

**ANALYZING AND IMPLEMENTING FEATURE SELECTION  
METHOD FOR PREDICTING PATIENT WAITING TIME IN  
CLINICS**

**SITI NOORAZIQA BINTI ONN**



**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**ANALYZING AND IMPLEMENTING FEATURE  
SELECTION METHOD FOR PREDICTING PATIENT  
WAITING TIME IN CLINICS**

**SITI NOORAZIQA BINTI ONN**

**This report is submitted in partial fulfilment of the requirements  
for the degree of Bachelor of Electronic Engineering with Honors**



اونيورسيتي تيكنيكل مليسيا ملاك

**Faculty of Electronics and Computer Technology and  
Engineering  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA  
Universiti Teknikal Malaysia Melaka**

**2024**

BORANG PENGESAHAN STATUS LAPORAN  
PROJEK SARJANA MUDA II

Tajuk Projek : Analyzing and implementing feature selection method for predicting patient waiting time in clinics.

Sesi Pengajian : 2022/2023

Saya SITI NOORAZIQA BINTI ONN mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

**SULIT\***

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

**TERHAD\***

(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.)

**TIDAK TERHAD**

Disahkan oleh:



**Ts. Dr. NOOR ASYIKIN BINTI SULAIMAN**  
Pensyarah Kanan

(Guaniti Dan Teknologi Dan Kejuruteraan Elektronik Dan Komputer)  
Universiti Teknikal Malaysia Melaka (UTeM)

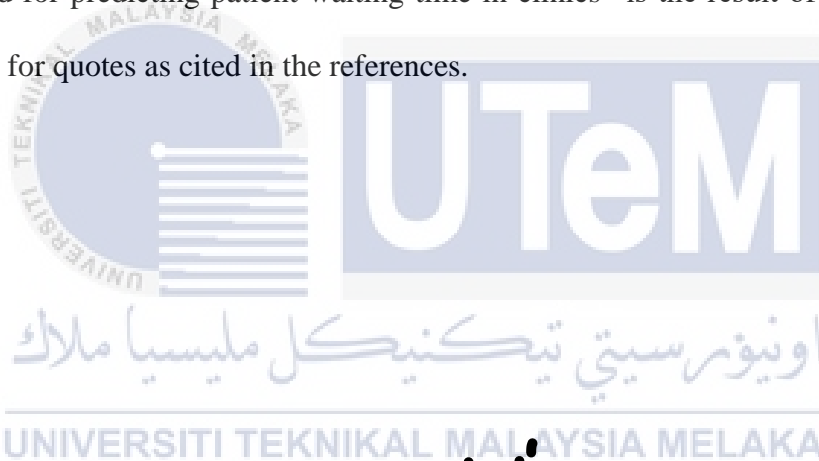
  
(TANDATANGAN PENULIS)

Alamat Tetap: NO.7, JALAN SEROJA 10, TAMAN SEROJA, BANDAR BARU SALAK TINGGI, 43900 SEPANG, SELANGOR

Tarikh : 10 JANUARY 2024 Tarikh : 22 JANUARY 2024

## DECLARATION

I declare that this report entitled “Analyzing and implementing feature selection method for predicting patient waiting time in clinics” is the result of my own work except for quotes as cited in the references.



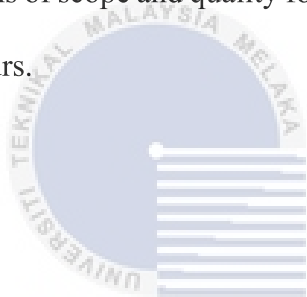
Signature : ..........

Author : .....SITI NOORAZIQA BINTI ONN.....

Date : .....10 JANUARI 2024.....

## APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.



اونيور سيتى تېكنيكل مليسيا ملاك

Signature :  .....

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Supervisor Name : NOOR ASYIKIN BINTI SULAIMAN .....

Date : 22 JANUARY 2024 .....

## DEDICATION

I dedicate this project to my supervisors, family, and friends whose guidance and expertise have shaped my academic growth and inspired me to pursue excellence.



## ABSTRACT

Queuing is an unavoidable aspect of life, and not knowing how long the wait will be leads to a major source of anxiety. Few companies frequently try to forecast the waiting times problem but none of them are successful. On the other hand, length of waiting time in queue is basically major problem patients need to withstand in hospitals and clinics. It leads to unaffected implementation of medical attention and quality care to those really in needs. In queuing system, advanced approaches for instance machine learning and deep learning have played an important role. This project aims to identify the important features in predicting patients waiting time in clinics. In choosing the important features, preprocessing data is accomplished first which involve preparing raw data and constructing the data to be competent to machine learning. After that, the data will be retrieved for feature selection with different feature selections method. In this project, implementation of correlation-based feature selection, Recursive Feature elimination and Sequential Feature Selection are developed using the same dataset. Thus, to analyse the accuracy of selected features, predictive analysis is implemented, and the performance of the predicted model is analysed using primary features and selected features.

## ABSTRAK

Masa menunggu adalah aspek yang tidak dapat dielakkan dalam kehidupan, dan tidak tahu berapa lama tungguan akan menjadi punca utama kegelisahan. Beberapa syarikat sering mencuba meramalkan masalah masa menunggu tetapi tiada yang berjaya. Di sisi lain, panjang masa menunggu dalam barisan adalah masalah utama yang pesakit perlu hadapi di hospital dan klinik. Ia membawa kepada pelaksanaan rawatan perubatan dan penjagaan berkualiti yang tidak terjejas kepada mereka yang benar-benar memerlukannya. Sistem beratur dalam waktu menunggu merupakan satu pendekatan canggih seperti machine learning dan pembelajaran mendalam telah memainkan peranan penting. Projek ini bertujuan untuk mengenal pasti ciri-ciri penting dalam meramalkan masa menunggu pesakit di klinik. Dalam pemilihan ciri penting, pra pemprosesan data dilaksanakan terlebih dahulu yang melibatkan penyediaan data primer dan membina data untuk kompeten kepada machine learning. Selepas itu, data akan diambil untuk pemilihan ciri dengan kaedah pemilihan ciri yang berbeza. Dalam projek ini, pelaksanaan pemilihan ciri berasaskan korelasi, eliminasi ciri berulang, dan Pemilihan Ciri Berseorangan dikembangkan menggunakan dataset yang sama. Oleh itu, untuk menganalisis ketepatan ciri yang dipilih, analisis prediktif dilaksanakan, dan prestasi model yang diramalkan dianalisis menggunakan ciri utama dan ciri yang dipilih.



## ACKNOWLEDGEMENTS

First and foremost, I would like to thank my outrageous esteemed supervisor Ts. Dr. NOOR ASYIKIN BINTI SULAIMAN for her invaluable supervision, support, and tutelage during the Bachelor's Degree Project 1 and Project 2 course. I also would like to thank her for initiating such an interesting project and providing me with definite direction, professional guidance, constant encouragement, and moral support in the process of the integrated design project.

My gratitude extends to the Faculty of Electronic Engineering and Computer Engineering for the funding opportunity to undertake our studies at the Department of Electronic Engineering, University Technical Malaysia Melaka. I would like to thank a few other friends, classmates, and research team for a cherished time spent together to complete this designed project, and in social settings. My appreciation also goes out to my family and friends for their encouragement and support all through my studies.

## TABLE OF CONTENT

<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>i</b>
<b>Dedication</b>	<b>i</b>
<b>Abstract</b>	<b>i</b>
<b>Abstrak</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>table of content</b>	<b>iv</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>xi</b>
<b>LIST OF APPENDICES</b>	<b>xii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Project Background	2

1.2	Problem statement	2
1.3	Project objectives	3
1.4	Scope of Project	3
1.5	Project Significant	4
<b>CHAPTER 2 BACKGROUND STUDY</b>		<b>6</b>
2.1	Quality Care and Services	7
2.2	Feature selection	8
2.2.1	Correlation-based feature selection	12
2.2.2	Recursive Feature Elimination	13
2.2.3	Sequential Feature Selection	15
2.2.4	Decision Tree predictive model.	17
2.2.5	Logistic regression predictive model.	18
<b>CHAPTER 3 METHODOLOGY</b>		<b>20</b>
3.1	Overview of the project	21
3.2	Project Implementation	23
3.3	Preprocessing Data	23
3.4	Split data into training and testing	29
3.5	Dataset Shape in Python	30
3.6	Develop features selection.	32
3.6.1	Correlation-based feature selection.	32

3.6.2	Recursive Feature Elimination.	34
3.6.3	Sequential Feature Selection	38
3.7	Develop predictive model.	41
3.7.1	Decision Tree Model	41
3.7.2	Logistic Regression model.	42
3.7.3	Comprehensive Analysis	43
3.8	Development of Web application for waiting time prediction	45
<b>CHAPTER 4 RESULT AND DISCUSSION</b>		<b>48</b>
4.1	Feature Selection	49
4.1.1	Correlation-based Feature Selection	49
4.1.2	Recursive Feature Elimination	51
4.1.3	Sequential Feature Selection	55
4.2	Analyze the accuracy performance of the feature selection method.	59
4.3	Develop predictive model.	61
4.3.1	Decision Tree Model	62
4.3.2	Logistic Regression Model	64
4.3.3	Comprehensive Analysis	66
4.4	Waiting Time Prediction using Developed Web Application Interface.	69
4.5	Testing web application to predict Waiting Time.	70
4.6	Environmental and Sustainability.	73

4.7	Chapter summary.	74
<b>CHAPTER 5 conclusion and future works</b>		<b>76</b>
5.1	Conclusion.	77
5.2	Future works.	78
<b>REFERENCES</b>		<b>79</b>
<b>APPENDICES</b>		<b>84</b>



## LIST OF FIGURES

Figure 3-1: Flowchart of the system design.....	22
Figure 3-2:Original data allocated from Klinik Kesihatan Alor Gajah .....	27
Figure 3-3: The pre-processing data. ....	28
Figure 3-4: The final Dataset after pre-processing data and data cleaning.....	28
Figure 3-5 : Splitting dataset into training and testing files.....	29
Figure 3-6: Importing important libraries into the Jupyter Notebook. ....	30
Figure 3-7: Dataset value.....	31
Figure 3-8: Data description .....	32
Figure 3-9: Coding implementation for correlation-based feature selection. ....	33
Figure 3-10: Loading the dataset into the workspace. ....	34
Figure 3-11: Training and testing the dataset. ....	35
Figure 3-12: The development of the recursive feature elimination.....	36
Figure 3-13: Applying Gradient Boosting Classifier algorithm into the data set. ....	37
Figure 3-14: Recursive feature ranking applied to the Data set.....	37
Figure 3-15: Preparation of Dataset to develop sequential feature selection.....	38
Figure 3-16: Applying Standard Scaler algorithm into the data. ....	39

Figure 3-17: Implementation of Sequential feature selector into the data set. ....	40
Figure 3-18: Establish metrics computation into the model. ....	40
Figure 3-19: Development of decision tree predictive model.....	42
Figure 3-20: Development of the Logistic Regression predictive model. ....	42
Figure 3-21: Implementation of comprehensive analysis of testing dataset. ....	44
Figure 3-22: Implementation of comprehensive analysis of training dataset. ....	44
Figure 3-23: Convert prediction model into Pickle file. ....	45
Figure 3-24: Embedded function from pickle file into VS Code.....	47
Figure 3-25: Creating web page layout.....	47
Figure 4-1: Number of selected features using correlation-based feature selection. .	49
Figure 4-2: Correlation Matrix Heatmap .....	50
Figure 4-3: Accuracy of the correlation-based model. ....	51
Figure 4-4: Recursive Feature Elimination command .....	52
Figure 4-5: Accuracy of the Recursive Feature Elimination model. ....	53
Figure 4-6: Correlation matrix heatmap .....	54
Figure 4-7: Number of selected features using sequential feature selection.....	55
Figure 4-8: The accuracy of the sequential model.....	56
Figure 4-9: Feature index of the sequential model. ....	57
Figure 4-10: The performance of the selected features.....	58
Figure 4-11: Performance accuracy of different feature selection methods. ....	59
Figure 4-12: Accuracy of the Decision tree model.....	63
Figure 4-13: Accuracy of the Logistic regression predictive model.....	65
Figure 4-14: Comprehensive analysis in logistic regression predictive model. ....	66

Figure 4-15: Comprehensive analysis in decision tree predictive model. ....	67
Figure 4-16: Web Application Interface. ....	70
Figure 4-17: Example input data Class 1 .....	71
Figure 4-18: Example input data Class 2 .....	72
Figure 4-19: Example input data Class 3 .....	72





## LIST OF TABLES

Table 3-1: Data cleaning .....	24
Table 3-3: List of description.....	31
Table 3-4: Library function of web application.....	46
Table 4-1: Statistical algorithm applied to the selected features. ....	57
Table 4-2: Analyzing selected features with different feature selection methods. ....	61
Table 4-3: Performance comparison between primary and selected features.....	63
Table 4-4: Performance comparison between primary and selected features.....	65
Table 4-5: Comparison of the comprehensive analysis. ....	68
Table 4-6: The predicted output using decision tree model.....	71

## LIST OF APPENDICES

Appendix A: Python Code to Build Logistic Regression Model.....	84
Appendix B: Python Code to Build Decision Tree Model. ....	85
Appendix C: Python Code to Develop Web Application in JavaScript. ....	85
Appendix D: Python Code to Develop Web Application in HTML. ....	86



# CHAPTER 1

## INTRODUCTION



This chapter will discuss more in detail the project in the general background, problem statement, objectives, the scope of the project and project significant.

## 1.1 Project Background

Queuing is an unavoidable aspect of life, and not knowing how long the wait will be leads to a major source of anxiety. A few companies frequently try to forecast the waiting times problem but none of them are successful. On the other hand, the length of waiting time in queue is basically a major problem patients need to withstand in hospitals and clinics. It leads to unaffected implementation of medical attention and quality care to those really in need. In queuing systems, advanced approaches, for instance machine learning and deep learning have played an important role. This project aims to identify the important features in predicting patients waiting time in clinics. In choosing the important features, preprocessing data is accomplished first which involves preparing raw data and constructing the data to be competent to machine learning. After that, the data will be retrieved for feature selection by applying numerous methods. In this project, Correlation-based feature selection, Recursive Feature Elimination and Sequential Feature Selection are developed. Furthermore, to analyze the accuracy of selected features, a prediction model is utilized using the primary features and selected features.

## 1.2 Problem statement

Basically, healthcare systems aim to deliver efficient, equitable, high-quality care in a timely manner to patients but many service-oriented businesses, including health clinics, are involved in queues, or waiting lines. Continuous waiting lines could lead to frustrating and stressful experiences for the patients. Plus, many health facilities are still using manual registration that includes varieties of personal information and it is not obliterated with other health facilities which appointment books are contemporary. Consequently, it enumerates more to the patient's waiting

time. Thus, prediction waiting times are developed with varieties of techniques. Nevertheless, there is not a pound used to identify the selected features according to the developed prediction waiting time. Commonly, the developed prediction model uses insignificant features and causes consuming time to predict the waiting time. Hence, this project aims to develop the important features in predicting patients waiting time in clinics.

Machine learning algorithms and models utilize data in different ways. Any machine learning models have a set of properties that allow it to compare the data in a variety of ways. Basically, the primary goal of model comparison and selection is to improve the efficiency and accuracy of machine learning software or solutions. With allocated enormous features at hand, it is easy to focus on the features that can provide fast processing and higher efficiency. Nonetheless, if the selected features are too excessive, it could expedite bootless errands during patient's registration in fact if the selected features are incompetent, the prediction model could be inaccurate. Thus, this project aims to analyze the accuracy of selected features using logistic regression model and decision tree model. This is to avoid low accuracy and focus more on high accuracy prediction model.

### **1.3 Project objectives**

- To identify the important features in predicting patients waiting time in clinics.
- To analyze the accuracy of selected features using logistic regression and decision tree.

### **1.4 Scope of Project**

In this project, the study was examined at Klinik Kesihatan Alor Gajah in Melaka to develop a prediction waiting time from the varieties of data available in the

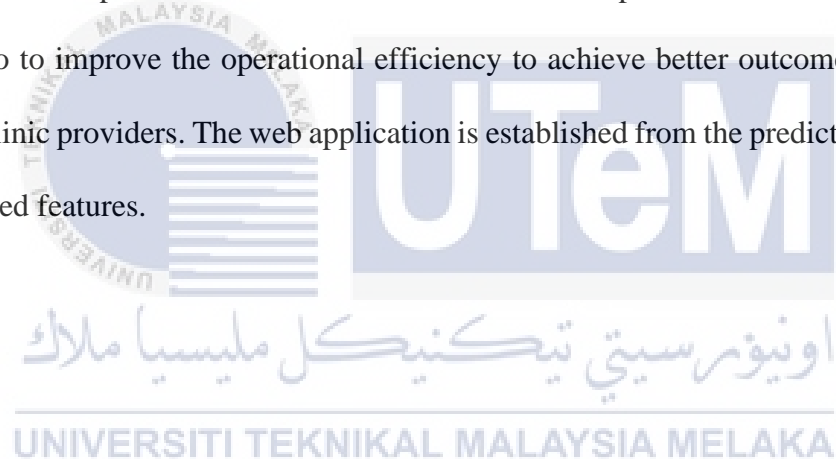
dental health facility information system. In order to ensure that not all of the features accessible within the dataset are equally essential, feature selection is developed which is a technique for removing significant characteristics from the dataset. This project will gather in Python programming with a machine learning library of the Python language. It is made up of multiple algorithms and supports professional libraries.

