School of Engineering
TEMASEK POLYTECHNIC

Diploma in Biomedical Engineering

Diabetes Onset Detection

Submitted By

KWEK TZE XUAN, ONG TONG SING, YONG TZE KYE MATTHEW

Class: PE01

Declaration of Originality

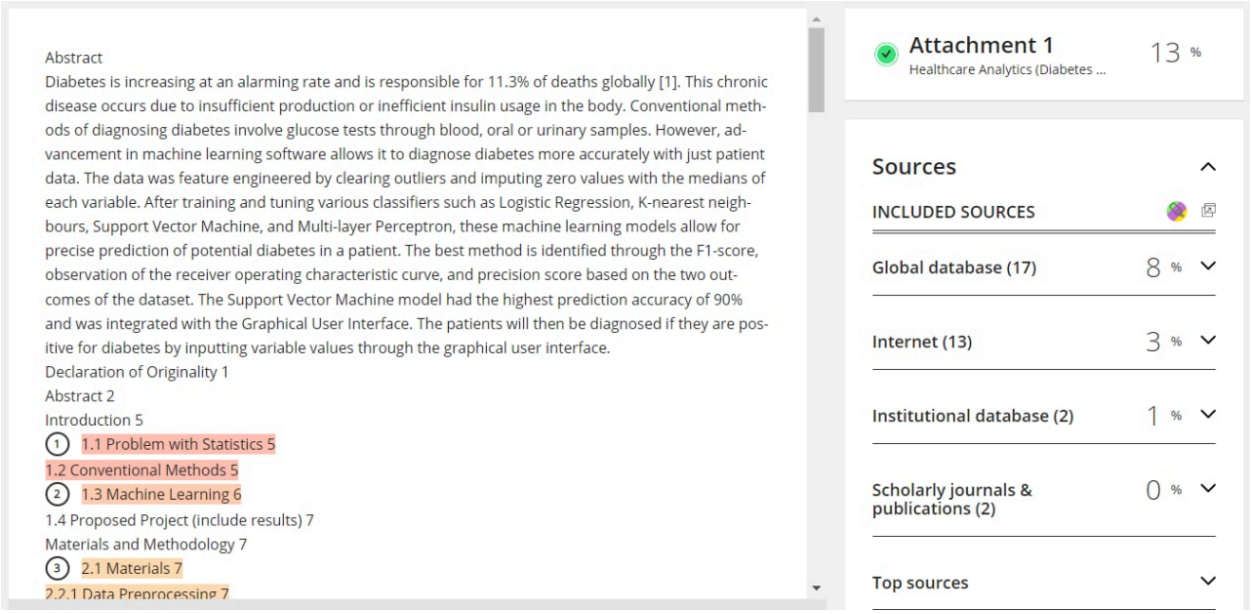
I am the originator of this work and I have appropriately acknowledged all other original sources used in this work.
I understand that Plagiarism is the act of taking and using the whole or any part of another person's work and presenting it as my own without proper acknowledgement.
I understand that Plagiarism is an academic offence
and if I am found to have committed or abetted the offence of plagiarism in relation to this submitted work, disciplinary action will be enforced.

Students' Signatures: 

AY2022/2023 APR SEMESTER

School of Engineering
TEMASEK POLYTECHNIC

SafeAssign Originality Report



Abstract
Diabetes is increasing at an alarming rate and is responsible for 11.3% of deaths globally [1]. This chronic disease occurs due to insufficient production or inefficient insulin usage in the body. Conventional methods of diagnosing diabetes involve glucose tests through blood, oral or urinary samples. However, advancement in machine learning software allows it to diagnose diabetes more accurately with just patient data. The data was feature engineered by clearing outliers and imputing zero values with the medians of each variable. After training and tuning various classifiers such as Logistic Regression, K-nearest neighbours, Support Vector Machine, and Multi-layer Perceptron, these machine learning models allow for precise prediction of potential diabetes in a patient. The best method is identified through the F1-score, observation of the receiver operating characteristic curve, and precision score based on the two outcomes of the dataset. The Support Vector Machine model had the highest prediction accuracy of 90% and was integrated with the Graphical User Interface. The patients will then be diagnosed if they are positive for diabetes by inputting variable values through the graphical user interface.
Declaration of Originality 1
Abstract 2
Introduction 5
1 1.1 Problem with Statistics 5
1.2 Conventional Methods 5
2 1.3 Machine Learning 6
1.4 Proposed Project (include results) 7
Materials and Methodology 7
3 2.1 Materials 7
2.2.1 Data Preprocessing 7

Attachment 1
Healthcare Analytics (Diabetes ... 13 %

Sources

INCLUDED SOURCES

Global database (17) 8 %

Internet (13) 3 %

Institutional database (2) 1 %

Scholarly journals & publications (2) 0 %

Top sources

Abstract

Diabetes is increasing at an alarming rate and is responsible for 11.3% of deaths globally [1]. This chronic disease occurs due to insufficient production or inefficient insulin usage in the body. Conventional methods of diagnosing diabetes involve glucose tests through blood, oral or urinary samples. However, advancement in machine learning software allows it to diagnose diabetes more accurately with just patient data. The data was feature engineered by clearing outliers and imputing zero values with the medians of each variable. After training and tuning various classifiers such as Logistic Regression, K-nearest neighbours, Support Vector Machine, and Multi-layer Perceptron, these machine learning models allow for precise prediction of potential diabetes in a patient. The best method is identified through the F1-score, observation of the receiver operating characteristic curve, and precision score based on the two outcomes of the dataset. The Support Vector Machine model had the highest prediction accuracy of 90% and was integrated with the Graphical User Interface. The patients will then be diagnosed if they are positive for diabetes by inputting variable values through the graphical user interface.

