Dwell time is defined as the duration a key has been pressed while flight time refers to the interval between releasing one key and pressing another. The approach emphasizes the temporal nature of typing behavior, forming a mechanism for continuous monitoring to identify users. Additionally, the stress on real-time

revalidation means that a user's identity can be detected in changes occurring during an operation session where applications are developed continuously and may change their nature.

Comparative Analysis and Implications

This explains why, in keystroke biometrics , the drawbacks and advantages of different data collection methods need to be compared. Smart mobile The scope of application is greatly broadened, but this extension could also lead to a variable environment for typing. However, transit-time user authentication by building dwell time times flight time in the circulating area is definitely real time--but it must be carefully calibrated to determine accuracy and response. Mastering these subtleties is essential for choosing an efficient data collection methodology suitable to the unique characteristics and constraints of a particular application.

2.2. Feature Extraction/Selection Techniques

The key to keystroke biometrics is effective feature extraction. This step determines whether or not the system can capture unique patterns of typing characteristics for each individual user. The chosen research papers use various methods, each enhancing the richness and discrimination of the keystroke biometric system.

Emphasis on Pressure Dynamics

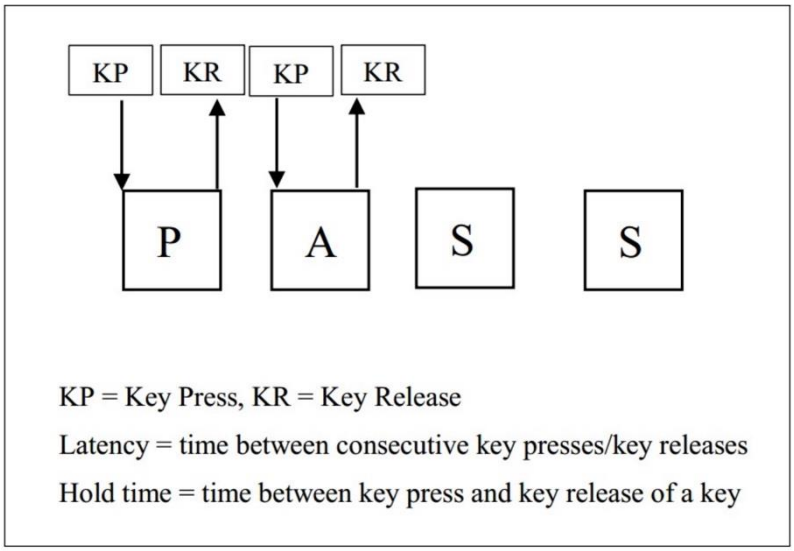

Figure 1 Timing Features of Keystrokes.

Latency and hold time--these timing features are employed in both to make a template of the user, and at testing time compare its timing features with these for identifying one. Time between key press of first key and key press of second key = KP2 - KP1, time between key press of first key and key release of second key = KR2 - KP1, time between key release of first key and key press of second key = KP2 - KR1, time between key release of first key and key release of second key = KR2 - KR1

Pressure dynamics during typing receives a great deal of attention by (Yaacob et al., 2020) in the effort to improve feature extraction. Finger anatomy and typing habits play a role in the way people press down on keys. This approach attempts to capture the unique subtleties of interaction between users and keyboards by extracting features related to large pressure fluctuations. When considering the higher behavioral characteristics of pressure dynamics, keystroke biometrics may further enhance its accuracy and dependability.

Coupling Keystroke Timing with Linguistic Structures

In contrast, (Kasprowski, Borowska and Harezlak, 2022) takes a linguistic approach by incorporating keystroke timing features with user-specific linguistics. This method exploits the natural correlation between typing habits and text being generated. Combined with these linguistic patterns, the system can begin to judge what is your personal style of typing. Combination of temporal and linguistic features make the representation of individual keystroke dynamics more comprehensive, so that it can increase the discriminative power of biometric system.

Comparative Analysis and Implications

The comparison between these two feature extraction techniques reminds us how important it is to use methodologies that suit the complexity of keystroke dynamics.

Pressure dynamics is a manifestation of physical nuance within the biometric system; language patterns are another kind of cognitive dimension. Which route to take depends on application requirements and the balance desired between physiological aspects and behavioral parts. Exemplifying new trends in keystroke biometrics feature extraction are a combination of pressure and time characteristics discussed by (Yaacob et al., 2020) and (Kasprowski, Borowska and Harezlak, 2022). Perhaps in the future other types of contextual information, such as emotional states will be integrated into keystroke biometric systems to further enhance the detailing level at which features are extracted and even increase its power.

2.3. Classifier Utilization

In keystroke biometric pattern recognition, selection of classifiers is vital. It affects how well technology can detect differences in typing patterns and match up individuals clearly with their profiles. Although classifiers appearing in these research papers are quite varied, this shows that machine learning techniques can be applied to keystroke dynamics despite its specific manner.

SVMs (Support Vector Machines) for discriminant analyse

(Yaacob et al., 2020) takes a machine learning perspective, using Support Vector Machines (SVMs) as the classifier of choice. Nonlinear relationships are important in keystroke dynamics, and SVMs work very well with them. Mapping typing patterns into a high-dimensional space through transforms permits SVMs to make fine distinctions, and their resultant decision boundaries separate users according to how well they discriminate. Using this method, users can be distinguished based on their keyboard behavior. SVMs' ability to capture the complex relationships that exist in keystroke data is comprehensive.

Neural Networks for Adaptive Learning

On the other hand, (AbdelRaouf et al., 2023) explores applying neural networks--the key is in deep learning for adaptive learning. Since king-stop dynamics is hierarchical and nonlinear, neural networks which can automatically learn intricate patterns are harnessed to capture it. This strategy indicates that neural networks can adapt and generalize well to all kinds of typing habits. It means they are suitable, then, for keystroke biometric systems--where individual differences may be miniscule or changing with time itself. (Aldrich Jr et al., 2023)

2.3.1. Comparative Analysis and Implications

The comparative study of SVMs and neural networks for keystroke biometric pattern recognition shows that each method has its own advantages. SVMs are especially good at drawing well-defined decision boundaries in feature space, which makes them exceptionally robust. While neural networks are flexible, able to uncover complex patterns; They may be suited for situations with changing or subtle typing styles.

Therefore, when choosing between SVMs and neural networks the particular properties of keystroke datasets as well as requirements for a target application has to be taken into account. Unlike SVMs, neural networks are more flexible and adaptive in capturing the nuances of typing behavior. In cases where explicit decision boundaries work to an advantage, however, these may naturally suit better than do neural nets.

2.4. System Evaluation: Performance Metrics

In order to determine whether keystroke biometric systems can be effective in real-world applications, their performance needs analysis. The research papers selected make use of various performance indicators to judge the accuracy, maintainability and efficiency of their proposed systems. Thus one obtains a rich appreciation of strengths and weaknesses alike.

Sensitivity and Specificity Analysis using ROC Curves

The author (Shadman et al., 2023) carries out an in-depth system evaluation, focusing specially on Receiver Operating Characteristic (ROC) curve analysis. The ROC curve is a diagram that shows the relationship between sensitivity and specificity. Sensitivity considers how well the system can identify genuine users, and specificity concerns its ability to reject impostors. ROC curve plots these metrics against each other, giving a visual indicator of the performances system under different decision thresholds. This insight in particular sheds light on how changes in sensitivity and specificity affect the overall performance of a keystroke biometric system.

Integration of FAR and FRR for a More Comprehensive Evaluation

From a complementary perspective, (Gautam and Dawadi, 2017) puts forward an evaluation model involving both the False Acceptance Rate (FAR) and FRR; this offers a more complete portrait of system effectiveness. The FAR measures the proportion of incorrectly accepted impostors, while the FRR measure is used for correct rejection rate. With an appreciation of both the false acceptance and rejection rate, this framework offers a balanced appraisal. It testifies to how strong the system is against each type of error. The integrated evaluation system introduced in this paper is particularly effective for situations like those encountered at sites around the world, where it is important to minimize both false acceptance and rejection.