In machine learning, data pre-processing refers to the process of transforming raw data into a format that is suitable for a machine learning algorithm to learn from. The data and parameters obtained will be analyzed and filtered to seek the primary data using the prediction model which are logistic regression and decision tree model. As an example, for the data and parameters are personal information, age, type of treatment and gender. In the meantime, there are 12 features that have encountered the pre-processing data. Then, the features will be analyzed and compared the performance of the prediction model with primary features and selected features.

### 1.5 Project Significant

The aim of this project is to develop feature selection using several methods. In this project, correlation-based feature selection, recursive feature elimination and sequential feature selection are developed. This is because developing various feature selection is a process of selecting the attributes that will make the prediction more accurate and features with high correlation are more dependent. Other than that, different feature selection indicates different complexity and sensitivity. Predictive analysis is also used in the machine learning process to improve the accuracy of the feature selection. It additionally enhanced the algorithms' prediction potential by picking the most important variables and removing the redundant and unimportant ones.

This project also aims to identify the accuracy of the selected features selection using logistic regression and decision tree model. Logistic regression is a significantly simpler approach than other machine learning algorithms. It enhances the estimating approach and, more importantly, these strategies have simple equations and an easy-to-understand application on a modular level. On the other hand, Decision Tree model is beneficial because it allocates the importance of traits and it is straightforward to study, evaluate and comprehend. These prediction techniques prove that this project is well suited for estimating patient waiting times at the dental health facility. Furthermore, the web application interface is developed to provide estimated waiting times for the patients and to avoid frustration with unpredictable waiting times. This is also to improve the operational efficiency to achieve better outcomes for patients and clinic providers. The web application is established from the predicted model with selected features.



## CHAPTER 2

### BACKGROUND STUDY



This chapter describes the quality care and services in healthcare that happened in other healthcare too. This chapter also discusses the literature related to the quality care system that used machine learning to predict waiting time and projects that have developed feature selection. It is then followed by the type of feature selection used to improve the waiting time system.



## 2.1 Quality Care and Services

The term "waiting time" refers to the duration a patient spends in a healthcare facility from their arrival at the registration desk until their departure or completion of their last service. Specifically, it is the time interval between when a patient is added to a waiting list and the time spent waiting at each service point before receiving treatment. Lengthy waiting times can result in negative patient experiences, reduced patient satisfaction, and increased patient anxiety. As a result, decreasing waiting times is crucial to improving the quality of healthcare services [1].

Generally, quality care and services to patients are the goals of a health system. This is because health contributors aimed in providing quality healthcare services to encourage customer satisfaction. Patients who are satisfied with the service they receive are more inclined to return for further appointments or even to refer others to the same practitioner or facility. According to Hijry, Olawoyin, Edwards, McDonald, and Debnath [2] the problem of waiting time in healthcare institutions has been unequivocally ranked the top concern by patients, specifically in the outpatient department. Waiting times for patients presenting at health facilities are reported to be the worst compared with those experienced in general care and cold medical cases.

In a 2019 survey conducted by the Society of Actuaries, it was found that 60% of healthcare executives acknowledge the advantages of predictive analytics and have implemented it in their organization. Out of those who adopted this technology, 42% reported enhanced patient satisfaction, while 39% claimed to have reduced costs. Through continuous monitoring using machine learning and analysis of patient vital signs, predictive algorithms can identify patients who are likely to require intervention in the next hour. This enables healthcare providers to take proactive measures at an

early stage, based on early indicators of deterioration in the patient's condition. Predictive analytics can also estimate the likelihood of patient mortality or readmission within 48 hours post-ICU discharge, which can assist caregivers in deciding which patients can be safely discharged. This shows that, in the medical field, it offers improvement in the decision-making process by evolving predictive analytics. [3].

According to the National Program for improving access to Quality of Specialized Dental Care Centers, reducing inequality, and promoting equitable care delivery are among the primary objectives of access regulation in public health, which includes managing waiting times. Several studies have developed different methods and techniques to analyze and predict waiting times of the patients. A study to explore the factors associated with the waiting time for access to specialized dental care by using binary logistic regression model is obtained. The study found that shorter waiting times for specialized oral healthcare were linked to various factors, including the size of the service, the presence of a dedicated manager and integration with primary healthcare. These findings suggest feature selection that implementing regression model will improve access in specialized oral healthcare [4]

## **2.2 Feature selection**

For the purposes of this work, feature selection can alternately be referred to as variable selection or attribute selection. Brownlee (2016) [5] defines feature selection as the process of selecting a subset of relevant characteristics for use in model creation. The feature selection strategy allows in the development of reliable prediction models. It may be used to detect and remove unnecessary, irrelevant, and redundant features from data that do not add to or may even impair model accuracy.

The usage of fewer features reduces model complexity, and simpler models are easier to grasp and explain. According to a few research, many researchers are perplexed by the distinction between feature selection and feature extraction. The essential distinction between the two is that feature selection preserves a subset of the original characteristics while feature extraction develops entirely new ones. Both feature extraction and feature selection can improve efficiency, reduce computational complexity, develop better generalization models, and reduce storage requirements.

Algorithms for feature selection can be divided into three general classes which are filter method, wrapping methods, and embedded methods. Using an elimination approach, each feature is given a score based on statistical criteria. Ranking features according to scores determines whether they should be kept in the data collection or removed. The performance of the wrapping method is assessed using the pre-defined learning algorithm and returned to the feature search component for the subsequent round of feature subset selection. The final set will be determined by which feature set performs the best.

The Embedded technique evaluated regardless features were most important to the model's accuracy at the time the model was being created. The normalization approach is the most typical kind of embedded feature selection technique. In classification problems, supervised feature selection is usually implemented. Having access to the class labels allows supervised feature selection algorithms to choose discriminative features that effectively separate instances from various classes.

Other than that, a study of a high-performance malware detection system using feature selection methodologies is introduced by Esraa, Riyadh, Zaid and Husam [6]. The data encountered with the preprocessing and correlation-based feature selection is applied to generate various feature-selected datasets. The datasets are then utilized to train dense and LSTM-based deep learning models. In the case of the second dataset, the performance was impacted by the dimension reduction process due to its large number of attributes and a relatively small number of records. This high dimensionality made the correlation values between the classification column and other predictors very similar, resulting in the removal of more columns during the threshold-based feature selection process compared to the first dataset, which had fewer columns. The study found that reducing 81.77% of the total columns resulted in a 3.79% decrease in validation accuracy, while reducing 93.5% of the total columns resulted in a 9.44% reduction in validation accuracy. Thus, in all experiments, the results proved that using 20% as a test of feature selection was the best option.

Furthermore, Masoudi-Sobhanzadeh, Motieghader and Masoudi-Nejad [7] have described feature selection approaches that offers a user-friendly and uncomplicated method that can be applied in various types of research and is suitable for both balanced and unbalanced data. It utilizes multiple score functions, such as accuracy, sensitivity, and specificity that will enable efficient selection of relevant features. However, the software tools will also have drawbacks that lead to malfunctioning devices. This is because successful predictive analysis is crucial to have access to updated information. The accuracy of a prediction heavily relies on time, as data from the past may no longer be relevant in predicting current trends and patterns in the global market. Failure to consider this could result in significant financial losses for an organization.

Moreover, machine learning also introduced speech emotion recognition and recently it is widely utilized in human computer interaction. The recognition rate of multiple speech emotions can decrease due to increased emotional confusion. To address the issue, a proposed approach for speech emotion recognition involves utilizing a decision tree model with Fisher feature selection to identify the most effective emotional speech features and establish an accurate recognition model. The Fisher criterion is used during feature selection to filter out features with higher distinguishability.

In the emotion classification stage, an algorithm is developed to determine the structure of the decision tree. The decision tree enables the two-step classification of rough and fine classifications, eliminating redundant parameters and improving emotion recognition performance. Experimental results indicate that the proposed method, which utilizes a decision tree with feature selection strategy, achieves an 83.75% recognition rate for speech emotion recognition. This shows a 9% improvement compared to traditional decision tree and an 8.08% improvement compared to decision tree without feature selection. These findings confirm that Fisher feature selection by utilizing decision tree is effectively reduce emotional confusion and improve emotion recognition rates [8]

In 1999 Molodtsov developed a System Suitability Test Extension (SST) to address the problem of parameterization incompatibility, that approaches rough set and fuzzy set theories. There are instances of those that have failed to become effective parameterization tools. However, the limitations of the current parameterization techniques are dependent on a few variables, such as incorrect hardware and software choices. In addition to problems with rough sets and fuzzy sets, other parameterization

techniques are also having trouble evaluating the massive amounts of data that require a lot of memory and taking a long time to evaluate. For instance, implementation of fuzzy rough feature selection is based on fuzzy divergence measure. It requires lengthy processing time especially involving huge dataset.

On the other hand, feature selection of multi-label fuzzy method is developed to identify the precise samples from various classes based on the entire label space. It was also proposed for multi-label learning that leads to outperforming other cutting-edge algorithms and produce reliable upper and lower approximations of the datasets. When using more data than 5000 samples for each dataset, the multi-label fuzzy method performed better. This shows that feature selection has successfully improved the predictive model by making the process more accurate and efficient.

### 2.2.1 Correlation-based feature selection

In general, correlation-based feature selection is a technique used in machine learning and data science to identify the most relevant features in a dataset. Correlation-based feature selection works by calculating the correlation between each feature and the target variable and selecting the features with the highest correlation. Correlation-based feature selection is a filter approach for feature selection that is not reliant on the final classification model. It assesses feature subsets based solely on intrinsic data properties, specifically the correlations between features. The objective is to identify a feature subset with minimal feature-feature correlations to eliminate redundancy and maximum feature-class correlations to preserve or enhance predictive capability [9]. In machine learning, the performance of a model often depends on the quality and quantity of features used for training. However, using many features can lead to several issues, including overfitting, increased computational complexity, and

decreased model interpretability. Correlation-based Feature Selection helps to address these issues by selecting a subset of features that are most relevant for predicting the target variable.

By reducing the number of features, Correlation-based feature selection can improve the accuracy and efficiency of machine learning models. Fewer features mean that the model requires less computational resources to train and can make predictions faster. Moreover, using fewer features reduces the risk of overfitting, which occurs when a model is too complex and captures noise in the training data instead of the underlying patterns. Eliminating irrelevant or redundant information can also improve model interpretability, as it allows for a clearer understanding of the relationship between the features and the target variable. This can be particularly useful in fields where interpretability is important, such as healthcare or finance. Overall, Correlation-based feature selection is a powerful tool for improving the performance and interpretability of machine learning models by identifying the most informative and independent features for prediction [10].

### **2.2.2 Recursive Feature Elimination**

Selecting a subset of important features from a large original set of data in term of pre-defined criteria is known as feature selection. Feature selection plays an important role in machine learning, and it is also important in sample classification where the number of training samples is lower than the number of features. In 2022, Chen, Manongga and Dewi [11] have developed the recursive feature elimination for improving learning points on hand-sign recognition is developed to improve the accuracy of digit hand-sign detection. In the project three different datasets to train

models are used with different number of features meanwhile the fourth dataset is not used to train any model. As a result, the accuracy of the model in detecting hand signs is improved by removing the non-essential hand landmarks. As stated, fewer models trained with 10 features shows higher accuracy rather than using the original 21 features. Some hand landmark detections are distorted during the feature extraction stage when the images are rotated during image augmentation. When compared to Dataset 2, the data quality deteriorates, making the models less accurate. This shows that the recursive feature elimination successfully developed the accuracy of the hand-sign recognition.

Low performance of feature selection might be caused by high dimensional of an excessive number of features or known as overfitting dataset. Other than that, high dimensional datasets tend to create deficiency in space and need high computing power so the models could fit to low datasets classification accuracy. To achieve the greatest results on machine learning tasks as datasets grow, it is crucial to choose the ideal feature subset from the original dataset and an effective selection strategy must be utilized. A hybrid-recursive feature elimination method for efficient feature selection which combines with support vector machine and random forest are developed by. In this project, it uses the method of Hybrid Recursive Feature Elimination, Simple Sum, and Weighted Sum. The proposed method uses eight benchmarks, for instance satellite, messidor and sonar. As a result, the SVM-RFE produced the highest classification accuracy. This shows that the Hybrid-Recursive Feature Elimination has developed better performance to create the model.

Based on Yaoxin Wang, Yingjie Xu and Zhenyu Yang [12], Recursive Feature Selection with Random Forest are developed to improve the protein structural class



prediction for low similarity sequences. In the project, it also uses a sequence feature involving sequence content feature, sequence position feature, and reduced sequence feature. The results show that the efficiency of the protein structural class prediction has improved persuasively using the recursive feature selection with random forest. It also shows that the prediction accuracy improved by 4.6% to 13.3% with the use of less than 5% features. Other than that, the predicted secondary structural features using different protein features has achieved the best performance of protein features which 8% to 31% higher than other features. This shows that the protein features prediction has successfully achieved the improvement of protein structural class.

### 2.2.3 Sequential Feature Selection

A supervised method of feature selection is sequential feature selection. This method employs a supervised model and may be used to remove insignificant characteristics from a big dataset or to pick important features by progressively adding them. In accordance with Yulianti and Saifudin (2020) [13], feedback from customer churn is very important to the company when switching operators were made. It also states that higher costs are required to attract new customers than maintaining existing customers. Companies may take a quick step to retain clients by estimating customer turnover. The customer data may be analyzed using data mining techniques to develop predictions. This paper recommends using Sequential Feature Selection to choose significant features that might increase the performance in customer churn prediction models. In this paper, the use of feature selection was suggested to pick characteristics that are relevant and have a beneficial influence on prediction models. According to studies, models that use feature selection can deliver greater performance. The

suggested approach helps identify clients who are about to unsubscribe, allowing action to be taken to prevent it.

Furthermore, removing irrelevant features is traditionally in feature selection algorithm by reducing the redundancy between the significant and insignificant features. Clustering-based Feature selection are developed by approaches of pre-processing data, combined with search strategy and search strategy alternatives. As indicated by Alimoussa, Porebski, Vandenbroucke and Oulad Haj Thami (2021) [14], the proposed method for the high dimensional data classification is irrelevant feature removal, correlation-based feature clustering and Sequential Feature Selection. Initially, the algorithms are divided into the number of clusters which each of the features consist correlated features after it has removed the insignificant features. By selecting only applicable and irredundant features, it will help in speed up the search algorithm and acquired discriminant and compact features. Based on the paper, it also stated that the results were compared with three feature selection scheme and four clustering-based feature selection. The results showed that, in comparison with current filter model-based approaches, this paper indicates a high level of dimensionality reduction, adequate classification accuracy and equitable processing. This shows that the approaches of the sequential feature selection are successfully effective, and the result of the efficiency is meticulous.

Theoretically, Sequential Feature Selection is an algorithm that emphasis the features to enhance the predictive model and to reduce the model's computational complexity. However, by directly developing the sequential feature selection model, it might result in inaccurate prediction model. Therefore, there are methods to improve the performance of the Sequential Feature selection. Based on A. Suruliandi, G.