School of Engineering
TEMASEK POLYTECHNIC

Introduction

1.1 Problem with Statistics

Diabetes mellitus, or simply diabetes, is a leading non-communicable disease (NCD) globally, almost quadrupled since 1980 to 422 million adults [2]. It develops when the pancreas produces insufficient insulin or when the body is unable to respond normally to it, causing blood glucose concentration to be unnaturally high [3]. Diabetes is categorized as a chronic illness, and there is no known cure for it [4].

Gestational diabetes is a type of diabetes that transpires when a woman is carrying a child and is recurrent in the last months of pregnancy. It typically disappears once the child is born, but a woman is often likely to have diabetes even after delivery. This prevails if there are any of the following conditions: previous family background of diabetes, the mother is stout, the mother has delivered a baby whose weight was more than four kilograms during birth, the mother has Polycystic ovary syndrome (PCOS), and other such factors [5].

During Pregnancy, the mother's placenta produces more hormones to sustain the pregnancy. These hormones cause cells to become more resistant to insulin. As a result, the pancreas produces extra insulin to compensate for the cell's increased resistance. However, sometimes it is unable to keep up, resulting in excess glucose in the blood and causing gestational diabetes [6].

In 2012 alone, Diabetes caused 1.5 million deaths. Its complications can lead to heart attack, stroke, blindness, kidney failure, and lower limb amputation [7]. Thus it is important to identify diabetes early to reduce the effects and casualties.

1.2 Conventional Methods

Conventional methods refer to the original methods of identifying diabetes without the help of machine learning. These tests were often inconvenient for healthcare workers and patients in terms of time as many tests often required follow-up tests to confirm results and schedule treatment.

Common conventional methods include
the Glycated haemoglobin (A1C) test, which indicates the average blood sugar level of the patient for the past two to three months. This test measures the percentage of blood sugar attached to haemoglobin and does not require fasting [8]. However, it can only accurately diagnose type 2 diabetes and prediabetes

and may give false results in people with certain conditions such as pregnancy and uncommon forms of haemoglobin [9] An A1C level of 6.5% or higher on two separate tests indicates diabetes, while A1C between 5.7 and 6.4% indicates prediabetes and below 5.7% is considered normal.

If the A1C test cannot be used, doctors would recommend these tests below
Random blood sugar test: Blood samples will be taken randomly at different times throughout the day. This method does not require fasting but can be quite biased. A blood sugar level of 200 milligrams per deciliter (mg/dL) — 11.1 millimoles per litre (mmol/L) — or higher suggests diabetes.

Fasting blood sugar test: Blood samples will only be taken after an overnight fast. This method may cause discomfort to the patient, especially those working in labour-intensive jobs. A fasting blood sugar level of less than 100 mg/dL (5.6 mmol/L) is normal. A fasting blood sugar level from 100 to 125 mg/dL (5.6 to 6.9 mmol/L) is considered prediabetes. If it's 126 mg/dL (7 mmol/L) or higher on two separate tests, you have diabetes.

Oral glucose tolerance test: A blood sample will be taken after an overnight fast and measured blood sugar level. A sugary drink is then given to the patient, and blood sugar levels are tested periodically for the next two hours. A blood sugar level less than 140 mg/dL (7.8 mmol/L) is normal. After two hours, a reading of more than 200 mg/dL (11.1 mmol/L) indicates diabetes. A reading between 140 and 199 mg/dL (7.8 mmol/L and 11.0 mmol/L) indicates prediabetes.

If type 1 diabetes is suspected. A urine sample will be tested to identify if any byproducts formed when muscle and fat tissue are used for energy due to the lack of insulin that can utilise the available glucose are present. Doctors will also run tests to identify any potential destructive immune system cells associated with type 1 diabetes called autoantibodies.

Doctors will also test for gestational diabetes if patients are at risk of developing it, such as in patients who are obese during pregnancy and had a previous history of gestational diabetes.

Such tests include:
Initial glucose challenge test: The patient will first drink a sweet glucose solution and have their blood sugar level measured one hour later via a blood sample. A blood sugar level below 140 mg/dL (7.8 mmol/L) is usually considered normal on a glucose challenge test, although result scoring may vary depending on which clinic/lab it is tested at.

Follow-up glucose tolerance testing: This test is similar to the Oral glucose tolerance test, with the main difference being that a higher glucose concentration solution is given to the patient and that their blood sugar level will be checked every hour for a period of three hours.
If at least two blood sugar readings are higher than the normal values established for each of the three hours of the test, the patient will be diagnosed with gestational diabetes.

1.3 Machine Learning

A subfield of artificial intelligence (AI) and computer science called machine learning focuses on using data and algorithms to simulate how humans learn, gradually increasing the system's accuracy. The rapidly expanding discipline of data science includes machine learning as a key element. Algorithms are trained using statistical techniques when fed with data to produce classifications or predictions and to find important insights in the data collected. The decisions made due to these insights influence key growth

indicators in applications and enterprises. Machine learning models comprise three key parts; decision making, error rate, and regularisation layers. The decision-making is where the data inputted into the machine learning model is split into training and testing data. The training data is fit into the model and the training efficacy can be evaluated via different metrics, depending on the type of problem (classification or regression). The predictions are made via inputting the testing data into the trained model. The second part of a machine learning model, an error rate or metric, is used to determine a machine learning model's overall quality, robustness, and accuracy. Lastly, after measuring the accuracy of a machine learning model, one can choose to adjust or tune specific hyperparameters within a machine learning model to improve it [10].