2.4.1. Comparative Analysis and Implications

System performance can be considered from various perspectives, as revealed by a comparison of ROC curve analysis and the FAR/FRR integration. The ROC curve is an in-depth examination of the tradeoff between sensitivity and specificity. According to this analysis, designers can set a system's operating point precisely based on how much emphasis wants assigned for any given application. By contrast, the FAR and FRR structure provides a look at things as a whole because it considers both error types; thus giving us some idea of how reliable the system really is in actual use.

The selection of performance metric is determined by the application's needs and for how large an error it can tolerate in various situations. Applications with high security usually require a system that places more emphasis on maximum reduction of both false acceptance and false rejection. Other applications, however, would adjust the sensitivity or specificity according to their own special criteria.

2.4.2. Synthesis and Trends

Integration of Mobile Devices and Accelerometer Data

Through critical analysis of these selected research papers on keystroke biometric pattern recognition, you can gain insight into developing trends and important insights. These obviously demonstrate the dynamic nature of this field itself. These cover advances, problems and possible future developments.(Alsuhibany and Almuqbil, 2021)

Detecting early signs of Mental Illness

(Liu et al., 2023)With the implementation of keystroke patterns enhanced with time range and type count as quantitative indicators, assists in tracing early signs of loneliness (Lim et al., 2023) or depression. With the proposed use of curve fitting to provide a better picture, post-hoc analysis was conducted to further understand the time intervals and performance based on specific days of the week.

Diverse Classifier Utilization

The range of uses for classifiers is also impressively displayed. For example, (Yaacob et al., 2020) employs Support Vector Machines (SVMs), while researcher Ming Li explores the application potential inherent in neural networks. This trend agrees with the gradual development of biometric systems that have realized no one classifier is necessarily perfect for all situations. As the keystroke biometrics system strives for adaptability and robust discrimination, there will likely be development of hybrid approaches combining SVMs with neural networks or other classifiers.

Comprehensive Performance Evaluation

This concern for comprehensive performance evaluation, which we see in the ROC curve analysis described by (Shadman et al., 2023), and even more so in FAR/FRR integration advocated thereafter (see works cited herein)1 points toward a balanced assessment of system effectiveness. This point that different applications may require varying degrees of sensitivity and specificity leads to the conceptual design of systems which are customizable for particular needs. this trend recognizes that the strength of keystroke biometric systems depends on a variety of factors and cannot be based solely upon one aspect.

Conclusion of Literature Review

Keystroke biometric pattern recognition is indeed a multilayered process. The literature review concludes that it involves various factors and stages through which the keystroke can be

considered as an effective characteristic for identification. Keystroke dynamics are designed for a robust authentication system whose component parts cohere while standing up well under scrutiny. On the other hand, solving existing problems and standardization are vital priorities for making keystroke biometrics really useful in real applications. In the wake of keystroke dynamics receiving steadily increased attention as an effective biometric modality, and with many researchers busy refining and advancing this technology forward, in just a few years one can envision improved security in wide areas.

CHAPTER 03. DISCUSSION

In the field of keystroke recognition systems, how data is collected turns out to be a key insect. Mobile devices' integration, as pointed out by (Kasprowski, Borowska and Harezlak, 2022), adds a new iron layer of possibilities and stretches beyond the traditional keyboard. As accelerometer data can enhance flexibility, however, it does come with its own set of problems. How are we to be sure that the analysis gleaned from this information is accurate and valuable in terms of describing a user interaction?

Then (Whiskerd et al., 2020) 'focus on real-time authentication is another aspect of this. This creates a delicate balance between truth and reactivity. Thus the issues related to data collection methodology are not limited only to technical aspects, but also include practical problems of integration and response.

For the extraction of features from keystroke dynamics, a development can be seen in (Yaacob et al., 2020) and 's refined methods. (Yaacob et al., 2020) goes into pressure dynamics, adding a physiological level to the analysis. It increases discriminatory power, and particularly when the additional physical characteristics serve to further identify users. In contrast, (Kasprowski, Borowska and Harezlak, 2022)'s linguistic method adds a cognitive aspect to the keystroke biometric system. This fits with the wider tendency toward a total user profile, going beyond temporal analysis of conceiving patterns to picture how physiological and behavioral factors interrelate.

The degree of classifier utilization becomes a key factor affecting the effectiveness of keystroke biometric systems. As (Yaacob et al., 2020) and (AbdelRaouf et al., 2023) demonstrate, sometimes it is necessary to be very flexible in the applications of classifiers. For instance in the case where explicit decision boundaries are important, SVMs (Support Vector Machines) prove their advantage. But neural networks are flexible, particularly effective with subtle or changing typing behaviors.

When we move to system performance evaluation, the significance of this very subject becomes clear. ROC curve analysis and FAR/FRR integration help tease apart subtleties in reliability. ROC curve analysis can be selected according to application requirements through the sensitivity-specificity tradeoff. The layered understanding of performance resulting from such a complex set of application requirements leads system designers into an overall assessment for keystroke biometric systems.

Approaches basically can be summarized as data collection methods, feature extraction techniques, classifier implementations and performance measurement. Challenges, innovation and practicality all play a part in this future where keystroke biometrics will be taking the lead position among secure and convenient authentication method

CHAPTER 04: CONCLUSION

Finally, this in-depth examination of keystroke biometric pattern recognition shows that the field is constantly changing. Through an in-depth literature review, we navigated the intricacies of data collection methodologies, feature extraction/selection techniques, classifier utilization, and system performance metrics, focusing specifically on the chosen biometric category: Keystrokes. The discussion shed light on the impacts of these factors upon proposed systems, revealing just how precarious that balance between security and convenience can be. Through merging mobile devices and subtle feature extraction, using diverse classifiers or striving for detailed system assessment--each piece further enriches the terrain of keystroke biometrics.

Challenges and Future Directions

Technology-based typing behaviors are still shaped by standardization in data collection. In the future, such problems may figure prominently in research that aims to resolve them. The lack of standardization in methods for collecting data is a bottleneck, which precludes the smooth comparison between results across studies. Given that keystroke biometric systems aims to provide generalized results, representative samples of the population are clearly necessary. In addition, external factors that affect keystroke robustness include user fatigue and moods which may be more serious in the case of elderly users. Variations in typing conditions (i.e., handwriting vs print), weak physical strength due to arthritis or eye troubles also introduce problems requiring further consideration by researchers.

Combined with trends and insights, the picture is of unending refinement and innovation that faces such challenges as standardization in data collection or diversity among datasets. The key lies in shaping keystroke biometrics into a robust and effective means of authentication for the future.