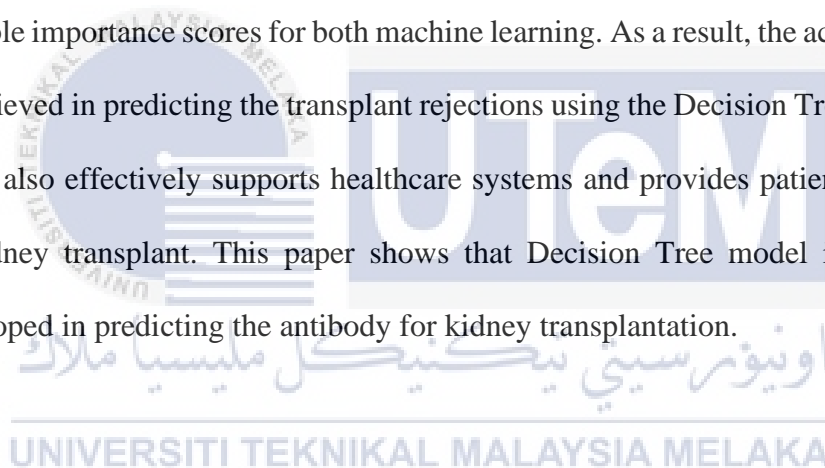
Mariammal and Raja (2021)[15], the paper proposes a general scheme for hybrid feature selection algorithm to improve the performance of the prediction model. The primary rationale for developing hybrid feature selection algorithms is the ability to use two distinct Feature Selection methods, each of which is beneficial in different circumstances. This paper demonstrates experimentally that by combining faster but weaker filter of Sequential Feature Selection criteria with slower but potentially more appropriate to wrapper Feature Selection criteria, the results will achieve a comparable to wrapper-based Feature Selection but in filter-like time. According to the paper, the relevance of the data on a scale of 0 to 1 is generally recognized. The lower values may be expected to provide more filter, while higher values yield more wrapper. Values around 0 allow for hybridized feature selection in higher-dimensional situations than values near 1.

#### 2.2.4 Decision Tree predictive model.

A paper titled Water quality classification using machine learning algorithms has discussed the use of machine learning classification algorithms for predicting water quality [16]. This article compares different machine learning classifiers and their ensemble models for the classification of water quality data according to the Water Quality Index (WQI). The subsequent sections discuss the proposed system's process, involving dataset explanation and manipulation as well as identifying various classifiers employed such as Support Vector Machine, Random Forests Logistic Regression Decision Tree, CATBoost, XGBoost and Multilayer Perceptron. Several metrics such as sensitivity, accuracy, confusion matrix, F1 score precision-recall curve, ROC curve and average precision were employed throughout these evaluations.

In conclusion, the paper presented CATBoost model has achieved a supreme individual classifier's highest accuracy percentage with 94.51%.

On the other hand, prediction in antibody incompatible kidney transplantation is published by the biomedical signal processing and control journal [17]. The paper proposed two methods of machine learning which are decision tree and random forest that forecast the result. In accordance with the paper, a small dataset is developed involving 80 patients with pre-transplant features. The features stated are human leukocyte antigen, donor specific and cytometry cross-transplant. As stated in the paper, the statistical methods implied are accuracy, sensitivity, ROC curve and variable importance scores for both machine learning. As a result, the accuracy of 85% is achieved in predicting the transplant rejections using the Decision Tree Model. The result also effectively supports healthcare systems and provides patients satisfaction in kidney transplant. This paper shows that Decision Tree model is successfully developed in predicting the antibody for kidney transplantation.



### **2.2.5 Logistic regression predictive model.**

A logistic regression predictive model is developed to improve the outcome after 6 months in patients with severe traumatic brain injury using physiological cerebral parameters. In accordance with Frank C. Bennis, Bibi Teeuwen and Frederick A. Zeiler (2020) [18], the prediction model is implemented to improve current models with continuous neuromonitoring data obtained from the intensive care unit neuromonitoring. Two teaching hospitals and 45 patients with intracranial pressure and cerebral perfusion pressures are monitored. The paper stated that 14 features with high frequency physiological were selected for instance Corticosteroid Randomization

after Significant Head Injury (CRASH) score, dynamic cerebral volume, cerebral compliance, and cerebrovascular pressure reactivity indices. The selected parameters will be trained using the logistic regression model to predict the outcome after 6 months and the output is validated using cross validation. As a result, the logistic regression model has an outcome of 0.76 probability in the area under the curve and the highest probability in the area under the curve 0.90 is achieved with additional 5 features that describe the mean arterial blood pressure and physiological cerebral indices. This shows that the prediction model can be improved by the addition features of neuromonitoring.

Moreover, encountering and recognizing the symptoms of heart disease is crucial in decision-making for patients health. Utilizing machine learning will lead to viable solutions to improve the accuracy of the heart disease diagnosis. In accordance with Mohammad Yaseliani and Majid Khedmati (2022) [19], the paper proposes a logistic regression model to predict heart disease while evaluating the impact of numerous predictors from the outcome. The model consists of 299 patients and 13 features. On the other hand, statistical analysis is also implemented for instance Akaike information criterion (AIC) and Receiver operating characteristic (ROC) to predict the outcome of the prediction model accurately before comparing the prediction model with another predictors. As a result, the sensitivity levels show an output of 84.21% while yielding at 90% and the total accuracy of 87.77%. In conclusion, the logistic regression model has effectively predicted the outcome and the method is compatible to diagnose the heart disease.

## CHAPTER 3

### METHODOLOGY



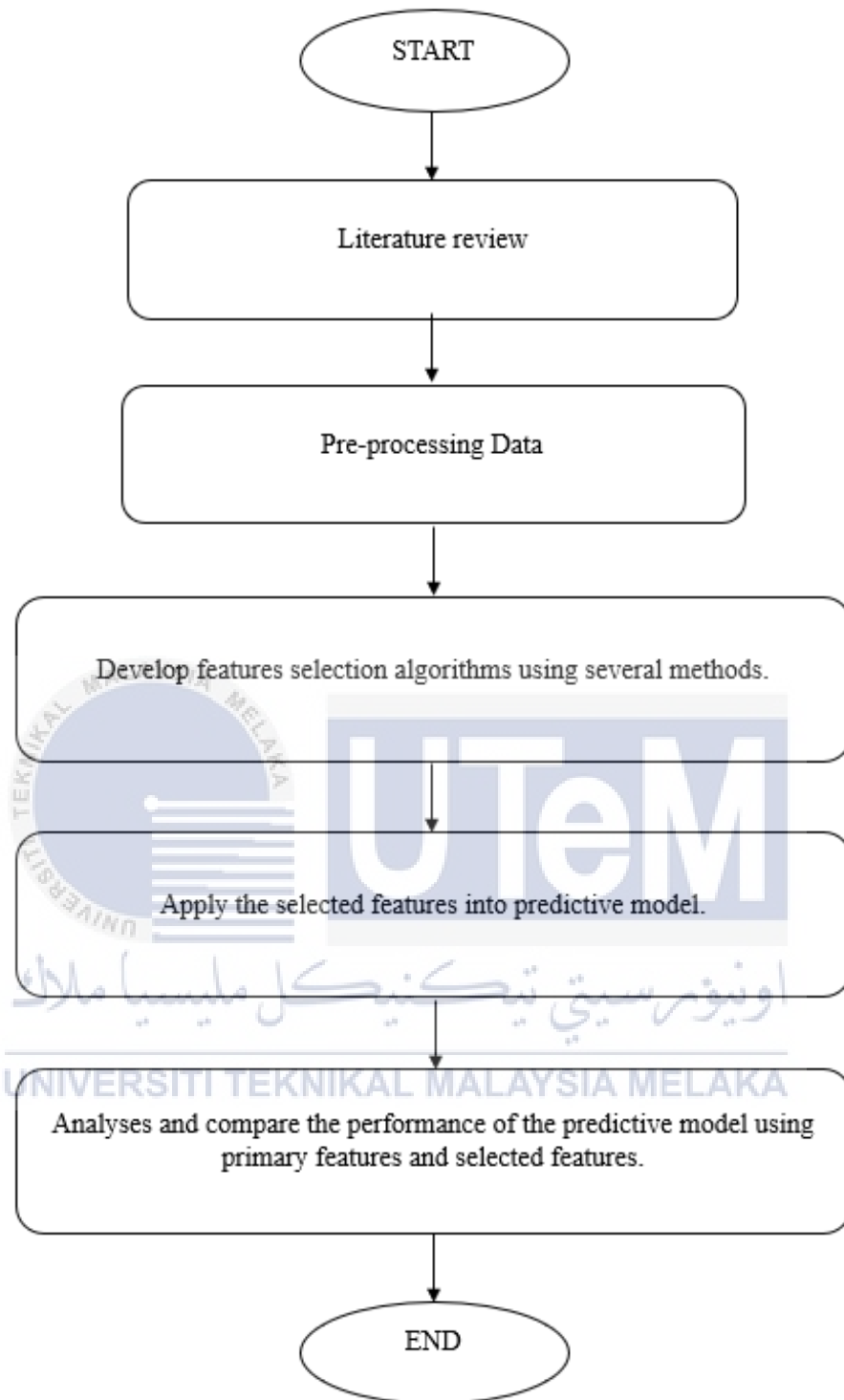
In this chapter, the plan to achieve feature selection method and analyze the waiting time prediction objectives will be explained. This part will also explain the overall approach and techniques conducted in this project. It also outlines the systematic process involved in the project to achieve the project objectives. The methodology will explain clear understanding in analyzing the data, conducting the feature selection, and developing the prediction model.

### 3.1 Overview of the project

In queuing systems, advanced approaches, for instance machine learning and deep learning have played an important role. This project aims to identify the important features in predicting patients waiting time in clinics. In order to perform appropriate feature measures and reduce downtime, feature selection will assist in identifying probable faults or malfunctions in advance. In this project, feature selection is crucial since it determines the most pertinent and instructive features for failure prediction.

On the other hand, choosing the important features, preprocessing data is accomplished first which involves preparing raw data and constructing the data to be competent to machine learning. After that, the data will be retrieved for feature selection by applying correlation-based feature selection, recursive feature elimination and sequential feature selection using the Jupyter Lab software. Other than that, using different feature selection methods results in improving model generalization by selecting the significant features and reducing the overfitting. Thus, to analyze the accuracy of selected features, predictive analysis using the primary features and selected features is implemented.

Furthermore, the web application interface is developed using Visual Studio Code software. The implementation of the web application is to provide estimated waiting times for the patients and to avoid frustration with unpredictable waiting times. This is also to improve the operational efficiency to achieve better outcomes for patients and clinic providers. The web application is established from the predicted model with selected features.



**Figure 3-1: Flowchart of the system design**



### 3.2 Project Implementation

In this project, the feature selection was developed using the Python language on jupyter lab software and Visual Studio Code. The project utilized Python 3 software version 3.9, which is currently in high demand and includes a typing system. The data was pre-processed and uploaded into the program, which aided in randomly partitioning it into training and testing datasets. On the other hand, Microsoft Excel was used for cleaning and splitting the data into training and testing datasets. The project was executed as per the schedule mentioned in the Gantt Chart in the appendices. There were no financial expenses associated with this project as it was completed solely using software tools.

### 3.3 Preprocessing Data

In data preparation, data preprocessing refers to the manipulation of raw data to make it compatible with a subsequent data processing procedure. Historically, it has been a critical initial stage in the data mining process. The goal of data pre-processing is to improve the quality of the data and increase the accuracy of the machine learning model. There are several steps involved in data pre-processing which are data cleaning and data transformation. As for data cleaning, it involves removing any irrelevant data or duplicating the data from the original data. Data cleaning also involves correcting any errors in the data and handling and missing values from the data. Plus, data cleaning involves reducing the amount of data to use for the machine learning model. The goal of data pre-processing is to convert raw data into a format that can be easily interpreted and analyzed by a computer algorithm or human analyst. By doing so, data pre-processing enables accurate and reliable insights to be extracted from the data. As shown in Table 1, the data cleaning was developed by restoring all the data into numerical.

**Table 3-1: Data cleaning**

DATASET	CONDITIONS
FREQUENT	<ul style="list-style-type: none"> <li>• BARU [1]</li> <li>• ULANGAN [0]</li> </ul>
ETHNICITY	<ul style="list-style-type: none"> <li>• MELAYU [1]</li> <li>• CINA [2]</li> <li>• INDIA [3]</li> </ul>
GENDER	<ul style="list-style-type: none"> <li>• FEMALE [1]</li> <li>• MALE [2]</li> </ul>
CATEGORY OF PATIENTS	<ul style="list-style-type: none"> <li>• TODDLE (0-4 YEARS OLD) [1]</li> <li>• PRE-SCHOOL [2]</li> <li>• PRIMARY SCHOOL [3]</li> <li>• SECONDARY SCHOOL [4]</li> <li>• OKU [5]</li> <li>• MATERNITY [6]</li> <li>• ADULT [7]</li> <li>• SENIOR CITIZEN [8]</li> </ul>
OFFICER	<ul style="list-style-type: none"> <li>• PENSIONERS [1]</li> <li>• GOVERNMENT PERSONNEL [2]</li> <li>• COMMUNITY COLLEGE AND OTHERS [3]</li> </ul>
FAST LANE	<ul style="list-style-type: none"> <li>• &lt;60 YEARS OLD (FEMALE) [1]</li> <li>• &lt;60 YEARS OLD (MALE) [2]</li> <li>• &gt;60 YEARS OLD (FEMALE) [3]</li> <li>• &gt;60 YEARS OLD (MALE) [4]</li> </ul>
YIELD	<ul style="list-style-type: none"> <li>• OUTPATIENT [1]</li> <li>• DENTURE [2]</li> </ul>
TREATMENT	<ul style="list-style-type: none"> <li>• TOOTH EXTRACTING [1]</li> <li>• SCALING [2]</li> <li>• TOOTH EXTRACTION [3]</li> <li>• APP FS [4]</li> <li>• FLUORIDE VANISH [5]</li> <li>• ENDO ANTERIOR [6]</li> <li>• ENDO POSTERIOR [7]</li> <li>• OTHER SURGICAL [8]</li> <li>• DENTURE CASE [9]</li> </ul>
BLOOD PRESSURE	<ul style="list-style-type: none"> <li>• 140 AND BELOW [120]</li> <li>• 140 AND ABOVE [140]</li> </ul>

FAMILY HISTORY	<ul style="list-style-type: none"> <li>• YES [1]</li> <li>• NO [2]</li> </ul>
NUMBER OF PATIENTS WAITING	<ul style="list-style-type: none"> <li>• LESS THAN 30 PATIENTS [1]</li> <li>• MORE THAN 30 PATIENTS [2]</li> </ul>
WAITING PERIOD (CLASS)	<ul style="list-style-type: none"> <li>• 0-30 MINUTES [CLASS 1]</li> <li>• 30-60 MINUTES [CLASS 2]</li> <li>• 60-90 MINUTES [CLASS 3]</li> </ul>

The data was obtained from patients who received treatment at the Klinik Kesihatan Alor Gajah dental clinic in 2018 and 2019. The dataset for this project contains a total of 15835 instances and includes patient information such as registration number, appointment time, gender, patient type, referring doctor, treatments, and payment details, which are recorded in Microsoft Excel. The system is designed to calculate the waiting time of patients and record their time of arrival and departure based on whether they have an appointment or are walk-ins. The collected data will be analyzed based on various parameters that influence the waiting time of patients before receiving treatment.

Before performing feature selection, pre-processing data is a crucial step in identifying and correcting errors in the raw dataset that can negatively impact a predictive model. The errors can be of various types, including columns with insufficient information and duplicated rows. For instance, if the dataset is in a text-based format, it needs to be converted into a numerical value to enable clear comprehension by the computer program. Neglecting to address these errors at an early stage can lead to flawed pattern learning and introduce inaccuracies in the model. Initially, before the pre-processing data, the datasets allocated a number of 84 columns

with 29 features collected referring to Figure 3-3. As per Figure 3-4 and Figure 3-5, it shows the pre-processing data and final dataset after pre-processing and data cleaning were performed. As a result, the dataset was developed into 1000 rows (Instances) and 12 columns (Features).





Tarikh	No. Sin	Bera [1], Urangan [0]	Umur	Pezakit Luar						Tema/Janji						KUMPULAN ETNIK	JANTINA Perempuan [1], Lelaki [0]	KATEGORI PESAKIT Tawar (0-4) Terun [0], Pa-Selak [0], Terun [2], Rendan [3], Alangan [5], OKU [3] dan mangsa [6], Disaman [7], Urangan [8]	Pezara [1], Kekurangan Korangan [2], Kog [3], Komuniti [4], M/Lahain [5]	FAST LANE		HASIL	RAWATAN			TEKANAN DARAH NORMAL -120/80 HEPERTENSI 160/90 DAN KE ATAS [0]	TILDA SEBARAH [0], ADA SEBARAH [0]			
				Bi-Peaket	Wajah Depan	Wajah Pergi	Penaman Mata Menunggu	Bi (+3)	Bi-Peaka	Wajah Semula	Wajah Pergi	Penaman Mata Menunggu	Bi (+3)	Off (1) PEREMPUAN (1) LELAKI (2) +40 PEREMPUAN (3) LELAKI (4)	OP (1) DENTURE (2)					TAMPALAN GIGI	CABUTAN		PEMSKALERAN [1], APP FS (GIGI) [2], FV (SAPUAM) [3], ENDO ANTERIOR [4], ENDO POSTERIOR [5], SURGICAL LAHM [6], DENTURE KES [7], NONE [8]							
13/2019	574	1	16	1	8:19	8:40	A	21	1					0	0	India	0	4			1				2			8	1	0
13/2019	575	0	65	1	8:16	8:28	A	12	1					0	0	Melayu	0	6	1	3	1				2			8	0	0
13/2019	576	0	66	1	8:18	8:35	A	17	1					0	0	Melayu	0	9	1	3	1				2			8	1	1
13/2019	579	1	43	1	8:38	9:03	A	25	1					0	0	Cina	1	7			1				2			8	0	0
13/2019	580	1	19	1	8:38	8:44	A	6	1					0	0	Melayu	1	7			1				3			4	1	1
13/2019	584	0	22	1	8:48	9:15	A	27	1					0	0	Melayu	0	7			1				5			4	1	1
13/2019	588	1	20	1	8:55	9:20	A	25	1					0	0	Melayu	1	7			1				3			4	1	1
13/2019	589	1	18	1	9:10	9:35	A	25	1					0	0	Melayu	1	7			1				3			4	1	1
13/2019	591	1	30	1	9:11	9:42	A	25	1					0	0	India	1	7			1				3			4	1	1
13/2019	593	0	29	1	9:22	10:00	B	38	0					0	0	Melayu	1	7			1				2			8	1	1
13/2019	594	1	43	1	9:31	10:12	B	41	0					0	0	Melayu	1	7			1				1			8	0	1

Figure 3-3: The pre-processing data.