1.4 Proposed Project (include results)

In this project, the Pima Indian Diabetes dataset is used to train different machine learning models to predict the onset of diabetes. The data is first explored and analyzed to discover its nature. The data was found to include many outliers, which were removed using the interquartile range method. It was also observed through data analysis techniques such as measuring the correlation between variables and the outcome and through the use of a univariate statistical test that only half of the features had a strong impact on the target variable, the outcome. They are namely: the number of pregnancies, the plasma glucose concentration, the body mass index, and the age of the women. As such, the other predictor variables were dropped. The data for the positive class was also oversampled via random sampling by duplicating random rows of positive-classed data as there was a significant imbalance between the two classes. After scaling the feature-engineered data via standardization, the data was fed into 4 different machine learning models: a multi-layer perceptron, a support vector machine, a K-nearest neighbours model, and a logistic regression model. (Results). The 4 machine learning models were then imported into a graphical user interface (GUI) to serve as backend models. New data can be inputted into the GUI, which the 4 machine learning models will use to predict the onset of diabetes.

Materials and Methodology

2.1 Materials

The materials used are the Pima Indian Diabetes dataset and Tkinter. The dataset used for this project included 768 patients and 8 numerical features for each.The materials used are Pima Indian diabetes dataset and Tkinter. the dataset used for this project includes 768 patients, 8 numerical features, and one outcome. The numerical variables are Times Pregnant, Plasma Glucose Concentration, Diastolic Blood Pressure, Tricep Skin Fold Thickness, 2-hour Serum Insulin, BMI, and Diabetes Pedigree Function, which are the statistics for their family history, Age, and the binary outcome. All data was taken from the UCI machine learning repository. The dataset is collected from Native Americans known as Pima Indians that reside in Arizona, the USA, and Mexico. it was determined that the group's incidence of diabetes mellitus was high. Hence, the research on them was believed to be reflective of global health. Pima Indian Diabetes dataset is a well-known benchmark that includes females aged 21 and older, and people of underrepresented indigenous or minority communities are included as well [11]. The copy of the dataset used was obtained from Github and published on March 11, 2018 [12].

Tkinter is a way to create a Graphical User Interface (GUI). It was first released in 1991. It is the only framework built into the Python standard library and provides a fast and easy way to create GUI applications as machine learning models are easily imported. To start creating the GUI application, Tkinter

widgets can be used to construct buttons, menus, list boxes, and entries [13]. To store the machine learning models, Python Pickle is used. The final machine learning model will be stored into a variable and called when used to predict the outcome [14].

2.2 Methodology

2.2.1 Data Preprocessing

Data points that are significantly different from the rest of the data set are called outliers. These values in the dataset are capable of distorting any statistical analysis, causing bias-ness, reducing the power of statistical tests, and resulting in misleading interpretations. Firstly, the interquartile range method. IQR method calculates the interquartile range by subtracting the 75th percentile and 25th percentile, then proceeds to find the upper and lower fence. Any value that falls outside of the upper and lower fence will be considered an outlier.

Interquartile range: Q3 – Q1
Upper fence: Q3 + (1.5 * IQR)
Lower fence: Q1 – (1.5 * IQR)

Secondly, the Z-score method. Z-score is the measure of the distance a data point is from the mean, in standard deviations. By setting the Z-score threshold to 3, which is a standard value, any value lesser than 3 will be removed from the dataset as an outlier.

Z score = (x -mean) / std. Deviation

Missing values in the dataset will affect the machine learning model's accuracy as it is learning from an incomplete dataset. Mean, median, and mode are the three primary missing value imputation strategies. Mean is the average of all values in the dataset, the median being the middle number in a sorted number by size, and mode being the most common numerical value. When managing skewed data, the median and mode are more appropriate than the strategy of replacing missing values with the mean value [15].

2.2.2 Multi-layer Perceptron (MLP)

A multi-layer perceptron is a feed-forward neural network where inputs are pushed through the multi-layer perceptron by taking the dot product of the weights before the hidden layers and the input layer [16]. The inputs are fed into the input layer and multiplied with their individual weights before entering the adjacent hidden layer. A summation of the weights with the biases for every neuron in the hidden layer is made before an activation function is implemented. The activation function then determines if the neuron can be fired to give a result. Below is an illustration of the methodology of a multi-layer perceptron:

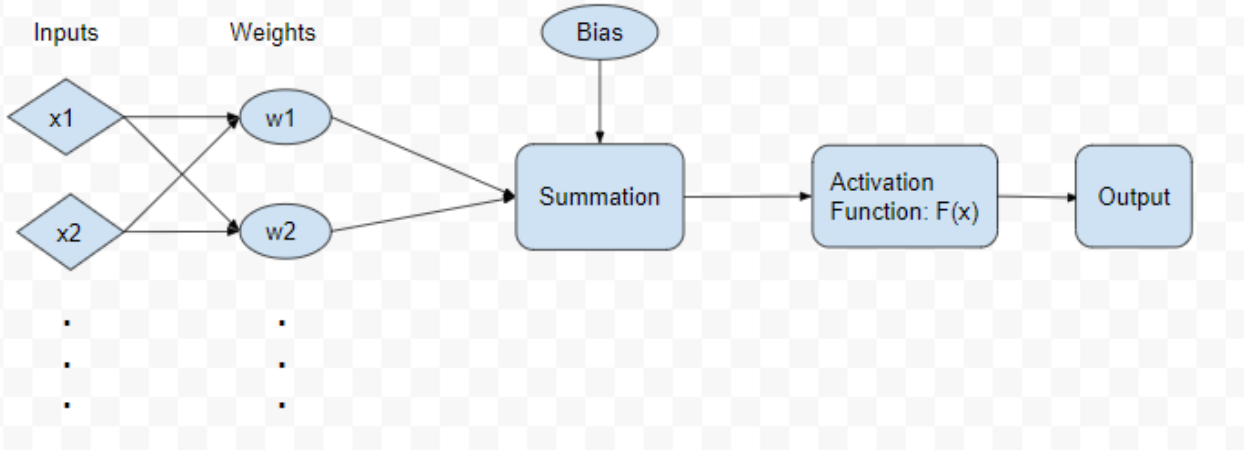


Figure 1: Conceptual Flow of Multilayer Perceptron

The summation of the weights and biases is given by the following formula for one hidden neuron: (x1*w1)+(x1*w2)+(Bias). In this project, the Rectifier Linear Unit (ReLu) activation function is used for the hidden layers to fire the hidden neurons. This is because it is simple to compute yet able to solve complex problems since it is non-linear. ReLu applies the concept of checking if the summation output value is below or above zero as shown by the graph below [17]:

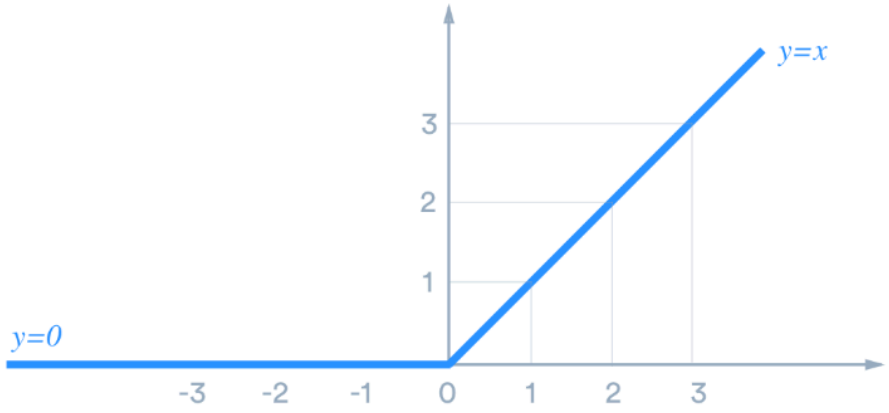


Figure 2: Graph of ReLu activation

For the output layer, the sigmoid function will be used as the output of the function always gives a value of between 0 and 1. This can be explained by observing the formula and the graph below, where the denominator is always greater than 1:

$$S(x) = \frac{1}{1 + e^{-x}}$$

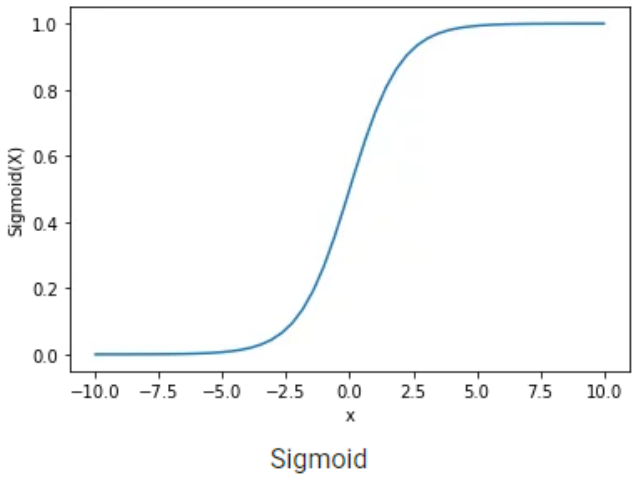


Figure 3: Formula and graph for Sigmoid [18]

The Sigmoid activation function is very relevant also since the loss function used will be Binary Crossentropy because the classification problem is a binary one between two outcomes: Positive or Negative. Binary Crossentropy utilizes two logarithmic functions, one for 0 values and one for 1 values in terms of probability, given by the formula below:

$$l = Log(L) = \sum_{i=1}^{N} [x_i Log(\mu) + (1 - x_i) Log(1 - \mu)]$$

Figure 4: Formula for Binary Crossentropy [19]

2.2.3 Support Vector Machine (SVM)

Support Vector Machine uses classification algorithms to perform two-group classification. SVM's parameters are C, kernels (Linear, Non-Linear, Polynomial, Radial Basis Function, Sigmoid, etc), gamma (applicable only to Radial Basis Function, Polynomial, and Sigmoid kernel function), and degree (only applies for Polynomial kernel function). C value determines the influence of the misclassification by informing the SVM optimization how much to avoid misclassifying each training example. The gamma determines how far the influence of a single training example reaches. Low values mean it's far while high values mean it's close. The degree parameter controls the flexibility of the decision boundary. Higher degree kernels mean a more flexible decision boundary. It takes down the data points and plots a hyperplane that best separates the tags. Hyperplanes are considered best when it separates the two class by a large margin as seen in figure 6, it has a bigger margin as compared to the one in figure 5. Predictions are based on the side of the hyperplanes that new data will fall into when it is mapped into the same space.