CHAPTER 05: PROJECT OUTPUTS

Output for Task 2.2 Neural Network

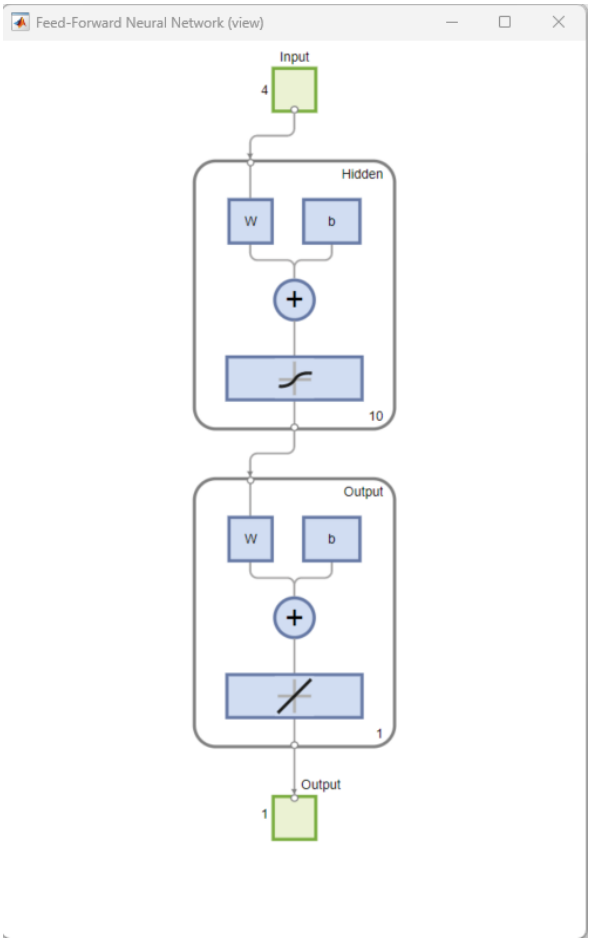

Figure 2 Hidden layer size 10

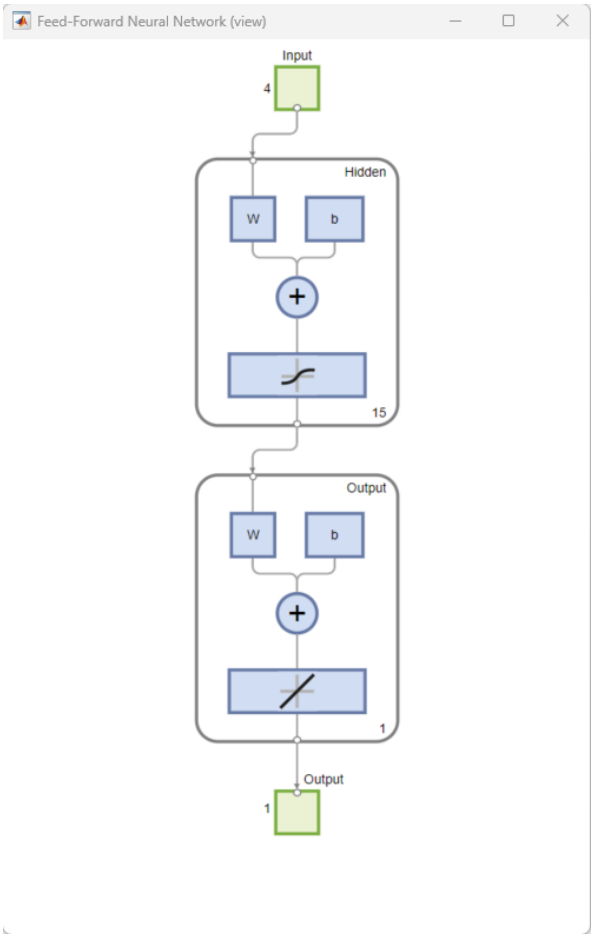

Figure 3 Hidden layer size 15

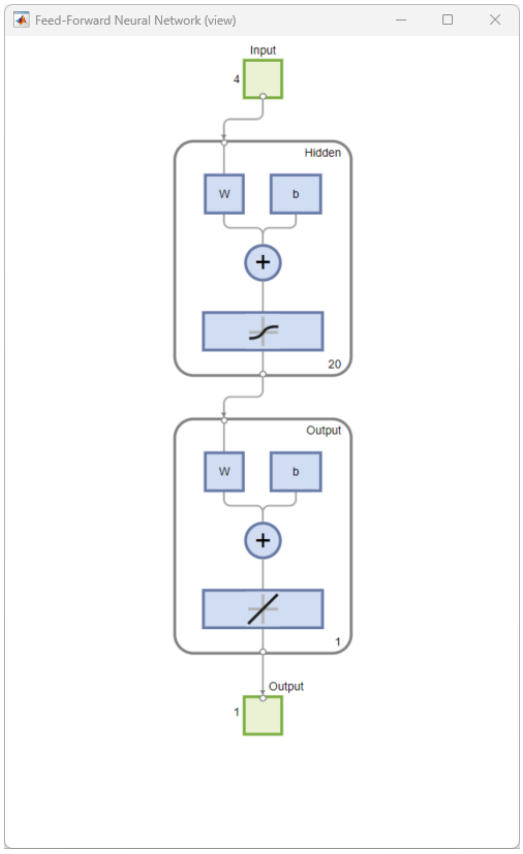

Figure 4 Hidden layer size 20

```
Command Window

  Hidden layer size 10 Experiment 1
  Accuracy: 96.6667

  Hidden layer size 10 Experiment 2
  Accuracy: 90

  Hidden layer size 10 Experiment 3
  Accuracy: 90

  Average percentage of correct classifications with 10 hidden layers and 3 experiments : 92.2222%


  Hidden layer size 15 Experiment 1
  Accuracy: 95

  Hidden layer size 15 Experiment 2
  Accuracy: 88.3333

  Hidden layer size 15 Experiment 3
  Accuracy: 88.3333

  Average percentage of correct classifications with 15 hidden layers and 3 experiments : 90.5556%


  Hidden layer size 20 Experiment 1
  Accuracy: 91.6667

  Hidden layer size 20 Experiment 2
  Accuracy: 91.6667

  Hidden layer size 20 Experiment 3
  Accuracy: 91.6667

  Average percentage of correct classifications with 20 hidden layers and 3 experiments : 91.6667%


fx >>
```