SIRI NUMBER	FREQUENT	AGE	ETHNIC GROUP	GENDER	CATEGORY OF PATIENTS	OFFICER	FAST LANE	YIELD	TREATMENT	BLOOD PRESSURE	FAMILY HISTORY	NUMBER OF PATIENTS WAITING	CLASS
621	2	39	2	1	7	4	1	1	2	120	2	1	A
624	1	29	2	2	7	4	1	1	2	120	2	1	A
625	2	66	3	2	8	4	4	1	2	140	2	1	A
628	1	52	1	2	7	4	1	1	3	120	2	1	B
629	2	30	1	1	7	4	1	1	2	120	2	1	A
631	2	41	1	1	7	4	1	1	4	120	2	1	A
634	1	26	2	2	7	4	1	1	5	120	2	1	A
635	2	71	3	2	8	4	4	1	1	140	1	1	B
636	2	20	1	1	7	4	1	1	3	120	1	1	C
639	2	71	1	2	8	1	4	1	2	120	2	1	C
642	1	20	1	2	7	4	1	1	1	120	2	1	B
644	1	25	1	1	7	4	1	1	2	120	2	1	C
645	1	63	1	1	8	4	3	1	2	140	2	1	C
650	1	70	1	1	8	4	3	1	2	140	1	1	C

Figure 3-4: The final Dataset after pre-processing data and data cleaning.

### 3.4 Split data into training and testing

Overfitting in machine learning is essential for unbiased the evaluation of prediction performance. Generally, by training and testing the data, machine learning will acknowledge the model how to behave. On the other hand, data splitting is also developed to amplify database manageability and to enhance query processing performance [20]. The collected data were split into training and testing dataset in accordance with Figure 3-6.

The testing datasets were used to measure the accuracy and precision of the developed training model meanwhile the training datasets were used to develop model by estimating different parameters. In order to develop the feature selection and analyze the accuracy of the selected feature using prediction model, 70% out of 1000 data were developed into training datasets meanwhile the rest 30% data out of 1000 data were utilized to evaluate the decision tree model and predict waiting times.

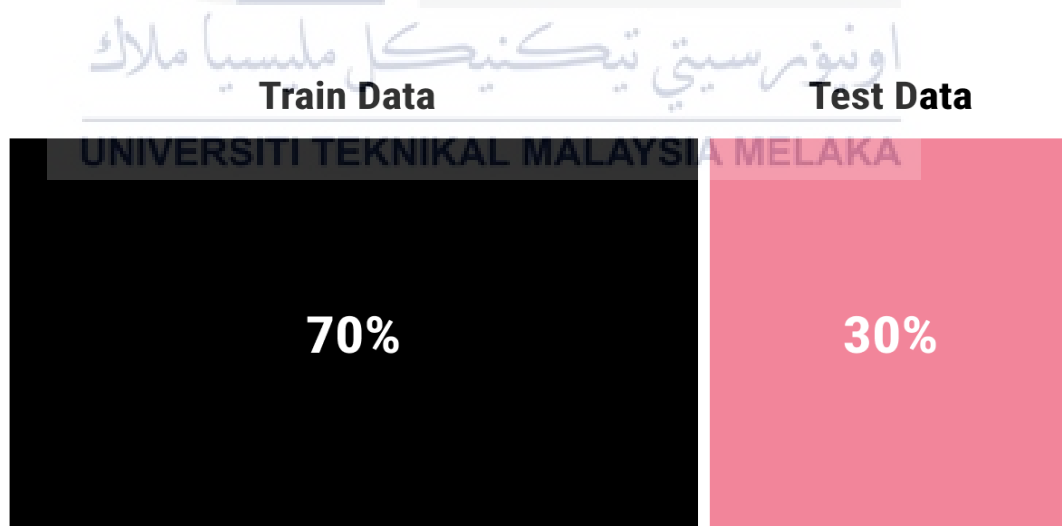


Figure 3-5 : Splitting dataset into training and testing files.

### 3.5 Dataset Shape in Python

Initially, the data is created as .csv file format and the file need to be uploaded into the Jupyter notebook by navigating on the Jupyter interface homepage. This is because this Jupyter Notebook is a cloud-based system operation [21]. Several libraries such as pandas, numpy and seaborn have been imported into the notebook. It will reduce complexity and simplify the python programming.

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
```

**Figure 3-6: Importing important libraries into the Jupyter Notebook.**

Moreover, the Pandas library is the essential library in python where it needs a manipulation data package for analysis and for machine learning elevation. Other than that, python packages such as NumPy are also defined to manipulate the arrays which in NumPy it consists of matrices and functions for mathematical and algebra. On the other hand, to create figures, and plot an area with figures, the *pyplot* function is implemented along with the *matplotlib.pyplot* is a set of routines that makes matplotlib behave like MATLAB. Another package involved in Matplotlib is the seaborn which is used to create statistical visuals. The package is connected to the Pandas data structure that is able to visualize, aid with data exploration and comprehension. Lastly, Scikit-Learn is a Python toolkit for implementing machine learning models and statistical modelling [22]. It may use scikit-learn to create multiple machine learning models for prediction, regression, classification, and clustering.



After uploading the dataset file into the cloud, the dataset file address was called using the *pd.read* command. The commands of *df.head()* are used to present the data content, data header, quantity of data, and the description of data. There are five types of attributes with 1000 instances in this dataset by referring to Figure 3.8. The dataset must not contain any unique object that reflects the lack of a value.

```
df = pd.read_csv("DENTALDATA.csv")
df.head()
```

	NUMBER OF PATIENTS WAITING	FREQUENT	GENDER	YIELD	FAMILY HISTORY	FAST LANE	ETHNIC GROUP	OFFICER	TREATMENT	CATEGORY OF PATIENTS	BLOOD PRESSURE	AGE	CLASS
0	1	2	1	1	2	1	2	4	2	7	120	39	1
1	1	1	2	1	2	1	2	4	2	7	120	29	1
2	1	2	2	1	2	4	3	4	2	8	140	66	1
3	1	1	2	1	2	1	1	4	3	7	120	52	2
4	1	2	1	1	2	1	1	4	2	7	120	30	1

**Figure 3-7: Dataset value.**

On the other hand, the *data.describe* will present the descriptions of the data set. According to Figure 3-9, it display the output from the *data.describe* function meanwhile in Table 2, it explained in detail of the listed description information.

**Table 3-2: List of description.**

Information	Description
Count	The number of values that are not empty.
Mean	The mean (average) value.
Std	Denotes the standard deviation.
Min	The smallest value.
Max	The highest possible value.
25%	Denotes the 25% percentile
50%	Denotes the 50% percentile

	NUMBER OF PATIENTS WAITING	FREQUENT	GENDER	YIELD	FAMILY HISTORY	FAST LANE	ETHNIC GROUP	OFFICER	TREATMENT	CATEGORY OF PATIENTS	BLOOD PRESSURE	AGE	CLASS
count	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000
mean	0.770186	1.391304	1.468944	1.065217	1.767081	2.611801	1.276398	3.583851	3.009317	6.388199	124.658385	37.335404	1.574534
std	0.421368	0.488802	0.499811	0.247293	0.423349	1.747696	0.617503	0.947592	1.762337	1.884909	8.466982	20.790587	0.737748
min	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	120.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000	1.000000	2.000000	1.000000	1.000000	4.000000	1.250000	7.000000	120.000000	22.000000	1.000000
50%	1.000000	1.000000	1.000000	1.000000	2.000000	2.000000	1.000000	4.000000	3.000000	7.000000	120.000000	36.000000	1.000000
75%	1.000000	2.000000	2.000000	1.000000	2.000000	5.000000	1.000000	4.000000	5.000000	7.000000	120.000000	55.750000	2.000000
max	1.000000	2.000000	2.000000	2.000000	2.000000	5.000000	3.000000	4.000000	9.000000	8.000000	140.000000	83.000000	3.000000

**Figure 3-8: Data description**

### 3.6 Develop features selection.

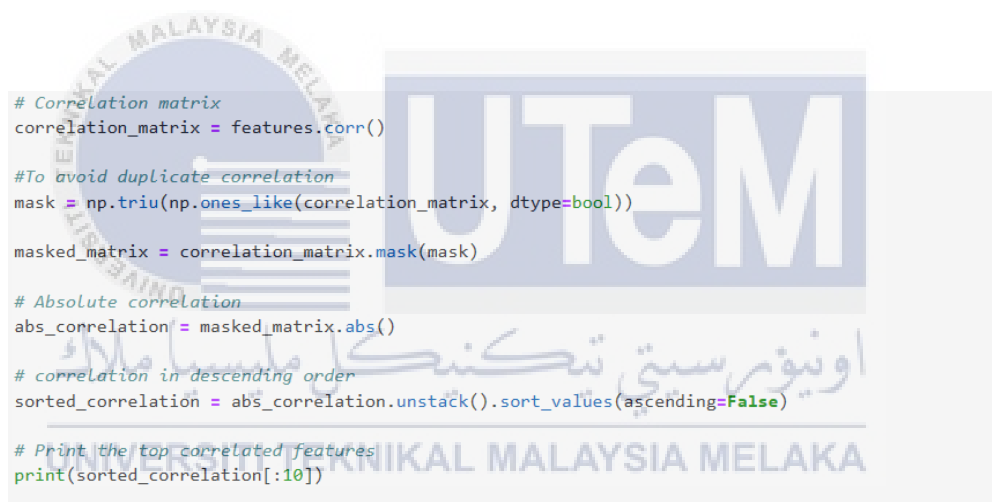
Basically, feature selection is analyzed to remove irrelevant data. Besides, feature selection is the process of selecting a subset of relevant features or variables from a larger set of features that will be used to build a predictive model [23]. The goal of feature selection is to improve the model's performance by reducing the number of irrelevant, redundant, or noisy features that can negatively impact the model's accuracy, complexity, and computational cost. Thus, in this task, the data will be crystallized to make the prediction more accurate and to improve the performance of the model.

#### 3.6.1 Correlation-based feature selection.

In verifying feature selection, the most relevant features are selected to evaluate the correlation between features and the target variable to determine their relevance. Thus, in this project, correlation-based feature selection is used to identify the relevant features. The data will be labelled with their respective types. Other than that, the data selected for analysis were labeled according to their respective types, which fell into eight categories. These categories include number of patients waiting,

age, gender, ethnicity, yield, officer, category of patients, fast lane, treatment, blood pressure, and family history. Data values for each parameter were collected to predict the waiting time in the queue.

In developing feature selection, Jupyterlab is used to develop the feature selection. Theoretically, there are few methods involved in developing the correlation-based feature selections which are Compute Correlation Matrix, determine Highly Correlated Features and lastly Select Features [24]. As shown in Figure 3-10, it is the coding generated in the Jupyter lab to develop correlation-based feature selection.



```

# Correlation matrix
correlation_matrix = features.corr()

#To avoid duplicate correlation
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

masked_matrix = correlation_matrix.mask(mask)

# Absolute correlation
abs_correlation = masked_matrix.abs()

# correlation in descending order
sorted_correlation = abs_correlation.unstack().sort_values(ascending=False)

# Print the top correlated features
print(sorted_correlation[:10])

```

**Figure 3-9: Coding implementation for correlation-based feature selection.**

In accordance with Figure 3-10, the *mask = np.triu* is used to create a mask under the correlation matrix to avoid duplicate correlation and the *masked\_matrix* is defined to apply the masked onto the coding. On the other hand, the *abs\_correlation* is used to get the absolute correlation values for each of the features and the *sorted\_correlation* is used to sort the features by the correlation values. The absolute correlation value is developed to define the magnitude of the correlation, as the greater

the absolute value, the stronger the correlation. Furthermore, the correlation is an indicator of statistical significance that describes the interaction between two variables. The correlation statistics can be used to either continuous or binary data.

### 3.6.2 Recursive Feature Elimination.

To identify the dataset's key features, recursive feature elimination is implemented which the process involves developing a model with the remaining features after repeatedly removing the least significant parts until the desired number of features is obtained. A filtering method is a common method developed in Recursive feature elimination [25]. The features will be evaluated individually, and the most important features are selected according to the correlation and mutual information. In this project, the recursive feature elimination is developed in Jupyter lab using the python languages. The dataset is loaded into the workspace and defined shown in Figure 3-

12.



```
data = pd.read_csv('DentalSet.csv')
data.shape
```

**Figure 3-10: Loading the dataset into the workspace.**

Furthermore, to develop recursive feature elimination, the datasets are also classified into training and testing sets to evaluate the performance of the feature selection. The training set is a subset used to accumulate the feature selection method and it also comprises of the relevant of goal variables and the input features. In order to determine the relevance of the attributes, the algorithm learns from this training

data. On the other hand, the testing set is a distinct section of the initial dataset that is utilized to evaluate the effectiveness of the feature selection. It is used to separate the dataset to evaluate the likelihood of the selected features to expedite new data. In this section, the data is separated into x train, x test, y train and y test. As referred to Figure 3-13, the *train\_test\_split* is clarified with test size 0.3 (30%) from the data.

```
X_train, X_test, y_train, y_test = train_test_split(
    data.drop(labels=['CLASS'], axis=1),
    data['CLASS'],
    test_size=0.3,
    random_state=0)
X_train.shape, X_test.shape
```

**Figure 3-11: Training and testing the dataset.**

Next, *SelectKBest* is defined to extract the best features of the given dataset. The *SelectKBest* method selects the features according to the k highest score. By changing the '*score\_func*' parameter, it can apply the method for the regression data. Selecting the best features is an important process when preparing a large dataset for training. It helps to eliminate less important parts of the data and reduce training time. The implementation of the feature selection is shown in Figure 3-14.

```

from sklearn.feature_selection import SelectKBest, chi2

best_features = SelectKBest(score_func=f_regression, k=5).fit(X_train, y_train)
selected_features = X_train.columns[best_features.get_support()]

print(f"Number of selected features: {len(selected_features)}")
print(f"Selected features : {list(selected_features)}")

```

**Figure 3-12: The development of the recursive feature elimination.**

On the other hand, to eliminate the insignificant features, the algorithm of gradient boosting classifier is applied. The algorithm combines the weak learning models to achieve powerful predicting model by repeatedly selecting the functions that lead to negative gradient or weak hypothesis. This algorithm not only predicts continuous target variable, but it could predict the categorical target variable. Plus, in the gradient boosting classifier, the desired number of features is defined to make the process more accurate.

So, in this algorithm, the desired number of features could vary at least half from the original allocated features, but in this project six features are desired in this method. Moreover, to improve the model robustness, the *RepeatedStratifiedKfold* is applied to control the randomness of the cross-validation. It is also used to remove repeated instances to reproduce multiple function call output and improve the estimated performance. Besides, a pipeline tool is used which will link all the manipulated data to create a pipeline. The coding implementation is present in Figure 3-15.

```

rfe = RFE(estimator=GradientBoostingClassifier(), n_features_to_select=5)

model = GradientBoostingClassifier()

pipe = Pipeline([('Feature Selection', rfe), ('Model', model)])
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=5, random_state=240)
n_scores = cross_val_score(pipe, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1)
np.mean(n_scores)

```

**Figure 3-13: Applying Gradient Boosting Classifier algorithm into the data set.**

In modelling process, pipeline is a structured and systematic in the analysis that helps in sequencing and automating multiple steps of machine learning. This is to ensure consistency and reproducibility workflow. Pipeline is also involved in data pre-processing and model training. In terms of recursive feature elimination, Pipeline is involved in feature elimination process by eliminating insignificant features in the dataset. This results in a more organized and facilitated process with different configurations.

Other than that, ranking the features is one of the processes in developing recursive feature elimination that involves using the feature importance or coefficients. So, the features are ranked according to their importance meanwhile the least important features will be eliminated from the data set. The ranking feature and the ranking scores are developed as shown in Figure 3-16. This continuous process will be implemented until the desired number of features is accomplished.