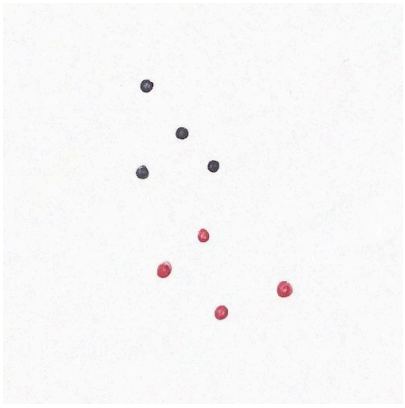


Figure 4: Example Data Points

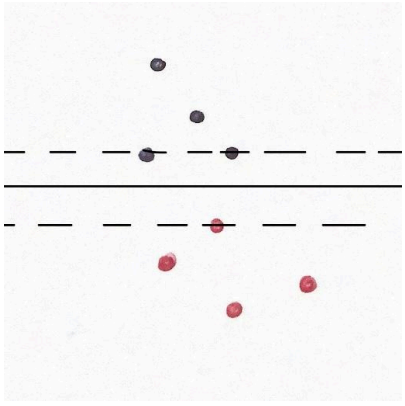


Figure 5: Normal Hyperplane

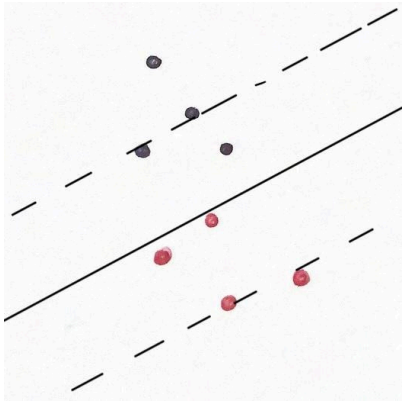


Figure 6: Best Hyperplane

2.2.4 K-Nearest Neighbours Model (KNN)

KNN or k-nearest neighbours is a supervised learning classifier that makes classifications and predictions based on the proximity of individual data points. It finds the distance between a query and all the examples in the data and selects the specified number of examples (k-value) closest to the query and

votes for the most frequent label or classification [20].
It is often used as a classification algorithm, working off the assumption that similar individual data points can be found close to each other. Usually, k-values should not be too small or too big, leading to inaccurate results.

For calculating distances, KNN uses a distance metric from the list of available metrics. Distance metrics can be changed to suit the dataset.

Minkowski distance: It is a metric intended for real-valued vector spaces. Minkowski distance can only be calculated in a normal vector space, where distance can be represented by a vector that has a length, and the length cannot be negative. The formula below is the generalised form and can be manipulated to get different distance matrices. When changing the p-value in the formula, different distance metrics such as Manhattan Distance and Euclidean distance can be achieved

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \qquad d = \sum_{i=1}^{n} |x_i - y_i| \qquad d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Figure 7, 8, 9: Different distance metric formulas, left: Minkowski distance, middle: Manhattan Distance, right: Euclidean distance [21]

When p is set to 1, Manhattan Distance is achieved, this distance is also referred to as taxicab distance or city block distance due to the way distance is calculated. It is the sum of the absolute difference of their coordinates, in other words, the sum of the adjacent and opposite lengths. This distance metric yields better accuracy than Euclidean distance in cases of high dimensionality.

When p is set to 2, Euclidean distance is achieved. This distance metric is the most used as it measures the true straight line distance between two coordinates, in other words, the hypothenuse length. It is used as the default metric that the SKlearn library of Python uses for K-Nearest neighbour

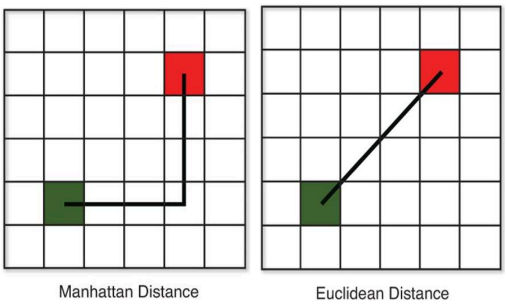


Figure 10,11: Illustration of Manhattan and Euclidean distance [21]

2.2.5 Logistic Regression Model

Logistic regression is a supervised learning algorithm that works based on the sigmoid function of $f(\mathbf{x})$: 1 / (1 + exp(x). As such, the result is often close to either 0 or 1. The function is often interpreted as the predicted probability that the output for a given $\mathbf{x}$ is equal to 1. Therefore, 1 − f(x) is the probability that the output is 0. What this means is that the model will classify all values above the middle value as 1 and all values under it as 0. This makes logistic regression very suitable for binary classification.

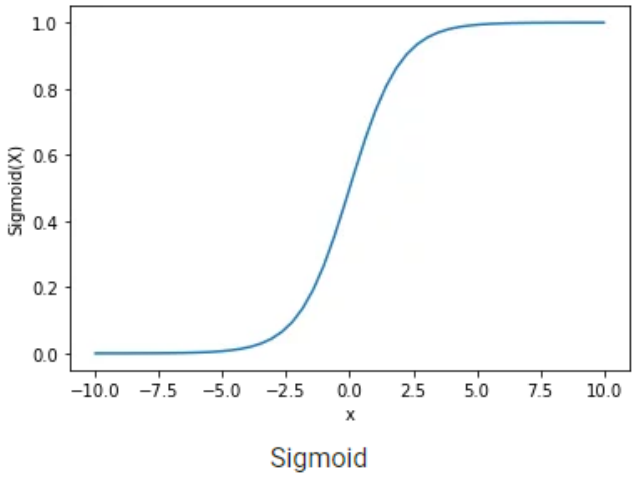


Figure 12: Sigmoid function [18]

2.2.6 Grid Search and Stratified K-Fold Cross Validation

Grid Search is a hyperparameter tuning tool to optimise machine learning models in an attempt to improve the models. Different machine learning models have different hyperparameters for training with the addition of relational hyperparameters, whereby some of these hyperparameters improve the model when trained together. Therefore, the trial and error process can be tedious, and Grid Search is an automated tool for searching for the best hyperparameter combinations. Afterwhich cross-validation increases the reliability by Grid Searching the hyperparameters on more data splits.
Cross Validation refers to when the dataset is split up randomly into 'k' groups. One group is then used as the test set, and the rest are assigned as the training set.