Figure 5 Neural network code output

Output for Task 2.3 K-means

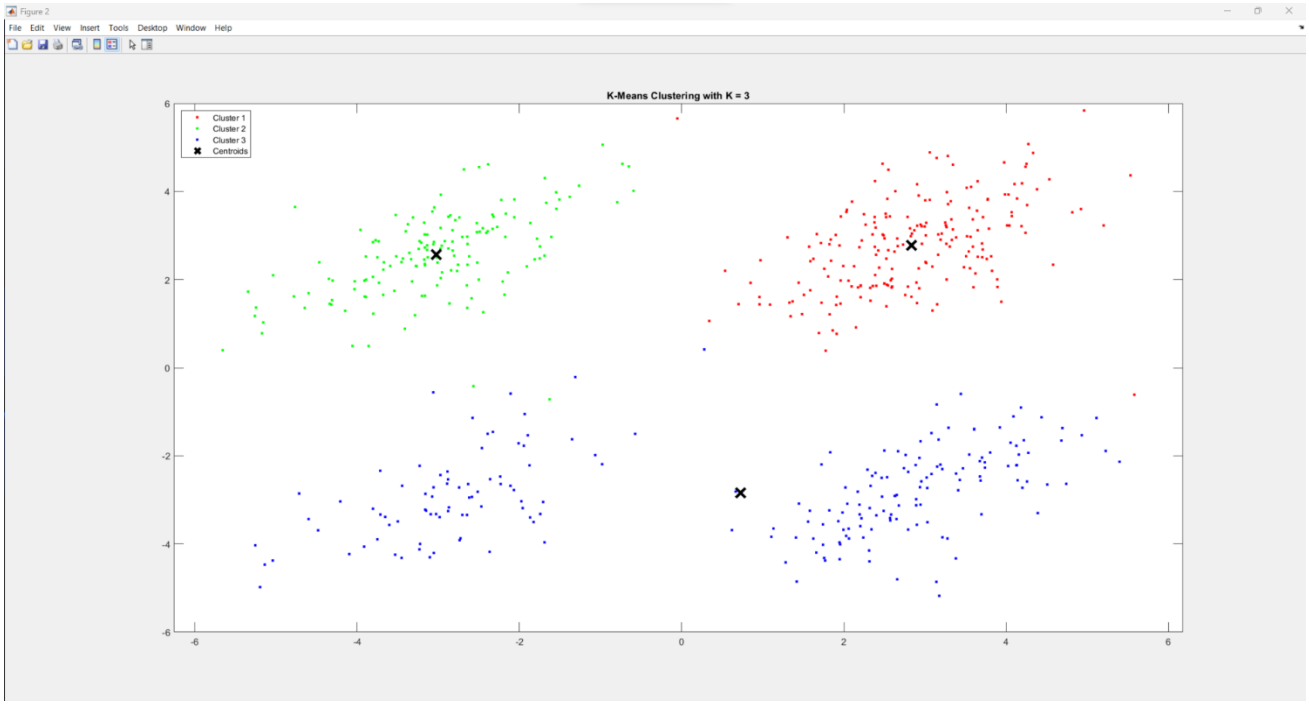

Figure 6 K = 3

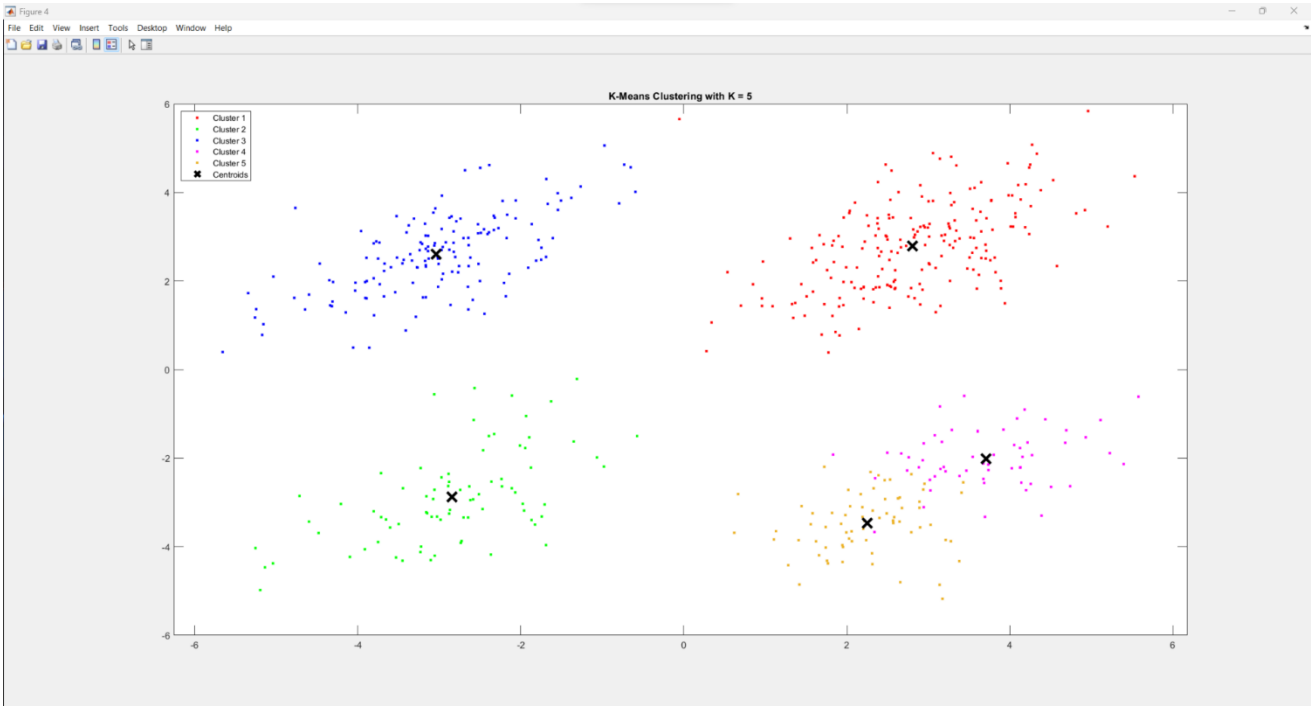

Figure 7 K = 5

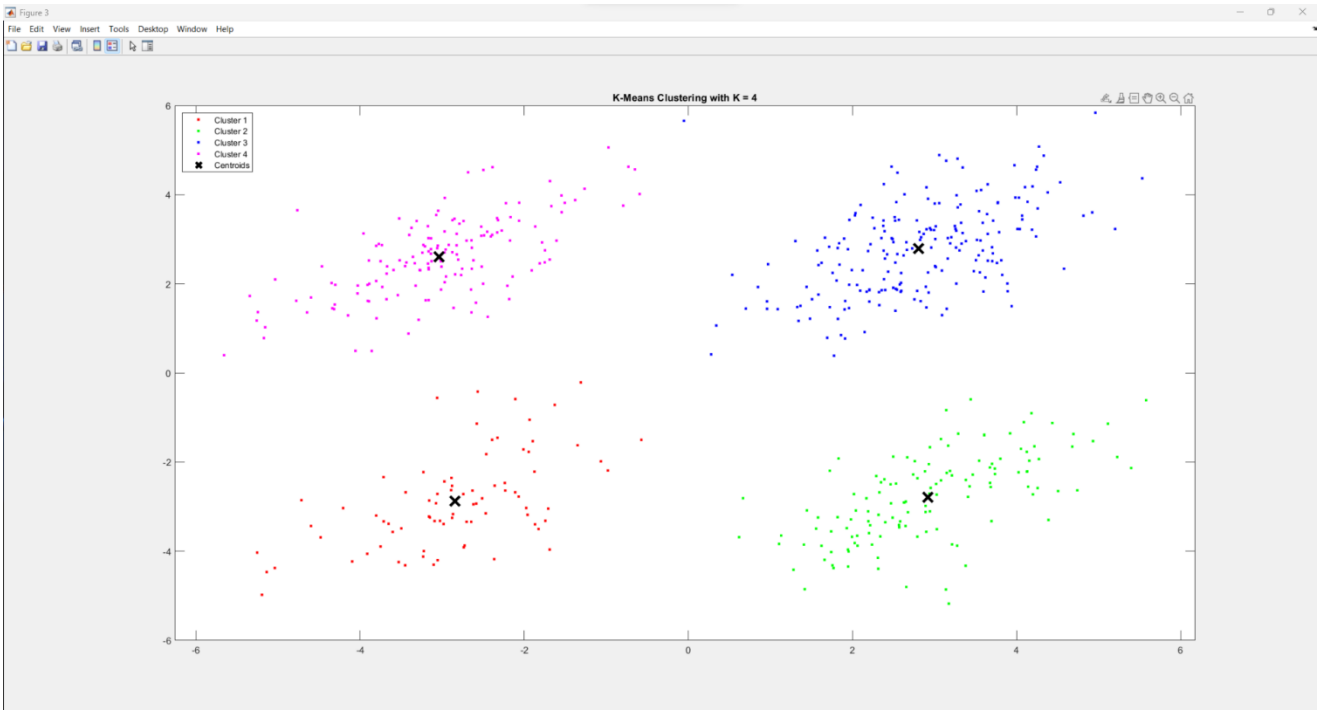

Figure 8 K = 4

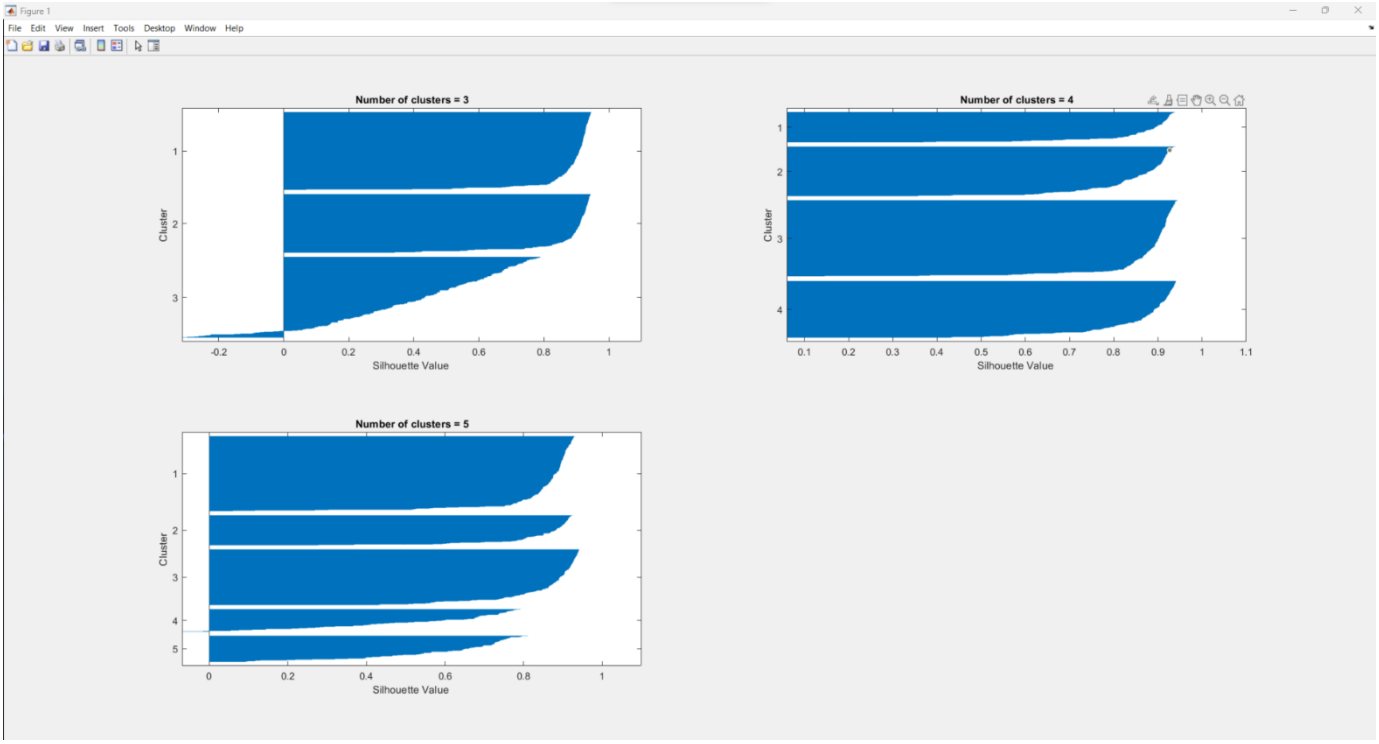