```

rfecv_df = pd.DataFrame(rfecv.ranking_, index=X.columns, columns=['Rank']).sort_values(by='Rank', ascending=True)
rfecv_df.head()
rfecv.ranking_
rfecv.grid_scores_

```

**Figure 3-14: Recursive feature ranking applied to the Data set.**

### 3.6.3 Sequential Feature Selection

The sequential Feature Selection is performed by constantly importing and eliminating features from the dataset to enhance the performance of the prediction model. Initially, the selector of the predictive model will define the number of features to select, the imprecision of the improvement and the scoring metric. The scoring measure is used to evaluate the model on the training set. The allocated features that elevate the model's cross validation score the most is added to the selected features set meanwhile the features that declines the model's cross validation score the least is eliminated, whichever delivers the most improvement in the scoring metric.

In this project, the Sequential Feature Selection is developed in Jupyter lab notebook using python programming. Initially, the import *matplotlib.pyplot* as *plt* is described so that the *matplotlib* will perform as *MATLAB*. Based on Figure 3-17, the necessary libraries are also imported for instance *pandas* and *numpy*.



```
#DATASET PREPARATION
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df = pd.read_csv('DENTALDATA.csv')
df.columns = ['NUMBER OF PATIENTS WAITING', 'FREQUENT', 'GENDER', 'YIELD', 'FAMILY HISTORY', 'FAST LANE', 'ETHNIC GROUP', 'OFFICER', 'TREATMENT', 'CATEGORY OF PATIENTS', 'BLOOD PRESSURE']
print('TREATMENT', np.unique(df['TREATMENT']))
df.head()
```

**Figure 3-15: Preparation of Dataset to develop sequential feature selection.**

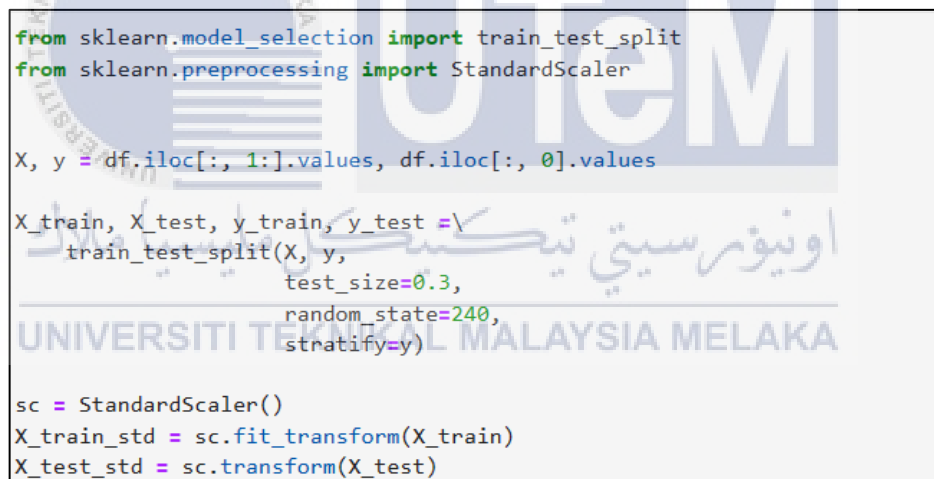
In addition, the columns of the data are defined to return index object or to represent the names of the columns in the Dataset. So, in selecting the best features in the Dataset using Sequential Feature Selection, the *KNeighborClassifier* is used based on choosing the nearest k-neighbors of the target point. It also provides functionality for



unsupervised and supervised data which will not make any assumption on underlying data. The algorithm also will store the trained data and will classify the new data into a category that is much similar to the trained data.

Other than that, the *StandardScaler* is also defined to standardize the features by subtracting the mean and scaling the features into unit variance by dividing all the values with standard deviation. As a result, the selected features in performed in an independent way. After the data is split into training and testing size, the data is transformed and fitted into the *StandardScaler* algorithm so that it can estimate the empirical mean and standard deviation. The coding implementation is presented in

Figure 3-18.



```

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X, y = df.iloc[:, 1:].values, df.iloc[:, 0].values

X_train, X_test, y_train, y_test = \
    train_test_split(X, y,
                    test_size=0.3,
                    random_state=240,
                    stratify=y)

sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

```

**Figure 3-16: Applying Standard Scaler algorithm into the data.**

Additionally, in selecting the best features using sequential feature selection is specify the number of features to select. In this project, the *SequentialFeatureSelector* algorithm is used to select the best features based on the cross-validation score and the best single feature with the highest score will be determined. In every stage, this

selector will remove the lowest cross-validation score to form a feature subset. Predominantly, through the addition of further features that have no impact on the criteria, features are gradually added to an empty feature set.

```

from mlxtend.feature_selection import SequentialFeatureSelector as SFS

sfs1 = SFS(model,
            k_features=5,
            forward=True,
            floating=False,
            verbose=2,
            scoring='accuracy',
            n_jobs=-1,
            cv=5)

sfs1 = sfs1.fit(X_train_std, y_train)

```

**Figure 3-17: Implementation of Sequential feature selector into the data set.**

Furthermore, in inspecting the result the *metric\_dict* is generated to provide metadata into the tensorboard. The key-value of the corresponding value in *metric\_dict* is also generated for evaluating and sampling all features in a specified range. Based on Figure 3-20, the *metric\_dict* is evaluated in a range of values or confidence intervals of 0.95 or 95% confidence intervals so that 95% probability falls within the range.

```

#INSPECTING RESULT
sfs1.subsets_
metric_dict = sfs1.get_metric_dict(confidence_interval=0.95)
metric_dict

```

**Figure 3-18: Establish metrics computation into the model.**

### 3.7 Develop predictive model.

In model development, numerous types of machine learning are trained and compared. In this project, decision tree and logistic regression is the systematic approach in building and optimizing the model. On the other hand, causal explanations are used to evaluate which features are more predictive. So, the Python library Scikit-learn is utilized in Machine Learning where the numerical and categorized data can be used.

#### 3.7.1 Decision Tree Model

The process of designing a Decision Tree model, the architecture prediction is initialized to the `clf` with a maximum depth of 3 and a random state of 240. The max depth option controls the maximum depth of the tree. This operation is to make sure there will not be any overfitting. The random state parameter ensures that the findings may be reproduced in further studies. The algorithm will now be fitted to the practice data.

Based on Figure 3-21, the Decision Tree was built from the imported training dataset and declared as a model. The most important feature is the ability to extract descriptive information useful for making decisions from the data given. A decision tree can be built with the data in a training or testing set. Based on Figure 3-21, the `train_test_split` function is used to train the models and shuffles the dataset into training and testing size. The `test_size` parameter is the proportion from the original of the dataset which it is define to 0.3 (30%) from the data is utilized for testing the model.

```

# Load the selected features dataset
data = pd.read_csv('SFSSelected.csv')

# Perform feature selection
features = data.drop('CLASS', axis=1)
target = data['CLASS']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=240)

# Train a decision tree model
model = DecisionTreeClassifier(random_state=240)
model.fit(X_train, y_train)

```

**Figure 3-19: Development of decision tree predictive model.**

### 3.7.2 Logistic Regression model.

Analyzing the relationship between dependent variables and independent variables involved a statistical method of logistic regression. The regression analysis consists of two possible outcomes which should be in binary for the dependent variables. Logistic regression aims to discover the optimal model that can forecast the likelihood of a categorical outcome based on independent variable values. The linear combination of independent variables is converted into probability estimates between 0 and 1 utilizing this function, with coefficients computed via maximum likelihood estimation. As referred to Figure 3-22, it shows the coding implementation in developing the logistic regression predictive model.

```

# Load the selected features dataset
data = pd.read_csv('SFSSelected.csv')

# Perform feature selection
features = data.drop('CLASS', axis=1)
target = data['CLASS']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=240)

# Train a model
model = LogisticRegression(random_state=240)
model.fit(X_train, y_train)

```

**Figure 3-20: Development of the Logistic Regression predictive model.**

By using a dataset with known outcomes, the logistic regression model is trained. During this process, coefficients that maximize the likelihood of observed outcomes are estimated meanwhile optimization techniques such as gradient descent are commonly used to accomplish this. Having completed training, it could anticipate probability results for observations belonging to positive classes and determine whether an observation exceeds or falls below specific threshold. The features that are above the threshold are classified as positives and the features that are not reaching the threshold considered negatives.

### 3.7.3 Comprehensive Analysis

In addition, to develop a well-rounded evaluation of the model performance, accuracy, precision, and recall is necessary. The accuracy provides the ratio of correctly predicted instances, precision assess the model's ability and recall evaluates the capacity of all identified instances. Balancing the trade-off from the imbalanced datasets, this comprehensive analysis is needed to enhance the model interpretability and ensure that the selected features are optimized according to desired goals. This comprehensive analysis also provides valuable insights into the model's behavior, contributing to informed decision-making in the feature selection process. Based on Figure 3-23 and Figure 3-24, it shows the implementation of the comprehensive analysis of the training and testing dataset of the selected features.

```

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy, precision, recall and f1 score of the testing size
accuracy = accuracy_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

```

**Figure 3-21: Implementation of comprehensive analysis of testing dataset.**

```

# Make predictions
y_pred1 = model.predict(X_train)

# Calculate accuracy, precision, recall and f1 score of the training size
accuracy = accuracy_score(y_train, y_pred1)
accuracy = accuracy_score(y_train, y_pred1)
precision = precision_score(y_train, y_pred1)
recall = recall_score(y_train, y_pred1)
f1 = f1_score(y_train, y_pred1)

```

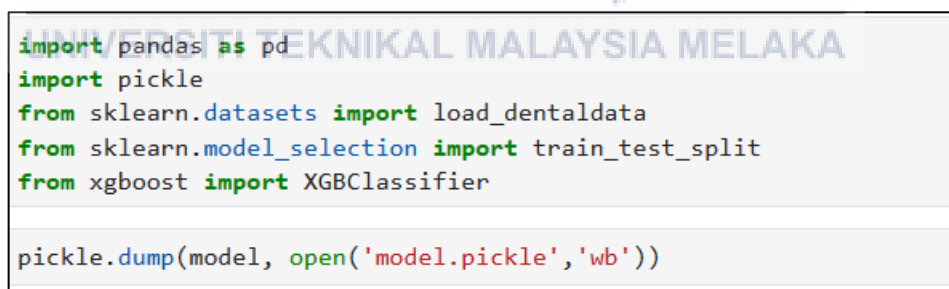
**Figure 3-22: Implementation of comprehensive analysis of training dataset.**

It is crucial to create a comprehensive evaluation of prediction patient waiting time by considering metrics like accuracy, precision, recall and F1 score in order to obtain an intricate view on the performance of a model. Even though accuracy serves as a commonly utilized metric, its usability may be limited particularly when dealing with datasets that lack balance between different classes. Generally, the metric 'precision' denotes the proportion of actual positive predictions in relation to predicted positives, indicating how accurately a model can detect important features [26]. On the other hand, recall divided by total actual positives illuminates the capacity of the model to capture all such instances. A combined measure known as F1 score utilizes precision and recall through their harmonic mean with consideration given also to false negatives and false positives, thereby providing more robust evaluation across metrics that assess both precision and recall simultaneously.

### 3.8 Development of Web application for waiting time prediction

After the feature selection and the accuracy of the prediction model were developed, a prediction waiting time was generated using Visual Studio Code (VSC) written in Python language and HyperText Markup Language (html). Basically, the markup language is displayed into the web browser for instance into Microsoft edge or Google Chrome. However, in this project the web browser used is Microsoft edge because it is more compatibility than google chrome since different web browsers have different levels of support to web technologies.

Moreover, after the prediction model is analyzed using primary features and selected features, the prediction model is converted into pickle file type. This is because pickle file is very flexible, in which different variables can be saved into pickle file and the file can be loaded back with different python session. The prediction model in pickle file is then downloaded from the Jupyterlab Notebook and imported into Visual Studio Code as referred to Figure 3-25.



```
import pandas as pd
import pickle
from sklearn.datasets import load_dentaldata
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

pickle.dump(model, open('model.pickle', 'wb'))
```

**Figure 3-23: Convert prediction model into Pickle file.**

In general, Visual Studio Code software is a platform where the interface is designed including debugging and task execution. It strives to give only the tools required for a speedy code-build-debug cycle, leaving more sophisticated workflows

out [27]. In accordance with Figure 3-25, there are several tools imported into the Visual studio code's editor. Several types of libraries need to be installed to run the command, for instance import pickle and XGBClassifier. Table 3 explains in detail each library function in developing the web application.

**Table 3-3: Library function of web application.**

Type of Library	Description
Pickle	Pickle is a Python object serializer and de-serializer, commonly known as marshalling or flattening. Serialization is the process of transforming a memory item to a byte stream that may be saved on a disc or sent over a network
XGBClassifier	Tools to allocate gradient-boosted decision tree for regression and ranking matter. It also optimizes the machine learning model.

Furthermore, to create the web interface, a predict waiting time function needs to be define in JavaScript. Basically, JavaScript is created to embed it directly to HTML pages. So in order to access the elements, the *getElementById()* function is used to correspond the return value elements. After that, it will load the predicted model in pickle file into the *waitingTimePrediction* function and the model will be retrieved into *getElementById()*. As referred to Figure 3-26, it shows the coding implementation in the Visual Studio Code. On the other hand, Figure 3-27 presents the coding implementation to develop a web page layout using html file.



```

var waitingTimePrediction = predictWaitingTimeFromModel(inputData);

document.getElementById('waitingTimePrediction').innerText = waitingTimePrediction;
}

function predictWaitingTimeFromModel(inputData) {
  var model = loadModelFromPickle('model.pkl');
  var prediction = model.predict(inputData);
  return prediction;
  return "Sample waiting time prediction";
}

```

Figure 3-24: Embedded function from pickle file into VS Code.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Waiting Time Prediction at Dental Clinic</title>
7   <link rel="stylesheet" href="CSstyles.css">
8 </head>
9 <body>
10  <div class="container">
11    <h1>Waiting Time Prediction at Dental Clinic</h1>
12    <h2>Please Fill in patient's data below:</h2>
13    <h3>Method: Sequential Feature Selection</h3>
14    <form id="queueForm">
15      <label for="waitingPatients">Number of Waiting Patients:</label>
16      <input type="number" id="waitingPatients">
17    </select><br><br>

```

اونيوم سيم تیکنیکا ملسيا ملاك

Figure 3-25: Creating web page layout.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## CHAPTER 4

### RESULT AND DISCUSSION



This chapter will go through the results from the preceding methodology part. To show the efficacy of the machine learning model, all gathered results will be presented in the form of a graph and an image. The accuracy and precision of each model will be thoroughly discussed and assessed. This section's findings will be used to examine the project's objectives.

## 4.1 Feature Selection

### 4.1.1 Correlation-based Feature Selection

The Correlation-based Feature Selection in predicting waiting time in Dental Clinic was successfully developed in Colab Jupyter Notebook. According to Figure 4-1, it describes the python program to develop Correlation based feature selection method. High correlation features are more linearly dependent and have approximately the same influence on the dependent variable. Thus, when two features contained high correlation features, the lower correlation can be dropped in the model.

CATEGORY OF PATIENTS	AGE	0.757033
FAMILY HISTORY	BLOOD PRESSURE	0.495924
BLOOD PRESSURE	AGE	0.474225
FAMILY HISTORY	AGE	0.473163
OFFICER	AGE	0.415093
NUMBER OF PATIENTS WAITING	FAST LANE	0.290732
OFFICER	CATEGORY OF PATIENTS	0.266822
FAST LANE	AGE	0.253515
CATEGORY OF PATIENTS	BLOOD PRESSURE	0.249405
OFFICER	BLOOD PRESSURE	0.239093
dtype: float64		
Number of selected features: 5		
Selected features : ['NUMBER OF PATIENTS WAITING', 'FREQUENT', 'TREATMENT', 'CATEGORY OF PATIENTS', 'AGE']		

**Figure 4-1: Number of selected features using correlation-based feature selection.**

In accordance with Figure 4-1, the selected features are number of patients waiting, frequent, treatment, category of patients and age. Plus, category of patients and age have the highest correlation between other features. Identification of redundancy is denoted by high correlation and enhance interpretability simultaneously. On the other hand, the lowest correlation relies between officer and blood pressure which results in eliminating the features. This is because by eliminating the lowest correlation, it could improve the model performance. Hence, by developing feature selection using correlation-based feature selection, the 5 highest correlations are selected out of 12

original features. The relationship between the features is interpreted and identified using the heatmap correlation matrix as shown in Figure 4-2.