The model is trained based on the training set and verified with the test set. This process is repeated until each group is used as the test set.
For this project, cv has been set to 5, which splits the dataset into 5 groups. The model would then be tested and trained 5 separate times, as seen in the picture below,

Training data	Test data

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 4

Figure 13: Cross-validation illustration

Cv values above 5 are usually not used as they are often time-consuming and computationally expensive. More details will be covered for all the machine learning models used [22].

Results

3.1 Data Preprocessing

3.1.1 Clearing outliers

To detect the outlier from each variable, histogram and Box and Whisker plots are used. Two methods were experimented with to remove the outliers: Interquartile Range (IQR) and Z-score. The options were evaluated before deciding which strategy was most appropriate and effective for the given data. As seen in table 1, the interquartile range and lower and upper bound of each variable are calculated. The values from the predictor variables that fall outside the lower and upper bound will be removed as outliers. For the z-score, each numerical value in the predictor variable will replace the 'x' in the z-score equation, and then compare with the threshold set. The rows containing any outliers will be removed from the dataset. Comparing the two methods of clearing outliers, table 2 provides the count of rows before and after the respective methods. Figures 7, 8, and 9 below are the histogram of the original dataset, the dataset after outliers removal through the IQR method, and the dataset after outliers removal through the z-score method.

Table 1: Data for Interquartile Range Method

Predictor Variables	Interquartile Range	Lower Bound	Upper Bound
Times Pregnant	5.0	-6.5	13.5
Plasma Glucose Concentration	41.25	37.125	202.125
Diastolic Blood Pressure	18	35.0	107.0
Triceps Skin Fold Thickness	32.0	-48.0	80.0
2-hour Serum Insulin	127.25	-190.875	318.125
BMI	9.3	13.35	50.55
Diabetes Pedigree Function	0.383	-0.329	1.2
Age	17	-1.5	66.5

Table 2: Rows Before and After Clearing Outliers

	Before Clearing Outliers	**After IQR Method**	**After Z-score Method**
Number of Rows	768	639	688

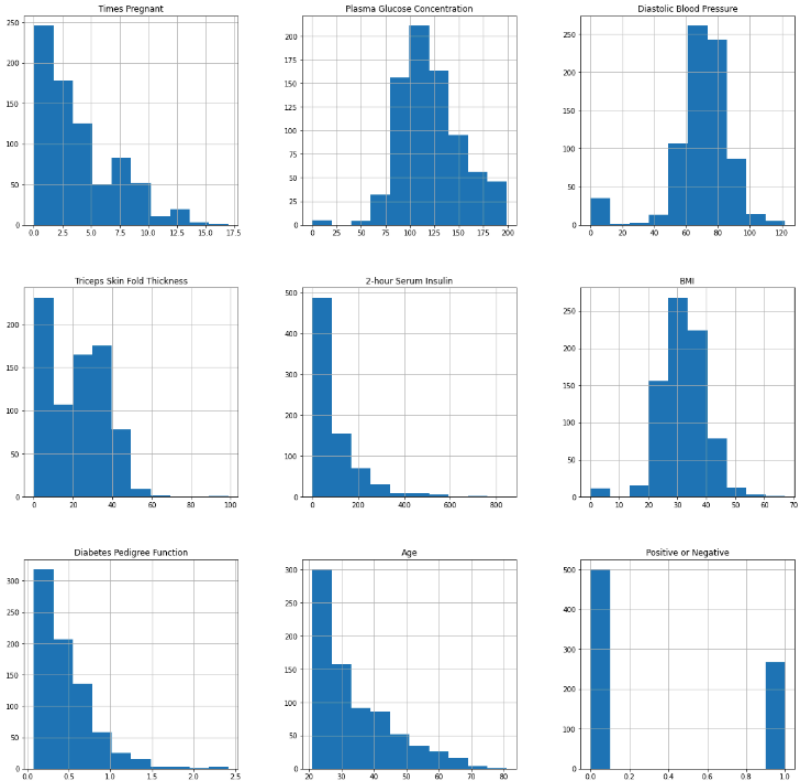


Figure 14: Spread of Variables Before the Removal of Outliers

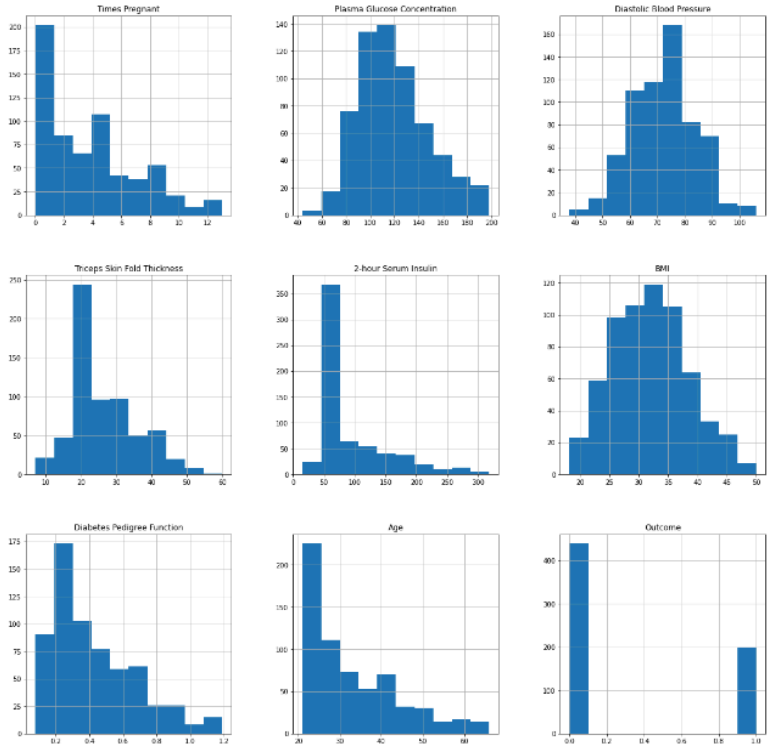


Figure 15: Spread of Variables After the Removal of Outliers (IQR method)