Figure 9 Silhouette for each cluster

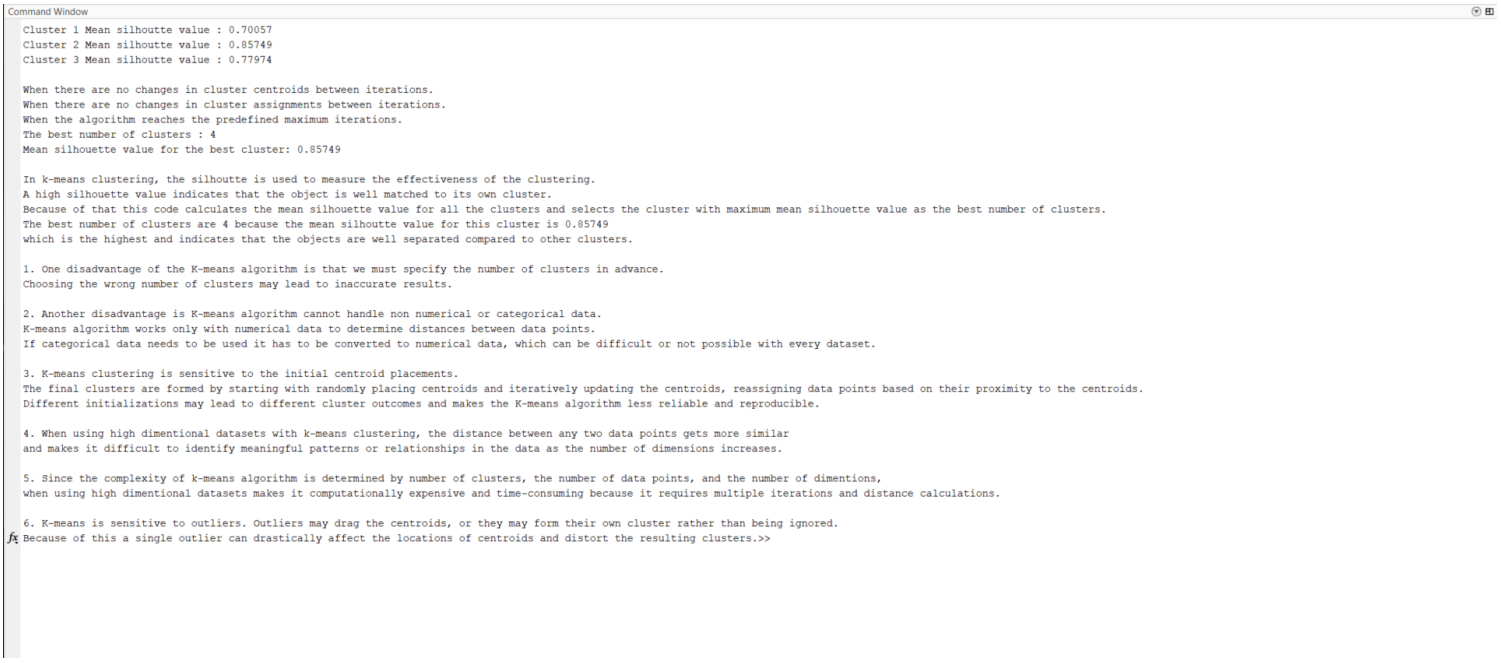

Figure 10 K-means code output

Output for Task 2.4 KNN

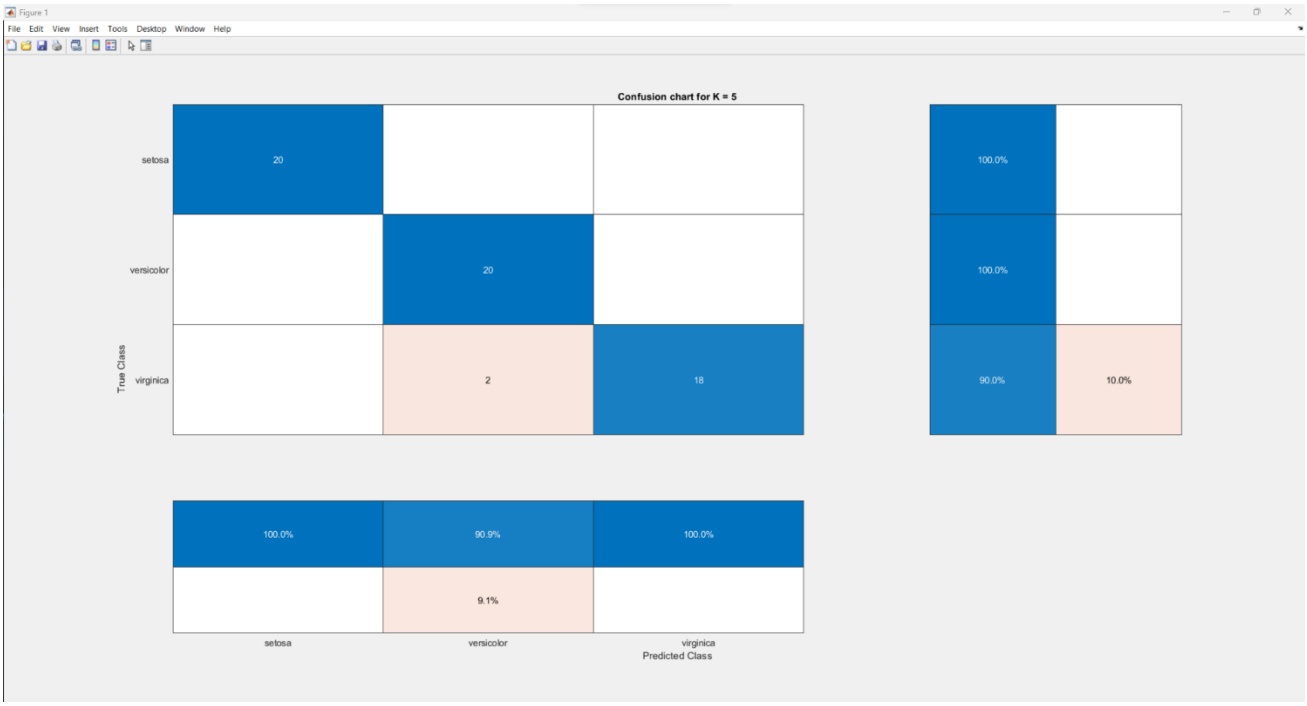

Figure 11 K = 5

Figure 12 K= 7

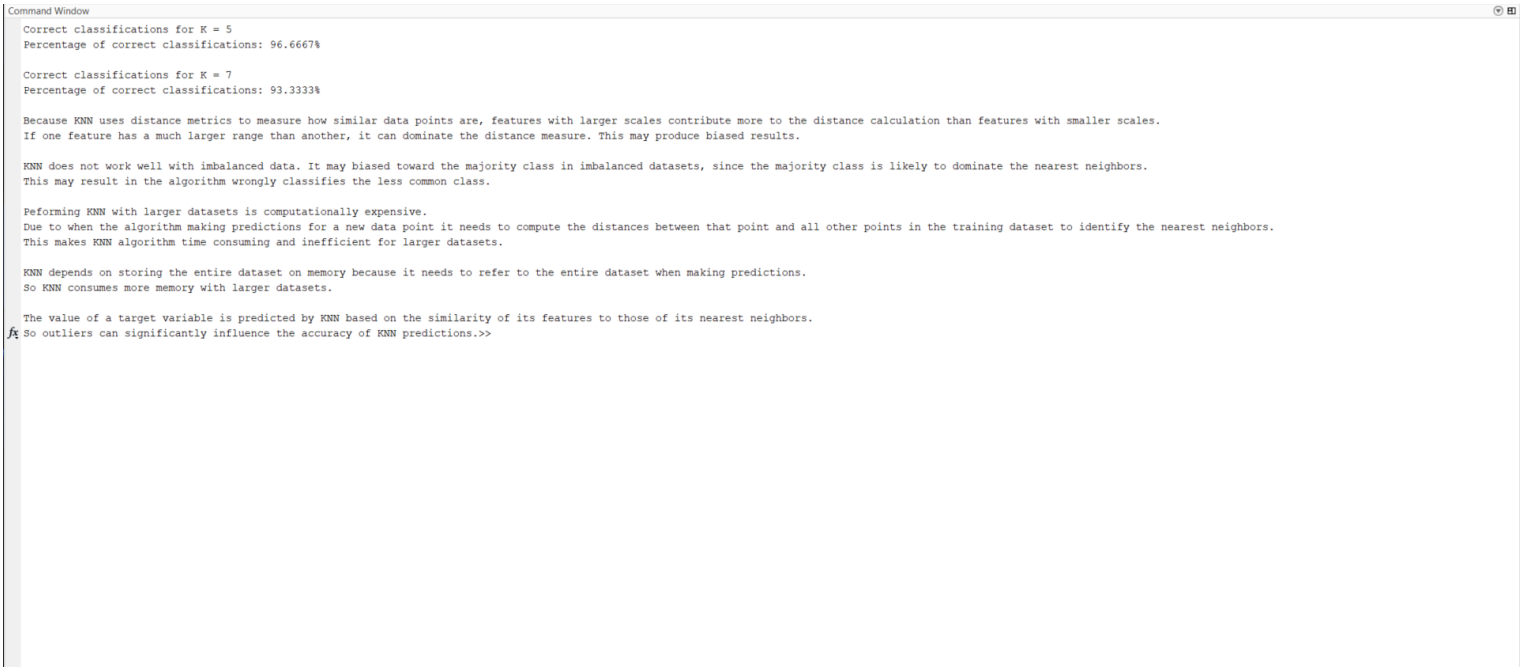