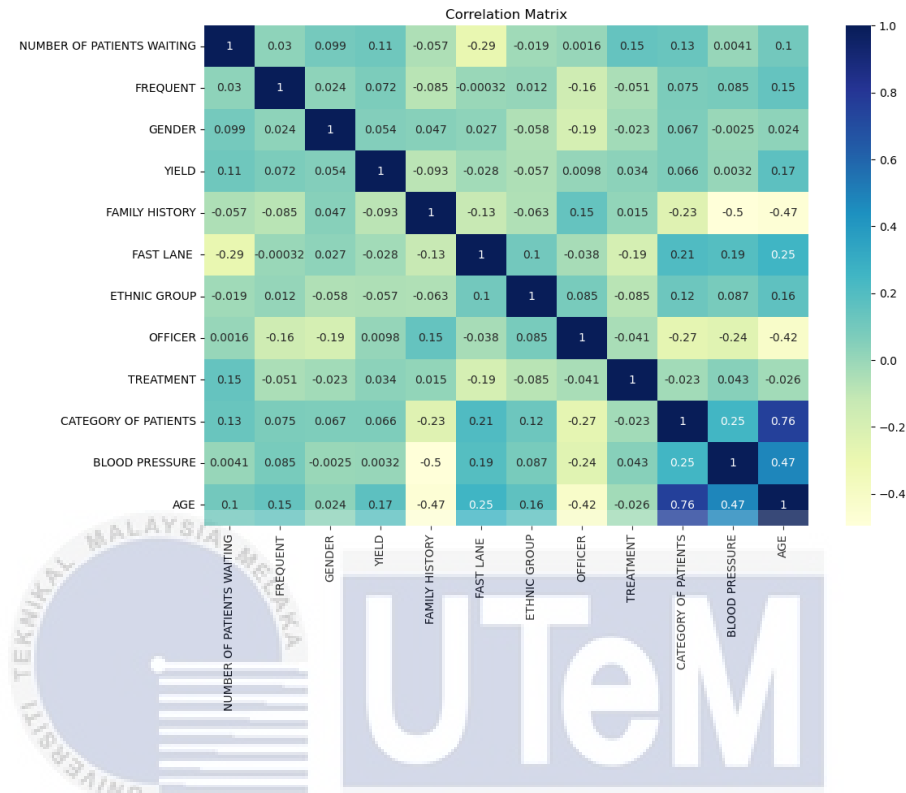


Figure 4-2: Correlation Matrix Heatmap

According to Figure 4-2, the correlation heatmap involves interpreting the color-coded matrix of the correlation coefficients between the features. In a heatmap, each cell corresponds to the correlation strength between the respective pair of variables, with colors ranging. For instance, the darkest blue in the category of patients indicates a strong positive correlation and the fare green in the family history indicates a strong negative correlation. A positive correlation suggests that as one variable increases, the other tends to increase as well, while a negative correlation indicates an inverse relationship. This shows that the features of category of patients and family history developed negative correlation. Thus, the intensity of the color reflects the strength of the correlation.

```

X_train_selected = best_features.transform(X_train)
X_test_selected = best_features.transform(X_test)

clf.fit(X_train_selected, y_train)
print(f"Accuracy : {clf.score(X_test_selected, y_test):.3f}")

Accuracy : 0.785

X_train_selected = best_features.transform(X_train)
X_test_selected = best_features.transform(X_test)

clf.fit(X_train_selected, y_train)
print(f"Accuracy : {clf.score(X_train_selected, y_train):.3f}")

Accuracy : 0.732

```

**Figure 4-3: Accuracy of the correlation-based model.**

As indicated in Figure 4-3, the selected features are trained and fitted into the model. The output shows that the accuracy of the testing model is 0.785 which approximately 78.5% meanwhile the accuracy of the training model is 0.732 that approximately 73.2%. The percentage accuracy represents the correctly predicted features and instances on the training and testing dataset. The testing accuracy shows that the model performed well in learning the patterns as the testing model has indicated the underlying pattern in the testing set. On the other hand, the training accuracy shows that the model performed well in generalizing the unseen data although there is a small significant difference between training and testing.

#### 4.1.2 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a feature selection that will fits a model and eliminate the inadequate feature until the desired features are achieved [28]. The RFE needs a certain number of characteristics to be retained so cross-validation is used

to score several feature subsets and the top scoring is picked to determine the optimal number of features. The *RFECV* visualizer depicts the number of features in the model as well as the cross-validated test score and variability. Initially, the algorithm will fit into the primary features and performed statistical analysis for instance accuracy or other relevant analysis. Along with that, the algorithm will rank the features based on the significance scores. In accordance with Figure 4-4, it displays the selected features from the recursive ranking method.

```
pd.DataFrame(rfe.support_, index=X.columns, columns=['Rank']).head()
```

	Rank
NUMBER OF PATIENTS WAITING	True
GENDER	True
FAST LANE	True
ETHNIC GROUP	True
TREATMENT	True

```
X.columns
Index(['NUMBER OF PATIENTS WAITING', 'GENDER', 'FAST LANE ', 'ETHNIC GROUP',
      'TREATMENT'],
      dtype='object')
```

**Figure 4-4: Recursive Feature Elimination command**

In this project, the method of recursive feature elimination was chosen because the method could enhance in reducing the complexity of the trained model. It improves the training speed and intelligence. In addition, the trained model could produce noise from insignificant features. So, the recursive feature elimination method may possibly remove the unnecessary noise that leads to model regularization and restrain the overfitting. Then, the least significant features will be eliminated in the feature

elimination process. Lastly, it will refit the model with only important features and a comprehensive analysis will be conducted to the refit model. In accordance with Figure 4-5, the *GradientBoostingClassifier* command is used as a functional gradient algorithm that will repeatedly selects a less functional features. This command also combines deficient learning models to produce a dominant predicting model.

```
rfe = RFE(estimator=GradientBoostingClassifier(), n_features_to_select=5)

model = GradientBoostingClassifier()

pipe = Pipeline(['Feature Selection', rfe), ('Model', model)]
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=5, random_state=240)
n_scores = cross_val_score(pipe, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1)
np.mean(n_scores)

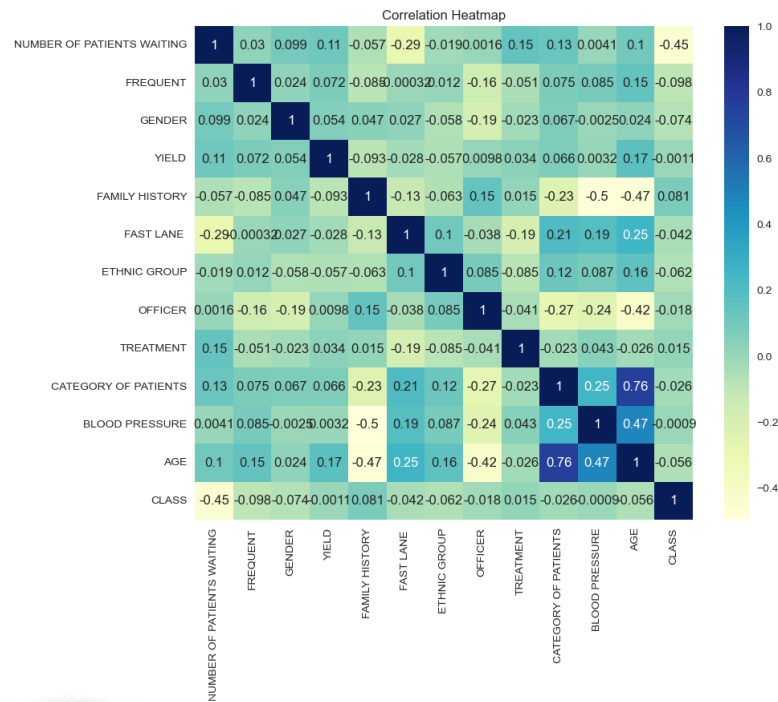
0.8764333333333333

n_scores = cross_val_score(pipe, X_test, y_test, scoring='accuracy', cv=cv, n_jobs=-1)
np.mean(n_scores)

0.8741666666666668
```

**Figure 4-5: Accuracy of the Recursive Feature Elimination model.**

In accordance with Figure 4-5, the training accuracy indicates 87.64% meanwhile the testing accuracy indicates 87.41%. This shows that during the feature elimination process, the model achieved a high level of accuracy by training the dataset. On the other hand, by reducing the set of features, it maintains the predictive power and avoids overfitting by testing the dataset. Plus, it shows that the recursive feature elimination is effectively demonstrated. In conclusion, using the *repeatedStratifiedKFold* enhances the reliability of the training and testing accuracy by repeatedly shuffling and splitting the data to improve robust assessment of the model's performance.



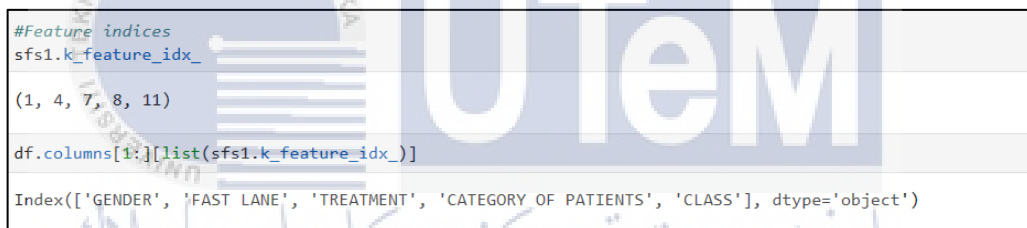
**Figure 4-6: Correlation matrix heatmap**

On the other hand, Figure 4-6 depict the correlation matrix heatmap. The heatmap provides an intuitive visual representation of the relationships between features. Other than that, Multicollinearity occurs when two or more features in a dataset are highly correlated, which can negatively impact the model's performance and the interpretability of feature importance. The heatmap correlation allows you to visualize these correlations and make informed decisions about which features to retain or eliminate.



### 4.1.3 Sequential Feature Selection

Theoretically, in improving the performance of the sequential feature selection model, a greedy algorithm that removes or adds based on the features contribution in the dataset is compulsory. Evaluating the predictive model with the selected features and assess it improves the model accuracy is one of the processes in analyzing the model performance. In this method, a sequential feature selector is utilized for the interpretation and clarification of the features. In accordance with Figure 4-7, the output displays the selected features which are gender, fast lane, treatment, category of patients and class.



```
#Feature indices
sfs1.k_feature_idx_

(1, 4, 7, 8, 11)

df.columns[1:][list(sfs1.k_feature_idx_)]

Index(['GENDER', 'FAST LANE', 'TREATMENT', 'CATEGORY OF PATIENTS', 'CLASS'], dtype='object')
```

**Figure 4-7: Number of selected features using sequential feature selection.**

Moreover, the selected features are trained and transformed into the sequential feature selector in terms of training and testing size. As referred to Figure 4-8, the training accuracy indicates 93% performance of the model meanwhile the testing accuracy indicates about 87% performance of the model. The training set is highly relative to the degree of accuracy and the model has developed the pattern and relationship between the training and testing set.

Aside from that, it is also crucial to determine whether the model is overfitting the training data even if the training accuracy is desirable. Noise or other specific characteristics that are well to unseen is one of the causes of overfitting the data. Apart from that, the test accuracy reflects on the model's performance that provides estimation of how well it trained to unseen features. The fact that the test accuracy is slightly lower than the training accuracy is common and expected. A small drop in accuracy is acceptable and can be attributed to the model's ability to generalize patterns learned during training to new instances.

```
X_train_sele = sfs1.transform(X_train_std)
X_test_sele = sfs1.transform(X_test_std)

model.fit(X_train_sele, y_train)
print('Training accuracy:', np.mean(model.predict(X_train_sele) == y_train)*100)
print('Test accuracy:', np.mean(model.predict(X_test_sele) == y_test)*100)

Training accuracy: 92.99610894941634
Test accuracy: 87.6923076923077
```

**Figure 4-8: The accuracy of the sequential model.**

To achieve robust and reliable model performance, complexity and generalization are necessary for training and testing the datasets. In accordance with the result, it shows that the model performed well in learning the patterns and it correctly predicted the outcome of the selected features. On the other hand, the testing accuracy shows that the model performed well in generalizing the unseen data although there is a small significant difference between training and testing.

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
1	(1,)	[0.7692307692307693, 0.7692307692307693, 0.784...	0.770437	(1,)	0.009289	0.007227	0.003614
2	(1, 11)	[0.8269230769230769, 0.7692307692307693, 0.764...	0.797662	(1, 11)	0.051644	0.040181	0.020091
3	(1, 4, 11)	[0.8076923076923077, 0.8076923076923077, 0.882...	0.85641	(1, 4, 11)	0.056628	0.044058	0.022029
4	(1, 4, 8, 11)	[0.8653846153846154, 0.7692307692307693, 0.980...	0.895551	(1, 4, 8, 11)	0.096801	0.075314	0.037657
5	(1, 4, 7, 8, 11)	[0.8653846153846154, 0.7692307692307693, 0.960...	0.891629	(1, 4, 7, 8, 11)	0.088683	0.068998	0.034499

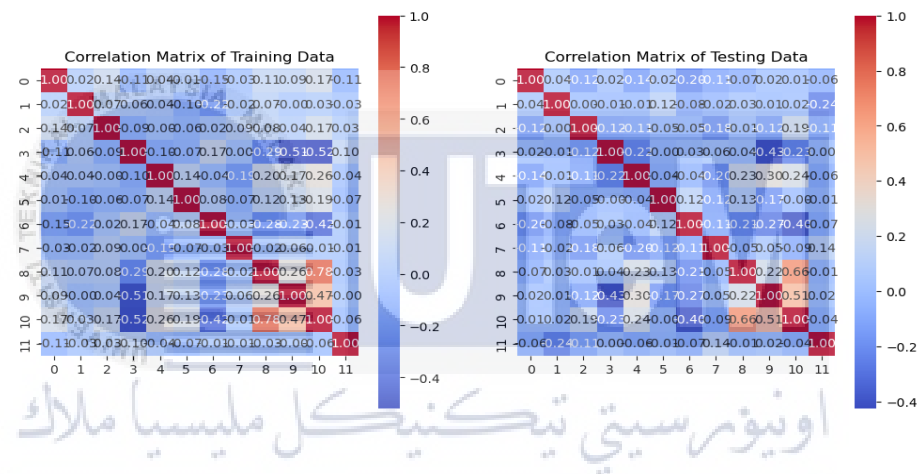
**Figure 4-9: Feature index of the sequential model.**

In accordance with Figure 4-9, every feature index is evaluated into a few statistical algorithms to assess the reliability and stability performance of selected features. The statistical algorithms are cross validation, standard deviation, and standard error. Theoretically, cross validation helps in assessing the selected features perform on the unseen data and it reduces the overfitting. Aside from that, the amount of dispersion is measured using standard deviation where it helps in quantifying the stability of the selected features across multiple iterations. Features with small standard deviation are highly consistent and relevant. Plus, the variability of a sample statistic is measured using the standard error. This algorithm will quantify the uncertainty and terminate the repetition of selected features in the analysis. The lower the standard error, the higher the confidence in the reliability of the selected features.

**Table 4-1: Statistical algorithm applied to the selected features.**

Statistical Algorithm	Average score	Cross-Validation	Standard Deviation	Standard Error
Score	0.8916	0.0887	0.0690	0.0345

In proportion with Figure 4-9, the cross validation of the selected features is 0.0887 which performed consistently across various selected features meanwhile the standard deviation indicates 0.0690 in the performance. The lower the standard deviation, the more selected features performance are highly consistent. The standard error conveyed an error of 0.0345 which resulted in the average performance from the cross-validation score. Thus, the lower the standard error, the higher the accuracy of the estimated mean performance. This feature selection process shows a stable result and reliable set of selected features.