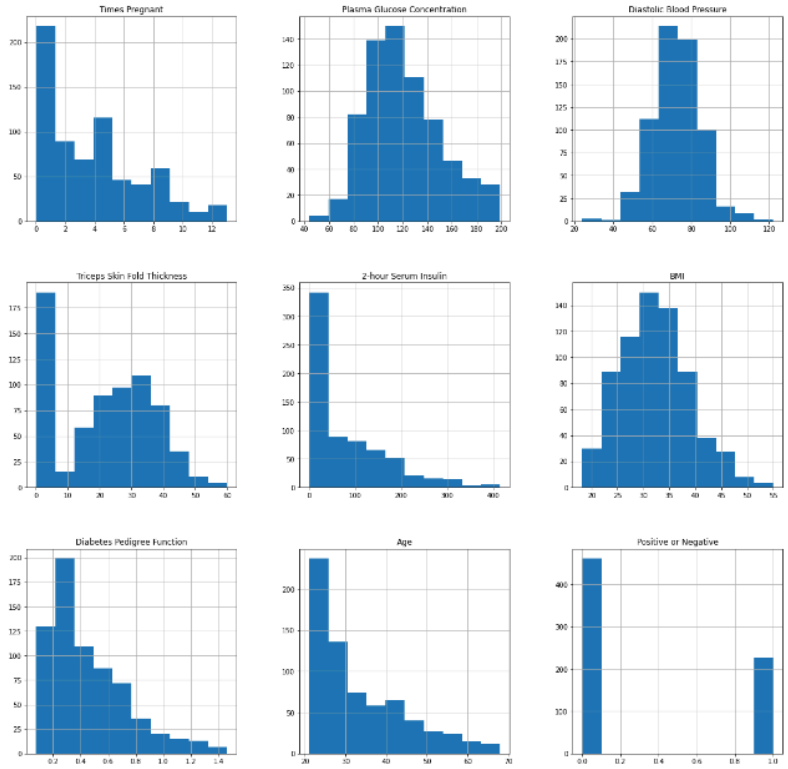


Figure 16: Spread of Variables After the Removal of Outliers (Z-score method)

3.1.1.1 Comparison of the Methods

It can be seen that the Interquartile Range (IQR) method clears more outliers as compared to the Z-score method as shown in the Box and Whisker Plot of the df_cleaned and new_df. Even though both methods eliminate relevant outliers, the IQR approach removes more outliers and concentrates on significant data.

3.1.2 Replacing 0s in Predictor Variables with Median

The predictor variables that contain zeros are 'Times Pregnant', 'Tricep Skinfold Thickness', and '2-hour Serum Insulin' with 88, 179, and 307 counts of zeros in the respective variables. The ‘Times Pregnant' variable will be ignored when it comes to the variables that contain 0 in it as it is normal to have 0 times pregnant. Focusing on the ‘Tricep Skinfold Thickness’ and ‘2-hour Serum Insulin’, a histogram will be plotted for each variable to see the spread. The method to remove the missing values will be in reference to the spread of the variable. as shown on the histogram, both data are skewed to the right. Hence, the median will be used to replace the missing values. The median is 23 and 65.93**,** for ‘Tricep Skinfold Thickness’ and ‘2-hour Serum Insulin’ respectively.

Table 3: Number of 0s in each variable

Predictor Variables	Count
Times Pregnant	88
Plasma Glucose Concentration	0
Diastolic Blood Pressure	0
Tricep Skin Fold Thickness	179
2-hour Serum Insulin	307
BMI	0
Diabetes Pedigree Function	0
Age	0