Figure 14 KNN code output

Output for Task 2.1 Dataset Statistics

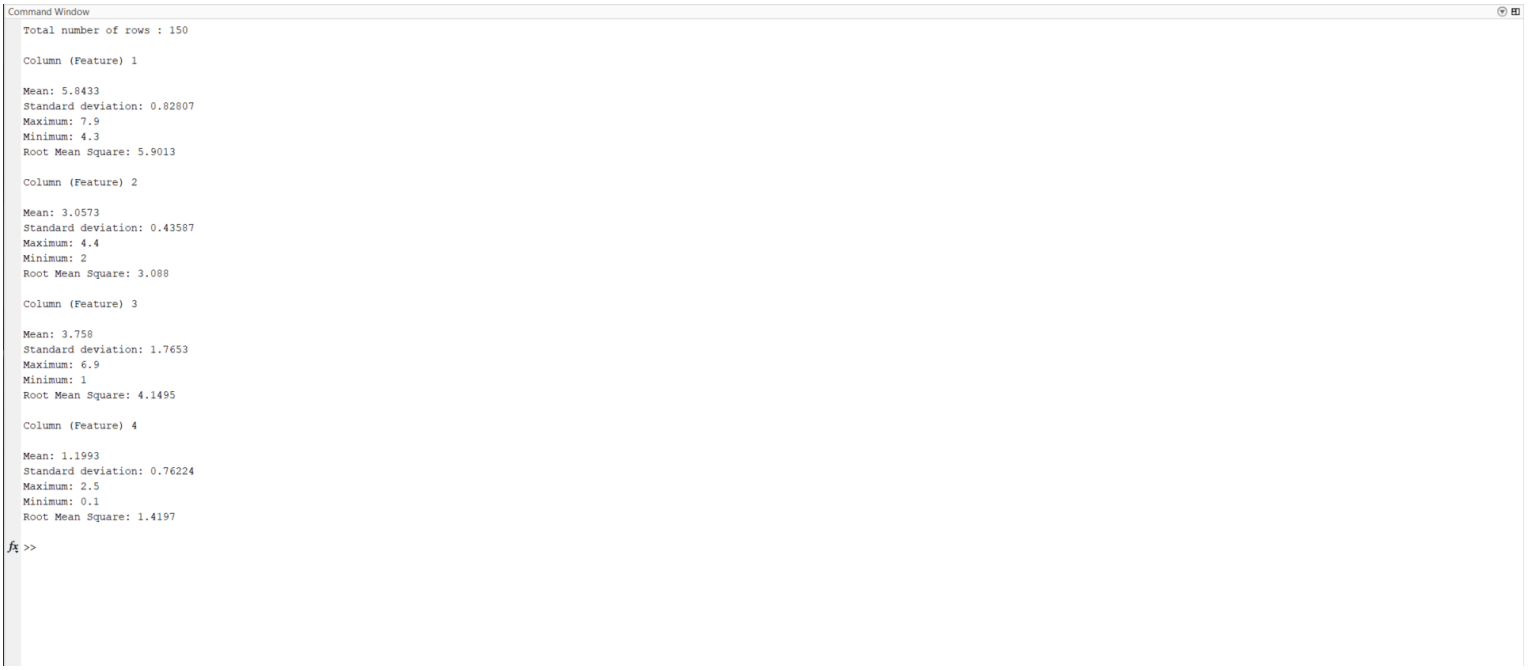

Figure 15 Dataset statistics code output

CHAPTER 06: APPENDIX

GitHub Repository for the Project : https://github.com/methjaya/AI-Project-Final-Year