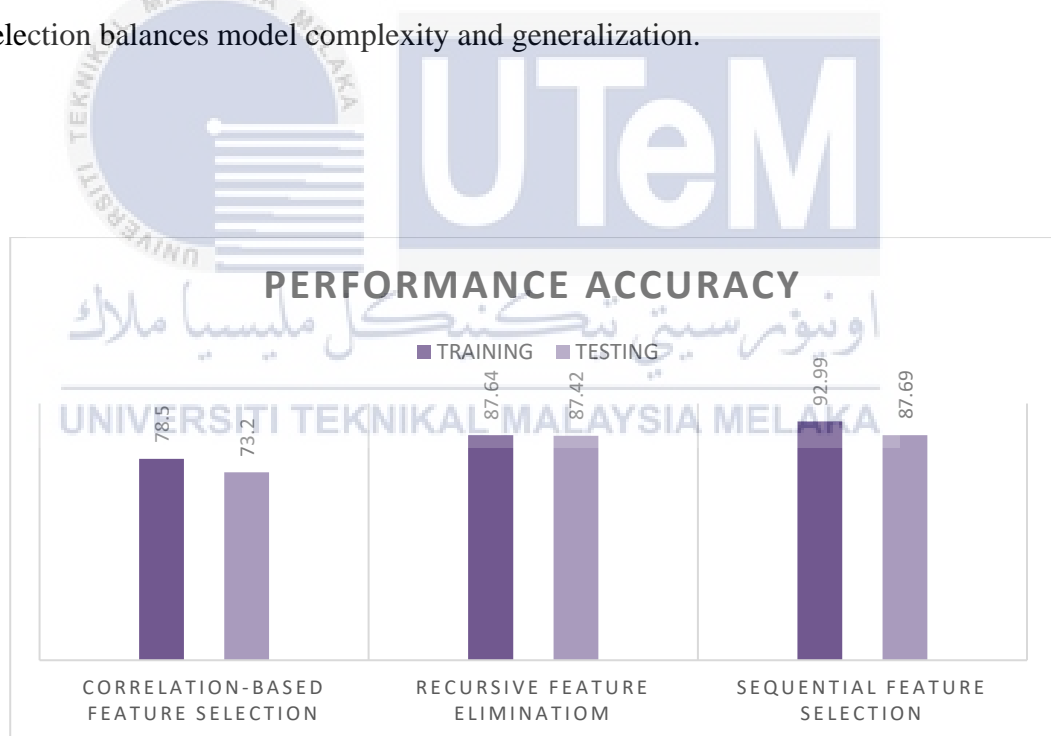


**Figure 4-10: The performance of the selected features.**

Based on Figure 4-10, it represents the heatmap correlation of the Sequential Feature Selection. The correlation heatmap can guide the exploration of feature subsets more efficiently. By considering both the relevance of individual features and their inter-feature relationships, SFS can be more effective in finding a subset that contributes to model performance without unnecessary redundancy. Understanding the correlation between features aids in making informed decisions during the sequential feature selection process. It provides insights into the relationships among variables, contributing to the overall interpretability of the selected feature subset.

#### 4.2 Analyze the accuracy performance of the feature selection method.

Evaluating the effectiveness of the feature selection with different methods is consequential in discovering the most effective approach evolving the dataset's characteristics, modelling goal and computational considerations. This evaluation helps in developing high accuracy, and robust across various settings. Comparing correlation-based feature selection, recursive feature elimination, and sequential feature selection provides insights into their strengths and limitations. Correlation-based feature selection is simple and quick, relying on feature-target relationships. Recursive Feature Elimination is iteratively eliminating features, accommodating complex relationships but may be computationally intensive. Sequential Feature Selection balances model complexity and generalization.



**Figure 4-11: Performance accuracy of different feature selection methods.**

As indicated in Figure 4-11, it presents the comparison of the accuracy performance with different feature selection methods. As referred to Correlation-based feature

selection graph, the training accuracy is 78.5% meanwhile the testing accuracy is 73.2%. This shows moderate accuracy with a relevant feature. However, with moderate accuracy, it might be caused by limitations and less reliability in handling complex relationships. Apart from that, the training and testing accuracy for recursive feature elimination are 87.64% and 87.42% respectively.

As a comparison, recursive feature elimination indicates lower accuracy than correlation-based feature selection. This might be caused by challenges in seeking an optimal subset of features or computationally intensive in refining the model. However, the last method is developed with strong performance, that is sequential feature selection with 92.99% training accuracy and 87.69% testing accuracy. This indicates the effectiveness of the feature selection even the presence of slight drop between training and testing accuracy. Hence, the sequential feature selection denotes robust performances with effective feature selection.

Other than that, every selected feature from every feature selection method is developed in a different method. For instance, the selected features from sequential feature selection method are analyzed using the correlation-based feature selection method. The result is presented in the table below. In accordance with Table 5, CFS stands for Correlation-based Feature Selection, RFE stands for Recursive Feature Elimination and SFS stands for Sequential Feature Selection.

**Table 4-2: Analyzing the selected features with different feature selection methods.**

Features	CFS		RFE		SFS	
	Training (%)	Testing (%)	Training (%)	Testing (%)	Training (%)	Testing (%)
CFS	78.50	73.20	76.30	73.30	87.20	78.5
RFE	89.90	88.97	87.64	87.42	88.78	89.67
SFS	88.33	81.54	89.88	86.15	92.99	87.69

As a comparison between the selected features with different feature selection method, the Sequential feature selection method has manifested the model's performance whereas with different selected features, the accuracy of the training and testing size were still highest than other feature selection methods. This is because sequential feature selection could accommodate with non-linear relationships with the iterative model evaluation, and it is also a model-dependent that allows flexibility in choosing the model. Thus, sequential feature selection is the most suitable approach in developing feature selection in predicting patient's waiting time in clinic.

### 4.3 Develop predictive model.

To optimize the performance of the predictive model, it is subjected to training, evaluation and hyperparameters fine-tuning. Its ability to work with new instances is ensured through thorough validation on a separate test data set. Transparency in terms of interpretability and explanation of predictions are crucial aspects during development as well as accurate documentation regarding capabilities and limitations

throughout the process helps create reliable models that make an impact across different domains. With continuous monitoring post-deployment along with iterative improvements towards sustained accuracy lends value for creating impactful predictive solutions into production environments. In this project, decision tree and logistic regression is the systematic approach in building and optimizing the model.

#### 4.3.1 Decision Tree Model

The decision tree model is developed in JupyterLab notebook. The selected features with the highest accuracy are applied into the predictive model. As discussed, the selected features from the sequential feature selection consist of the highest accuracy. Thus, it is analyzed into the predictive model. In this part, the selected features are trained into decision tree model and the performance of the model will be analyzed and compared using all primary features and selected features. Figure 4-12 shows the predictive model's accuracy in training and testing dataset. The training accuracy indicates 99.55% meanwhile the testing accuracy is 86.59%. This shows that the decision tree model has learned the patterns of the training data effectively rather than testing the data is performed on an unseen dataset.



```

# Train a decision tree model
model = DecisionTreeClassifier(random_state=240)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
y_pred1 = model.predict(X_train)

accuracy = accuracy_score(y_test, y_pred)
accuracy1 = accuracy_score(y_train, y_pred1)

print(f"Testing Accuracy: {accuracy}")
print(f"Training Accuracy: {accuracy1}")

Testing Accuracy: 0.865979381443299
Training Accuracy: 0.9955555555555555

```

**Figure 4-12: Accuracy of the Decision tree model.**

Evaluating and contrasting the effectiveness of a forecast model utilizing all features versus selected features is critical in enhancing its efficiency, and adaptability. Thus, understanding their performance establishes an initial appraisal of their capabilities. On the other hand, models with selected features aim to enhance efficacy by concentrating on significant variables while boosting interpretability and adaptivity. According to Table 6, it presents the performance comparison of the prediction model using primary features and selected features.

**Table 4-3: Performance comparison between primary and selected features.**

Features	Training	Testing
Primary features	99.44%	98.23%
Selected Features	99.55%	86.60%

In accordance with Table 6, the primary features indicate 99.44% accuracy for the training set meanwhile the selected features indicate 99.55% accuracy. This shows that the selected features were successfully trained using the decision tree predictive model by improved accuracy score.

#### 4.3.2 Logistic Regression Model

To optimize model efficiency and interpretability, it is important to analyze and compare the performance of a logistic regression predictive model that uses all features versus one with only selected features. While using all available variables may lead to overfitting, increased complexity, and challenges in interpretation; assessing its performance provides a baseline understanding. Conversely, selecting specific informative variables aims at improving efficiency while promoting an interpretable simplified model.

Moreover, by comparing both models through logistic regression analysis can reveal the impact of feature selection on predictive accuracy as well as evaluate trade-offs between complexity vs interpretabilities. This helps determine how effectively each generalized new data applications. Consequently, conducting such analysis enables informed decision-making about which feature-selection approaches are best suited for purposes resulting in more effective models for any given situation.

```

# Train a model
model = LogisticRegression(random_state=240)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Testing Accuracy: {accuracy}")

y_pred1 = model.predict(X_train)
accuracy1 = accuracy_score(y_train, y_pred1)
print(f"Training Accuracy: {accuracy1}")

Testing Accuracy: 0.8247422680412371
Training Accuracy: 0.8844444444444445

```

**Figure 4-13: Accuracy of the Logistic regression predictive model.**

According to Figure 4-13, it presented the accuracy of the logistic regression model for testing and training size. As referred to the accuracy score, the training accuracy indicates 88.44% meanwhile the testing accuracy indicates 82.47%. Basically, the training accuracy represents the model's performance meanwhile the testing accuracy represents model's generalization to unseen data. As a result, the training accuracy and testing accuracy have positive proximity whereas the trained model reasonably well in generalization. However, the testing and training accuracy are not an exceptional performance due to lower accuracy than the accuracy of decision tree model. On the other hand, to make the model more plausible, it is compared between the primary and selected features.

**Table 4-4: Performance comparison between primary and selected features.**

Features	Training	Testing
Primary features	88.00%	83.51%
Selected Features	88.44%	82.47%

In accordance with Table 7, the primary features indicate 88% accuracy for the training set meanwhile the selected features indicate 88.44% accuracy. This shows that the selected features were effectively trained using the logistic regression predictive model.

### 4.3.3 Comprehensive Analysis

Incomplete analysis of the selected features and prediction model could lead to a misleading and incomplete picture of the model's performance. So, comprehensive analysis is crucial to complete the evaluation of the predictive model along with selecting the most desirable predictive model. As discussed in Predictive model, decision tree and logistic regression are developed to analyze and compare the performance of the predictive model using primary features and selected features. Thus, comprehensive analysis is developed for both predictive models. According to Figure 4-14, it shows the comprehensive analysis for the logistic regression predictive model.

```

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

#Testing set
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

Precision: 0.8625
Recall: 0.9324324324324325
F1 Score: 0.8961038961038962

precision = precision_score(y_train, y_pred1)
recall = recall_score(y_train, y_pred1)
f1 = f1_score(y_train, y_pred1)

#Training set
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

Precision: 0.893048128342246
Recall: 0.9597701149425287
F1 Score: 0.9252077562326871

```

**Figure 4-14: Comprehensive analysis in logistic regression predictive model.**

From logistic regression predictive model, compelling patterns emerge while comparing outcomes from both training and testing phases. The training analysis indicates precision at 89.30%, recall scoring in at 95.98% and F1 score reach up till 92.52%. This shows that the model's performance predicts new data precisely as well as achieving prodigious results making it highly effective. Alternatively, the testing outcomes exhibit a precision of 86.25%, recall value of 93.24%, F1 score rating of 89.61%.

These statistics are indicative of the model's aptitude in handling its specific dataset. The logistic regression model demonstrates outstanding precision, recall and F1 score in both training and testing outcomes. This indicates that the model strikes an appropriate equilibrium between accurately recognizing positive instances while capturing all of them effectively. Furthermore, since F1 score offers a comprehensive measure by considering both precision as well as recall through harmonic mean calculation, it portrays remarkable robustness regarding performance assessment under varied contexts.

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

#Testing Set
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

Precision: 0.9066666666666666
Recall: 0.918918918918919
F1 Score: 0.912751677852349

precision = precision_score(y_train, y_pred1)
recall = recall_score(y_train, y_pred1)
f1 = f1_score(y_train, y_pred1)