Task 2.1 – Data statistics

```
% ----TASK 2.1 A----
% Loading the dataset
load('fisheriris.mat');

% ----TASK 2.1 B----
% Displaying the total number of rows
total_rows = size(meas, 1);
disp(['Total number of rows : ' num2str(total_rows)]);
disp(' ');

features = size(meas,2);

% ----TASK 2.1 C----
% Statistics for each column
for i = 1:features

    feature = meas(:, i);

    % Column
    disp(['Column (Feature) ' num2str(i)]);
    disp(' ');

    % Mean
    disp(['Mean: ' num2str(mean(feature))]);

    % Standard deviation
    disp(['Standard deviation: ' num2str(std(feature))]);

    % Max
    disp(['Maximum: ' num2str(max(feature))]);

    % Min
    disp(['Minimum: ' num2str(min(feature))]);

    % Root Mean Square
    disp(['Root Mean Square: ' num2str(rms(feature))]);

    disp(' ');
end
```

Task 2.2 – Neural Network Setup

```
load fisheriris

dataset = meas;

% ----TASK 2.2 (1)----
% Randomly shuffling the dataset
rand_indices = randperm(size(dataset, 1));

% Initializing percentage for training
training_percentage = 0.6;
training_size = round(training_percentage * size(dataset, 1));

% Dividing the dataset into training and testing
training_dataset = dataset(rand_indices(1:training_size), :);
testing_dataset = dataset(rand_indices(training_size+1:end), :);

% Creating training and testing targets
training_target = double(categorical(species(rand_indices(1:training_size))));
testing_target = double(categorical(species(rand_indices(training_size+1:end))));


% Assigning hidden layer sizes and number of experiments
hidden_layer_sizes = [10, 15, 20];
experiments_count = 3;

% ----TASK 2.2 (3)----
% Outer loop for hidden layer sizes
for l = 1:length(hidden_layer_sizes)
    hidden_layer_size = hidden_layer_sizes(l);

    accuracy_sum = 0;

    % ----TASK 2.2 (2)----
    % Initializing the feedforward neural network with hidden layer size
    net = feedforwardnet(hidden_layer_size);

    % Inner loop for all the experiments
    for e = 1:experiments_count

        % ----TASK 2.2 (4)----
        % Training the neural network with training data and target
        net = train(net, training_dataset', training_target');
        % View trained net
        view(net);

        % ----TASK 2.2 (5)----
        % Getting the predictions from the trained net
        predictions = net(testing_dataset');
        predicted_labels = round(predictions);
        % Calculating the classifier accuracy
        classifier_accuracy = (sum(predicted_labels == testing_target') /
length(testing_target)) * 100;

        % Displaying the performances
        disp(['Hidden layer size ' num2str(hidden_layer_size) ' Experiment '
num2str(e)]);
```

```
        disp(['Accuracy: ' num2str(classifier_accuracy)]);
        disp(' ');

        accuracy_sum = accuracy_sum + classifier_accuracy;
    end

    % ----TASK 2.2 (5)----
    % Calculating average performance
    disp(['Average percentage of correct classifications with '
num2str(hidden_layer_size) ' hidden layers and ' num2str(experiments_count) '
experiments : ' num2str(accuracy_sum/experiments_count) '%'])
    disp(' ');
    disp(' ');
end
```

Task 2.3 – K Means

```
% ----TASK 2.3 (1)----
% Load Kmeans data
load('kmeansdata.mat');

% Assign data
data_set = X;

% Clusters
clusters = [3, 4, 5];

% Initializing variables to store results
mean_silhouette_values = zeros(length(clusters), 1);
colors = [1 0 0;0 1 0;0 0 1;1 0 1;0.9290 0.6940 0.1250];
silhouette_figure = figure;

% ----TASK 2.3 (2)----
% k-means clustering for different number of clusters
for i = 1:length(clusters)
    k = clusters(i);

    % k-means clustering
    [idx, centroids] = kmeans(data_set, k);

    % ----TASK 2.3 (3)----
    % Plot silhoutte for each cluster
    figure(silhouette_figure);
    subplot(2,2,i);
    [silhouette_values,h]=silhouette(data_set, idx);
    title(['Number of clusters = ' int2str(k)]);
    xlabel 'Silhouette Value ';
    ylabel 'Cluster';
    hold on;

    % Store mean silhouette value
    mean_silhouette_values(i) = mean(silhouette_values);

    % ----TASK 2.3 (3)----
    % Display mean silhoutte values for each cluster
    disp(['Cluster ' num2str(i) ' Mean silhoutte value : '
num2str(mean_silhouette_values(i))]);

    % ----TASK 2.3 (4)----
    % Plot clusters and centroids
    figure;
    gscatter(data_set(:, 1), data_set(:, 2), idx,colors,'.', 8);
    hold on;
    plot(centroids(:, 1), centroids(:, 2), 'kx', 'MarkerSize', 15, 'LineWidth',
3);
    title(['K-Means Clustering with K = ' num2str(k)]);

    % Adding the legend for different clusters
    if k==3
        legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids','Location',
'NW');
    elseif k==4
        legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster
4','Centroids','Location', 'NW');
```

```
    elseif k==5
        legend('Cluster 1', 'Cluster 2', 'Cluster 3','Cluster 4','Cluster 5',
'Centroids','Location', 'NW');
    end
    hold off;
end
disp(' ');

% ----TASK 2.3 (4)----
% Stopping criteria for Kmeans
disp('When there are no changes in cluster centroids between iterations.');
disp('When there are no changes in cluster assignments between iterations.');
disp('When the algorithm reaches the predefined maximum iterations.');


% ----TASK 2.3 (5)----
% Best number of clusters
[best_silhouette, best_cluster_index] = max(mean_silhouette_values);
best_cluster = clusters(best_cluster_index);
disp(['The best number of clusters : ' num2str(best_cluster)]);
disp(['Mean silhouette value for the best cluster: ' num2str(best_silhouette)]);
disp(' ');

% ----TASK 2.3 (5)----
% Explanation for the best number of clusters
fprintf(['In k-means clustering, the silhoutte is used to measure the
effectiveness of the clustering.\n' ...
    'A high silhouette value indicates that the object is well matched to its own
cluster.\n' ...
    'Because of that this code calculates the mean silhouette value for all the
clusters and selects the cluster with maximum mean silhouette value as the best
number of clusters.\n' ...
    'The best number of clusters are ' num2str(best_cluster) ' because the mean
silhoutte value for this cluster is ' num2str(best_silhouette) '\n' ...
    'which is the highest and indicates that the objects are well separated
compared to other clusters.\n']);
fprintf('\n');


% ----TASK 2.3 (6)----
% Limitations/Drawbacks of k-means clustering
fprintf('1. One disadvantage of the K-means algorithm is that we must specify the
number of clusters in advance.\nChoosing the wrong number of clusters may lead to
inaccurate results.\n\n');
fprintf('2. Another disadvantage is K-means algorithm cannot handle non numerical
or categorical data.\nK-means algorithm works only with numerical data to
determine distances between data points.\nIf categorical data needs to be used it
has to be converted to numerical data, which can be difficult or not possible with
every dataset.\n\n');
fprintf('3. K-means clustering is sensitive to the initial centroid
placements.\nThe final clusters are formed by starting with randomly placing
centroids and iteratively updating the centroids, reassigning data points based on
their proximity to the centroids.\nDifferent initializations may lead to different
cluster outcomes and makes the K-means algorithm less reliable and
reproducible.\n\n');
fprintf('4. When using high dimentional datasets with k-means clustering, the
distance between any two data points gets more similar \nand makes it difficult to
identify meaningful patterns or relationships in the data as the number of
dimensions increases.\n\n');
```

```
fprintf('5. Since the complexity of k-means algorithm is determined by number of
clusters, the number of data points, and the number of dimentions, \nwhen using
high dimentional datasets makes it computationally expensive and time-consuming
because it requires multiple iterations and distance calculations.\n\n');
fprintf('6. K-means is sensitive to outliers. Outliers may drag the centroids, or
they may form their own cluster rather than being ignored. \nBecause of this a
single outlier can drastically affect the locations of centroids and distort the
resulting clusters.');
```