#Training set
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

Precision: 1.0
Recall: 0.9942528735632183
F1 Score: 0.9971181556195965
```

**Figure 4-15: Comprehensive analysis in decision tree predictive model.**

According to Figure 4-15, it shows the comprehensive analysis for the decision tree predictive model. From the decision tree predictive model, compelling patterns emerge while comparing outcomes from both training and testing phases. The testing analysis indicates precision at 90.66%, recall scoring in at 91.89% and F1 score reach up to 91.28%. Alternatively, the training outcomes exhibit a precision of 100%, recall value of 99.43%, F1 score rating of 99.71%. Despite the drop of accomplishment, the high precision recall F1 score obtained during testing phase indicates that there are remains relatively balanced ability by the decision tree model. Evaluating precision, recall and F1 score indicates that decision tree predictive model is excessively high performance considering the potential of reducing the overfitting. To make a clear comparison for the comprehensive analysis, Table 8 is implemented.

**Table 4-5: Comparison of the comprehensive analysis.**

Predictive Model	Decision Tree		Logistic Regression	
	Training	Testing	Training	Testing
Accuracy	99.56%	86.59%	88.44%	82.47%
Precision	100%	90.67%	89.30%	86.25%
Recall	99.43%	91.90%	95.98%	93.24%
F1 score	99.72%	91.28%	92.52%	89.61%

Upon comparison of the logistic regression model and decision tree model, the decision tree model achieves a better balance between training and testing performance. This is evident in its high accuracy levels, precision rates, recall scores as well as F1 score across both datasets. On the other hand, although having impressive

training metrics may seem promising for the logistic regression model but it faces hurdles when maintaining optimal results on new data due to overfitting issues.

To sum up, after the comparison between models, it can be stated that decision tree model emerges as an enhanced and better generalizing predictive model. Its performance on both training and testing datasets is well-proportioned indicating adaptability for real-world applications. Even though further improvement with regularization techniques could benefit both models but decision tree model seems to be a more encouraging option for practical deployment since it provides better balance of training and testing efficiency.

#### **4.4 Waiting Time Prediction using Developed Web Application Interface.**

The decision tree predictive model has the potential in enhancing the efficiency of the waiting time prediction. By developing the web application interface, an intuitive design magnifies the user's accessibility. Plus, the web application serves as a user-friendly platform for the administrator to enter relevant parameters, for instance number of patients waiting, type of treatment and fast lane. This web application also helps the patients in contemplating their expected time to get the treatment. The interface layout is presented in Figure 4-16.

← ↻ ⓘ File | C:/Users/User/PredictionGUI2.html

## Waiting Time Prediction at Dental Clinic

**Please Fill in patient's data below:**

**Method: Sequential Feature Selection**

Number of Waiting Patients:

Type of Treatment:

Category of Patients:

Fast Lane:

**Figure 4-16: Web Application Interface.**

#### 4.5 Testing web application to predict Waiting Time.

The developed web application is tested before distributing to the clinics to ensure the usability and the effectiveness of the systems. In accordance with Table 9, it presents the predicted output using decision tree model. Testing the web application helps in verifying the accuracy of the model generated by the application so that the predictive model works as expected. So, in the first test along with Figure 4-17, it is referred to the first row of the predicted output using decision tree model whereas the input for number of patients waiting is 1, the type of treatment is 2 which is scaling, fast lane is 1 represents female below 60, and category of patients is 7 represents adult. The predicted output displays 0 minute to 30 minutes which is in Class 1. This shows that the web application is trained effectively.

However, to assess better functionality, the second test is implemented according to the second in the table. The input data for the second test are number of patients



waiting is 3, fast lane is 1 represents female below 60, type of treatment is 5 which is Fluoride vanish treatment and lastly category of patients is 7 represent adult. As indicated in Figure 4-18, the predicted output displays 30 minutes to 60 minutes which is in Class 2. The predicted output from the web application is simultaneously with the predicted output from the decision tree model.

**Table 4-6: The predicted output using decision tree model.**

Number of patients waiting	Fast Lane	Type of Treatment	Category of patients	Class (Primary)	Class (Predicted)
1	1	2	7	1	1
3	1	5	7	2	2
2	4	1	8	3	3

### Waiting Time Prediction at Dental Clinic

**Please Fill in patient's data below:**

**Method: Sequential Feature Selection**

Number of Waiting Patients:

Type of Treatment:

Category of Patients:

Fast Lane:

Waiting Time Prediction: 0 to 30 Min

**Figure 4-17: Example input data Class 1**

## Waiting Time Prediction at Dental Clinic

**Please Fill in patient's data below:**

**Method: Sequential Feature Selection**

Number of Waiting Patients:

Type of Treatment:

Category of Patients:

Fast Lane:

Waiting Time Prediction: 30 Min to 1 Hour

**Figure 4-18: Example input data Class 2**

## Waiting Time Prediction at Dental Clinic

**Please Fill in patient's data below:**

**Method: Sequential Feature Selection**

Number of Waiting Patients:

Type of Treatment:

Category of Patients:

Fast Lane:

Waiting Time Prediction: 1 hour to 1 Hour 30 Min

**Figure 4-19: Example input data Class 3**

#### 4.6 Environmental and Sustainability.

Predominantly, environmental and sustainability have a wide impact of healthcare whether on the environment or community. In this project, feature selection is developed to create efficient and streamlined administrative processes. By implementing the web application in patient registration, it can significantly reduce paper usage and contributes more to environmental and sustainability. Other than that, predicting patient waiting times in clinic indicates to more energy-efficient clinic design whereas long wait times result to more operational hours. By accurately predicting the waiting times, the operating hours of the clinics can be optimized leading to more energy-efficient in term of power consumption such as lighting and cooling system.

Besides, developing feature selection in predicting patient waiting times leads to sustainable data storage and management. Basically, processing large datasets is involved in feature selection. So, by accomplishing a solution on sustainable data storage, it can be more energy-efficient compared to traditional on-premises servers that contributes to overall sustainability efforts. However, not forgetting the remote collaboration tools for feature selection can reduce physical meetings which promotes a sustainable approach by lowering transportation usage.

Finally, feature selection in predicting patient waiting times is also related to Sustainable Development Goals number 3 which is Good-health and well-being. In accordance with Veterinaria México and Jean-François Guégan (2023) [29], the sustainable development goal number 3 is ensuring healthy life and promoting well-being for all ages. As the goal plays a vital role in establishing policies for sustainable development purposes which frequently results from multiple efforts towards

bettering people's lives. Hence, in this project, feature selection is important in enhancing the healthcare system are efficient, responsive, and patient-centered.

#### 4.7 Chapter summary.

As referred to 4.2 which analyzes the accuracy performance of the feature selection method, the Sequential feature selection method has demonstrated the model's superior performance compared to other methods when comparing selected features. Even with various selected characteristics, this technique maintains higher training and testing size accuracy than alternative approaches due to its ability to adapt non-linear connections through iterative modeling evaluation. Additionally, it allows for flexibility in selecting models because it is dependent on the model employed. Consequently, using sequential feature selection is ideal for predicting a patient's waiting time at a clinic by developing effective feature selection techniques.

From the overall result obtained, this project focused on developing the decision tree model in predicting patient waiting times. As referred to Table 8, the decision tree model demonstrates superior balance between training and testing performance, with high accuracy levels, precision rates, recall scores and F1 score across both datasets. In conclusion of the models' comparison, it can be declared that the decision tree is an improved predictive model for generalization purposes. Its performance implies its adaptability in real-world applications. Although regularization techniques could enhance both models further, the decision tree stands out as a more positive option for practical deployment by providing better equilibrium of efficiency during training and evaluation processes.

Other than that, after meticulous examination, the outcome of the testing has been deliberated in relation to web application creation. The objective behind crafting this particular system is to anticipate patient wait periods using user-entered information. A range of examples have been utilized as input data for assessing and analyzing the functionality of this web interface. By prioritizing ease-of-use and convenience, developers aim to create a more inclusive platform that patrons will enjoy accessing with greater comfort levels overall.



## CHAPTER 5

### CONCLUSION AND FUTURE WORKS



This chapter will discuss the conclusion and recommendations for the future development of this project.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## 5.1 Conclusion.

In conclusion, the first objective referred to identify the important features in predicting patients waiting time in clinics has been effectively achieved by developing the sequential feature selection and decision tree predictive model. The implementation of sequential feature selection enabled numerous feature combinations and selecting the most significant factors in predicting patient waiting times. This resulted in efficiency of the predictive model by determining the important features that played an important role in discovering the accuracy.

Other than that, the second objective referred to analyze the accuracy of selected features using logistic regression and decision tree. Utilizing logistic regression and decision tree predictive model leads to comprehensive analysis of the selected features. Logistic regression is a statistical method in machine learning to predict value from dependent and independent variables meanwhile decision tree model will make predictions based on previous dataset. It offers explicable frameworks and apprehends non-linear interactions among the features.

By employing a blend of sequential feature selection and the decision tree model, crucial aspects that affect patient waiting times were detected alongside an exhaustive evaluation of their effects. This method's intelligibility allowed for simplified comprehension regarding how specific traits led to diversified wait periods leading to concrete recommendations aimed towards facilitating better outcomes in healthcare practices while catering best care services simultaneously under expert administration.

The project's goals were achieved through the implementation of a comprehensive methodology that incorporated feature selection approaches and predictive modeling techniques. The incorporation of logistic regression and decision tree models

strengthened the analysis by facilitating an exhaustive investigation into attribute relevance and correctness. In general, this approach supports streamlining clinic operations by offering a dependable structure for forecasting patient wait times, thereby elevating healthcare delivery proficiency while enhancing overall client satisfaction in return.

## 5.2 Future works.

In this era of rapid IT advancement, the web application design to predict the waiting time helps patients with much work in everyday life. The recommendation that can be suggested is to make notifications from a web application. Deploy the web application publicly after running the software using streamlit command. This can be done, which will include all event alerts. Every notice may or may not have a subject. The system will notify once it detects 10 minutes before getting the treatment. This improvement will alert users to their hospital or clinic appointments.

This project successfully demonstrated that implementing a machine learning model can diversify any circumstance, particularly in healthcare. Moreover, it was able to predict the patients waiting time by using the Decision Tree model build. However, types of treatments for subcategories of TAMPALAN and CABUTAN were not classified while performing pre-processing technique. For future advancement, all those subcategories should be included to give more details and accuracy to predict the best choice of waiting time.



## REFERENCES

- [1] J. Worlitz, L. Hettling, R. Woll, and D. Linh, "Perceived Waiting Time and Waiting Satisfaction: a Systematic Literature Review," in *Conference: 22th Quality Manage. and Organisational Develop. Int. Conf. on Quality and Service Sciences (QMOD/ICQSS '19)*, 2020.
- [2] H. Hijry, R. Olawoyin, W. Edwards, G. McDonald, D. Debnath, and Y. Al-Hejri, "Predicting Average Wait-Time of COVID-19 Test Results and Efficacy Using Machine Learning Algorithms," *International Journal of Industrial Engineering and Operations Management*, vol. 03, no. 02, 2021, doi: 10.46254/j.ieom.20210202.
- [3] H. Kharrazi, C. P. Gonzalez, K. B. Lowe, T. R. Huerta, and E. W. Ford, "Forecasting the maturation of electronic health record functions among US hospitals: Retrospective analysis and predictive model," *J Med Internet Res*, vol. 20, no. 8, 2018, doi: 10.2196/10458.
- [4] R. P. Cavalcanti *et al.*, "Factors associated with the waiting time for access to specialized oral healthcare services in Brazil," *Community Dent Oral Epidemiol*, vol. 50, no. 1, 2022, doi: 10.1111/cdoe.12720.

- [5] J. Brownlee, "Data Cleaning, Feature Selection, and Data Transforms in Python," *Proceedings - 2nd International Conference on Informatics, Multimedia, Cyber, and Information System, ICIMCIS 2020*, 2020.
- [6] E. S. Alomari *et al.*, "Malware Detection Using Deep Learning and Correlation-Based Feature Selection," *Symmetry (Basel)*, vol. 15, no. 1, 2023, doi: 10.3390/sym15010123.
- [7] Y. Masoudi-Sobhanzadeh, H. Motieghader, and A. Masoudi-Nejad, "FeatureSelect: A software for feature selection based on machine learning approaches," *BMC Bioinformatics*, vol. 20, no. 1, 2019, doi: 10.1186/s12859-019-2754-0.
- [8] L. Sun, S. Fu, and F. Wang, "Decision tree SVM model with Fisher feature selection for speech emotion recognition," *EURASIP J Audio Speech Music Process*, vol. 2019, no. 1, 2019, doi: 10.1186/s13636-018-0145-5.
- [9] A. Rastpour and C. McGregor, "Predicting Patient Wait Times by Using Highly Deidentified Data in Mental Health Care: Enhanced Machine Learning Approach," *JMIR Ment Health*, vol. 9, no. 8, 2022, doi: 10.2196/38428.
- [10] R. J. Palma-Mendoza, L. de-Marcos, D. Rodriguez, and A. Alonso-Betanzos, "Distributed correlation-based feature selection in spark," *Inf Sci (N Y)*, vol. 496, 2019, doi: 10.1016/j.ins.2018.10.052.
- [11] R. C. Chen, W. E. Manongga, and C. Dewi, "Recursive Feature Elimination for Improving Learning Points on Hand-Sign Recognition," *Future Internet*, vol. 14, no. 12, 2022, doi: 10.3390/fi14120352.

- [12] Y. Wang, Y. Xu, Z. Yang, X. Liu, and Q. Dai, "Using Recursive Feature Selection with Random Forest to Improve Protein Structural Class Prediction for Low-Similarity Sequences," *Comput Math Methods Med*, vol. 2021, 2021, doi: 10.1155/2021/5529389.
- [13] Y. Yulianti and A. Saifudin, "Sequential Feature Selection in Customer Churn Prediction Based on Naive Bayes," in *IOP Conference Series: Materials Science and Engineering*, 2020. doi: 10.1088/1757-899X/879/1/012090.
- [14] M. Alimoussa, A. Porebski, N. Vandembroucke, R. O. H. Thami, and S. El Fkihi, "Clustering-based sequential feature selection approach for high dimensional data classification," in *VISIGRAPP 2021 - Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2021. doi: 10.5220/0010259501220132.
- [15] A. Suruliandi, G. Mariammal, and S. P. Raja, "Crop prediction based on soil and environmental characteristics using feature selection techniques," *Math Comput Model Dyn Syst*, vol. 27, no. 1, 2021, doi: 10.1080/13873954.2021.1882505.
- [16] N. Nasir *et al.*, "Water quality classification using machine learning algorithms," *Journal of Water Process Engineering*, vol. 48, 2022, doi: 10.1016/j.jwpe.2022.102920.
- [17] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, "Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation," *Biomed Signal Process Control*, vol. 52, 2019, doi: 10.1016/j.bspc.2017.01.012.

- [18] F. C. Bennis *et al.*, “Improving Prediction of Favourable Outcome After 6 Months in Patients with Severe Traumatic Brain Injury Using Physiological Cerebral Parameters in a Multivariable Logistic Regression Model,” *Neurocrit Care*, vol. 33, no. 2, 2020, doi: 10.1007/s12028-020-00930-6.
- [19] M. Yaseliani and M. Khedmati, “Prediction of Heart Diseases Using Logistic Regression and Likelihood Ratios,” *International Journal of Industrial Engineering & Production Research*, vol. 34, no. 1, 2023.
- [20] V. R. Joseph and A. Vakayil, “SPlit: An Optimal Method for Data Splitting,” *Technometrics*, vol. 64, no. 2, 2022, doi: 10.1080/00401706.2021.1921037.
- [21] M. A. Rainey, C. A. Watson, C. K. Asef, M. R. Foster, E. S. Baker, and F. M. Fernández, “CCS Predictor 2.0: An Open-Source Jupyter Notebook Tool for Filtering Out False Positives in Metabolomics,” *Anal Chem*, vol. 94, no. 50, 2022, doi: 10.1021/acs.analchem.2c03491.
- [22] A. A. Asadi-Pooya, M. Kashkooli, A. Asadi-Pooya, M. Malekpour, and A. Jafari, “Machine learning applications to differentiate comorbid functional seizures and epilepsy from pure functional seizures,” *J Psychosom Res*, vol. 153, 2022, doi: 10.1016/j.jpsychores.2021.110703.
- [23] S. Demir and E. K. Sahin, “An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost,” *Neural Comput Appl*, vol. 35, no. 4, 2023, doi: 10.1007/s00521-022-07856-4.

- [24] P. Qiu and Z. Niu, "TCIC\_FS: Total correlation information coefficient-based feature selection method for high-dimensional data," *Knowl Based Syst*, vol. 231, 2021, doi: 10.1016/j.knosys.2021.107418.
- [25] P. Misra and A. S. Yadav, "Improving the classification accuracy using recursive feature elimination with cross-validation," *International Journal on Emerging Technologies*, vol. 11, no. 3, 2020.
- [26] T. B. Alakus and I. Turkoglu, "Comparison of deep learning approaches to predict COVID-19 infection," *Chaos Solitons Fractals*, vol. 140, 2020, doi: 10.1016/j.chaos.2020.110120.
- [27] B. Johnson, *Visual Studio Code: End-To-End Editing and Debugging Tools for Web Developers*, no. 0. 2019.
- [28] W. Lian, G. Nie, B. Jia, D. Shi, Q. Fan, and Y. Liang, "An intrusion detection method based on decision tree-recursive feature elimination in ensemble learning," *Math Probl Eng*, vol. 2020, 2020, doi: 10.1155/2020/2835023.
- [29] J. F. Guégan, G. Suzán, S. Kati-Coulibaly, D. N. Bonpamgue, and J. P. Moatti, "Sustainable development goal #3, 'health and well-being', and the need for more integrative thinking," *Veterinaria Mexico*, vol. 5, no. 2. 2018. doi: 10.21753/vmoa.5.2.443.

## APPENDICES

### Appendix A: Python Code to Build Logistic Regression Model.

```

#Logistic Regression model
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score
import numpy as np

# Load the selected features dataset
data = pd.read_csv('SFSSelected.csv')

# Perform feature selection
features = data.drop('CLASS', axis=1)
target = data['CLASS']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42)

# Train a model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy, precision, recall and f1 score of the testing size
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

```

## Appendix B: Python Code to Build Decision Tree Model.

```

#DECISION TREE MODEL
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, accuracy_score

# Load the selected features dataset
data = pd.read_csv('SFSSelected.csv')

# Perform feature selection
features = data.drop('CLASS', axis=1)
target = data['CLASS']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42)

# Train a decision tree model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy, precision, recall and f1 score of the testing size
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Acuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

```

## Appendix C: Python Code to Develop Web Application in JavaScript.

```

1 function predictWaitingTime() {
2     var waitingPatients= document.getElementById('waitingPatients').value;
3     var fastLane = document.getElementById('fastLane').value;
4     var patientCategory = document.getElementById('patientCategory').value;
5     var treatmentType = document.getElementById('treatmentType').value;
6
7     // Hardcoded example values for demonstration
8     var inputData = {
9         "waitingPatients": waitingPatients,
10        "fastLane": fastLane,
11        "patientCategory": patientCategory,
12        "treatmentType": treatmentType
13    };
14
15    var waitingTimePrediction = predictWaitingTimeFromModel(inputData);
16
17    document.getElementById('waitingTimePrediction').innerText = waitingTimePrediction;
18 }
19
20 function predictWaitingTimeFromModel(inputData) {
21     var model = loadModelFromPickle('model.pkl');
22     var prediction = model.predict(inputData);
23     return prediction;
24     return "Sample waiting time prediction";
25 }

```

## Appendix D: Python Code to Develop Web Application in HTML.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Waiting Time Prediction at Dental Clinic</title>
7   <link rel="stylesheet" href="CSStyles.css">
8 </head>
9 <body>
10  <div class="container">
11    <h1>Waiting Time Prediction at Dental Clinic</h1>
12    <h2>Please Fill in patient's data below:</h2>
13    <h3>Method: Sequential Feature Selection</h3>
14    <form id="queueForm">
15      <label for="waitingPatients">Number of Waiting Patients:</label>
16      <input type="number" id="waitingPatients">
17    </select><br><br>
18    <label for="treatmentType">Type of Treatment:</label>
19    <select id="treatmentType" name="treatmentType">
20      <option value="1.Tooth Extraction">1.Tooth Extraction</option>
21      <option value="2.Scaling">2.Scaling</option>
22      <option value="3.Tooth Extracting">3.Tooth Extracting</option>
23      <option value="4.APP FS">4.APP FS</option>
24      <option value="5.Fluoride Vanish">5.Fluoride Vanish</option>
25      <option value="6.Endo Exterior">6.Endo Exterior</option>
26      <option value="7.Endo Posterior">7.Endo Posterior</option>
27      <option value="8.Other Surgical">8.Other Surgical</option>
28      <option value="9.Denture Case">9.Denture Case</option>
29    </select><br><br>
30    <label for="patientCategory">Category of Patients:</label>
31    <select id="patientCategory" name="patientCategory">
32      <option value="1.Toddler">1.Toddler</option>
33      <option value="2.Pre-school">2.Pre-school</option>
34      <option value="3.Primary School">3.Primary School</option>
35      <option value="4.Secondary School">4.Secondary School</option>
36      <option value="5.OKU">5.OKU</option>
37      <option value="6.Maternity">6.Maternity</option>
38      <option value="7.Adult">7.Adult</option>
39      <option value="8.Senior Citizen">8.Senior Citizen</option>
40    </select><br><br>
41    <label for="fastLane">Fast Lane:</label>
42    <select id="fastLane" name="fastLane">
43      <option value="1.Female below 60">1.Female below 60</option>
44      <option value="2.Male below 60">2.Male below 60</option>
45      <option value="3.Female above 60">3.Female above 60</option>
46      <option value="4.Male above 60">4.Male above 60</option>
47    </select><br><br>
48    <button type="button" onclick="predictWaitingTime()">Predict</button>
49  </form>
50  <div id="waitingTimeResult"></div>
51 </div>
52
53  <script src="PredictionGUI2.js"></script>
54 </body>
55 </html>

```