Task 2.4 – Implementation of KNN

```
% ----TASK 2.4 (1)----
load fisheriris


% ----TASK 2.4 (2)----
% Shuffling and taking 60% for training and 40% for testing from each
% species equally

% Assign species to a categorical array to derive different categories(species)
species_categorical = categorical(species);

% Get each category of species
unique_species = categories(species_categorical);

% Initialize variables to store training and testing data
training_dataset = zeros(length(species) * 0.6, 4); training_target =
categorical();
testing_dataset = zeros(length(species) * 0.4, 4); testing_target = categorical();

% Data amount for each category
trn_percentage = (length(species) / length(unique_species)) * 0.6;
tst_percentage = (length(species) / length(unique_species)) * 0.4;

% Initializing variables to hold indices
training_data_indices = 1:1:trn_percentage;
testing_data_indices = 1:1:tst_percentage;

for i = 1 : length(unique_species)
    % Get indices of each species
    indcs = find(species_categorical == unique_species{i});

    % Shuffle the indices
    indcs = indcs(randperm(length(indcs)));

    %  Allocating 60% for training
    training_dataset(training_data_indices,:) =
[meas(indcs(1:round(length(indcs)*0.6)),:)];
    training_target(training_data_indices,:) =
[species_categorical(indcs(1:round(length(indcs)*0.6)),:)];

     % Allocating 40% for testing
    testing_dataset(testing_data_indices,:) =
[meas(indcs(1+round(length(indcs)*0.6):end),:)];
    testing_target(testing_data_indices,:) = [
species_categorical(indcs(1+round(length(indcs)*0.6):end),:)];

    % Using if statement to prevent unnecessary increment of the indices in the
last iteration
    if i <= length(unique_species)-1
    % Incrementing the indices
    training_data_indices = training_data_indices  + trn_percentage;
    testing_data_indices = testing_data_indices + tst_percentage;
    end
end
```

```
% Assigning K values
k_values = [5 7];


% Performing k-NN for all the K values using a loop
for i = 1:length(k_values)

    k_value = k_values(i);

    % ----TASK 2.4 (3)----
    % Training the k-NN classifier using fitcknn
    knn_mdl = fitcknn(training_dataset, training_target, 'NumNeighbors',
k_value,'Standardize',1);
    % Predicting using testing data
    predicted_classifications = predict(knn_mdl, testing_dataset);

    % ----TASK 2.4 (4)----
    % Displaying the confusion matrix
    figure;
    confusion_chart = confusionchart(testing_target, predicted_classifications);
    confusion_chart.RowSummary = 'row-normalized';
    confusion_chart.ColumnSummary = 'column-normalized';
    confusion_chart.Title = ['Confusion chart for K = ' num2str(k_value)];

    % Displaying the percentage of correct classifications
    disp(['Correct classifications for K = ' num2str(k_value)]);
    disp(['Percentage of correct classifications: ' num2str((sum(testing_target ==
predicted_classifications) / length(testing_target)) * 100) '%']);
    disp(' ');

end

% ----TASK 2.4 (5)----
% Limitations/Drawbacks of KNN
fprintf('Because KNN uses distance metrics to measure how similar data points are,
features with larger scales contribute more to the distance calculation than
features with smaller scales. \nIf one feature has a much larger range than
another, it can dominate the distance measure. This may
produce biased results.\n\n')
fprintf('KNN does not work well with imbalanced data. It may biased toward the
majority class in imbalanced datasets, since the majority class is likely to
dominate the nearest neighbors. \nThis may result in the algorithm wrongly
classifies the less common class.\n\n')
fprintf('Peforming KNN with larger datasets is computationally expensive. \nDue to
when the algorithm making predictions for a new data point it needs to compute the
distances between that point and all other points in the training dataset to
identify the nearest neighbors. \nThis makes KNN algorithm time consuming and
inefficient for larger datasets.\n\n')
fprintf('KNN depends on storing the entire dataset on memory because it needs to
refer to the entire dataset when making predictions. \nSo KNN consumes more memory
with larger datasets.\n\n')
fprintf('The value of a target variable is predicted by KNN based on the
similarity of its features to those of its nearest neighbors. \nSo outliers can
significantly influence the accuracy of KNN predictions.')
```

CHAPTER 07: REFERENCES

AbdelRaouf, H., Chelloug, S.A., Muthanna, A., Semary, N., Amin, K. and Ibrahim, M., 2023. Efficient Convolutional Neural Network-Based Keystroke Dynamics for Boosting User Authentication. *Sensors (Basel, Switzerland)*, 23(10). https://doi.org/10.3390/s23104898.

Aldrich Jr, B., Liu, Y., Almousa, M. and Anwar, M., 2023. *Translating Keystroke and Mouse Dynamics Data to Classify Human Mood*.

Alsuhibany, S.A. and Almuqbil, A.S., 2021. Analyzing the Effectiveness of Touch Keystroke Dynamic Authentication for the Arabic Language. *Wireless Communications and Mobile Computing*, 2021, pp.1–15. https://doi.org/10.1155/2021/9963129.

Gautam, P. and Dawadi, P.R., 2017. Keystroke biometric system for touch screen text input on android devices optimization of equal error rate based on medians vector proximity. In: *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. pp.1–7. https://doi.org/10.1109/SKIMA.2017.8294136.

Kasprowski, P., Borowska, Z. and Harezlak, K., 2022. Biometric Identification Based on Keystroke Dynamics. *Sensors (Basel, Switzerland)*, 22(9). https://doi.org/10.3390/s22093158.

Lim, S., Kim, C., Cho, B.H., Choi, S.-H., Lee, H. and Jang, D.P., 2023. Investigation of daily patterns for smartphone keystroke dynamics based on loneliness and social isolation. *Biomedical Engineering Letters*. [online] https://doi.org/10.1007/s13534-023-00337-0.

Liu, W.-M., Yeh, C.-L., Chen, P.-W., Lin, C.-W. and Liu, A.-B., 2023. Keystroke Biometrics as a Tool for the Early Diagnosis and Clinical Assessment of Parkinson's Disease. *Diagnostics*, 13, p.3061. https://doi.org/10.3390/diagnostics13193061.

Shadman, R., Wahab, A.A., Manno, M., Lukaszewski, M., Hou, D. and Hussain, F., 2023. Keystroke Dynamics: Concepts, Techniques, and Applications. https://doi.org/10.48550/arXiv.2303.04605

Whiskerd, N., Körtge, N., Jürgens, K., Lamshöft, K., Ezennaya-Gomez, S., Vielhauer, C., Dittmann, J. and Hildebrandt, M., 2020. Keystroke biometrics in the encrypted domain: a first study on search suggestion functions of web search engines. *EURASIP Journal on Information Security*, [online] 2020(1), p.2. https://doi.org/10.1186/s13635-020-0100-8.

Yaacob, M.N., Idrus, S.Z.S., Ali, W.N.A.W., Mustafa, W.A., Jamlos, M.A. and Wahab, M.H.A., 2020. A Review on Feature Extraction in Keystroke Dynamics. *Journal of Physics: Conference Series*, 1529(2), p.022088. https://doi.org/10.1088/1742-6596/1529/2/022088.