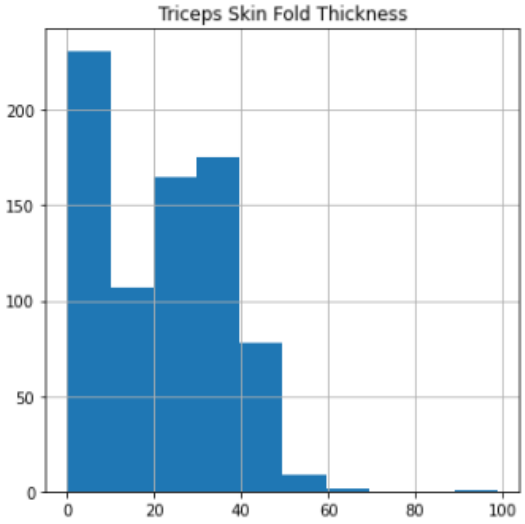


Figure 17: Histogram of Triceps Skin Fold Thickness

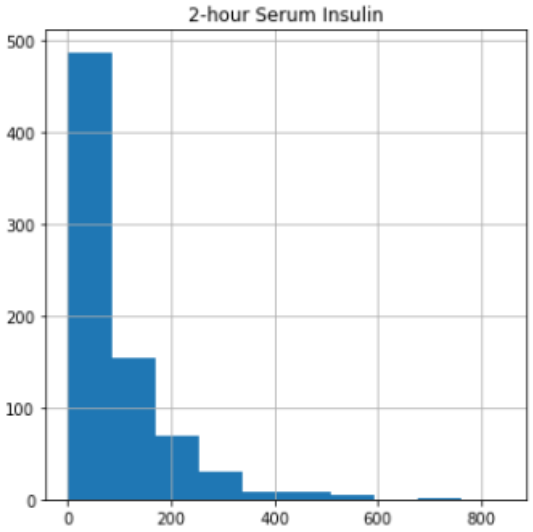


Figure 18: Histogram of 2-hour Serum Insulin

These 0 values may not be realistic for ‘Tricep Skin Fold Thickness’ and ‘2-hour Serum Insulin’. On the other hand, a woman can have 0 pregnancies. In this case, a 0 value is realistic.

Table 4: Median of Predictor Variables

Predictor Variables	Median
Tricep Skin Fold Thickness	23.0
2-hour Serum Insulin	65.9311

3.1.3 Feature Selection

Feature selection is needed to filter out unnecessary variables that may decrease the performance and accuracy of the prediction model. Different types of data have better feature selection methods depending on the data's nature. For Pima's Diabetes Dataset, the predictor variables or input variables are numerical and the target variables or output variables are numerical as well. After deciding on the features to be used for the prediction model, the other features are dropped via indexing.

3.1.3.1 Feature Selection using Pearson's Correlation Coefficient

The correlation values between the predictor variables and the target variable are generally quite low, except for Plasma Glucose Concentration having a correlation value of 0.49 with the Outcome. After careful analysis, it is observed that the data for Times Pregnant, Plasma Glucose Concentration, Tricep Skinfold Thickness, BMI, and Age are the variables that have a medium correlation with at least one other predictor variable. However, the correlation value between Tricep SkinFold Thickness and the Outcome is at a low of 0.15 despite its medium correlation value of 0.57 with the BMI predictor variable. Henceforth, the relationship between the data for Tricep SkinFold Thickness and the Outcome is weak and will not be selected as a feature for the model. To ensure greater reliability in this method of feature selection (since the correlation values are relatively weak), a univariate statistical test and violin plots are used to select the best features as well for the prediction model.

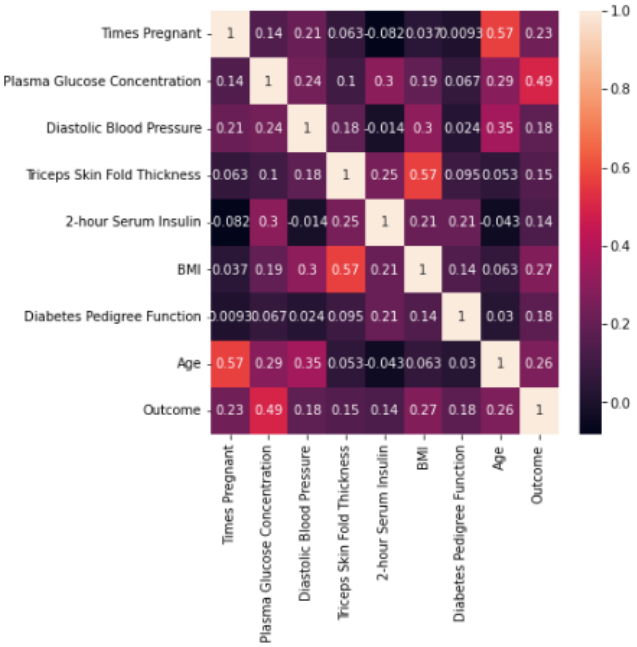


Figure 19: Heatmap with Correlation Values

3.1.3.2 Feature Selection using Univariate Statistical Test

There are many advantages to the use of Univariate Statistical Tests, namely: reducing overfitting of the data on the prediction model by removing noise or irrelevant data, increasing precision and accuracy as the data used will be less misguiding, and reducing the training duration of the prediction model since there is fewer data to process. The cleaned dataset is first split into X and Y, with the predictor variables in X and the target variable in Y. The select K best method from scikit learn is done using ANOVA F-value

test. The f_classif function for the SelectKBest method carries out the Anova F-value test for the variables. The F score is a scoring metric to show how well a certain feature discriminates between two classes. The K score formula is given as F score = (The distance between means of class distributions) / (variance of every single class), which in layman's terms means the distance between the classes divided by the compactness of the classes. The SelectKBest method can then be specified to select the best K number of variables for usage. The scores are then displayed in descending order in the following table:

Table 5: K scores for Predictor Variables

Predictor Variables	K scores
Plasma Glucose Concentration	204.2492
BMI	49.8375
Age	47.4678
Times Pregnant	34.9252
Diastolic Blood Pressure	22.1460
Diabetes Pedigree Function	21.1033
Triceps Skin Fold Thickness	14.8654
2-hour Serum Insulin	12.1999

3.1.3.3 Feature Selection using Violin Plots

Lastly, to confirm the features selected, violin plots are used to show any significant differences between the plots for the variables for both binary outputs. If the medians and the shapes of the violin plots are generally different between the outputs for each variable, then the particular variable is selected as a feature for the prediction model. Furthermore, the shapes of the violin plots for a positive outcome, which indicate the spread of the data, are significantly different from those of the negative outcome. This proves that the predictor variables below directly affect the outcome since the nature of the data for these predictor variables differ greatly for each binary output.

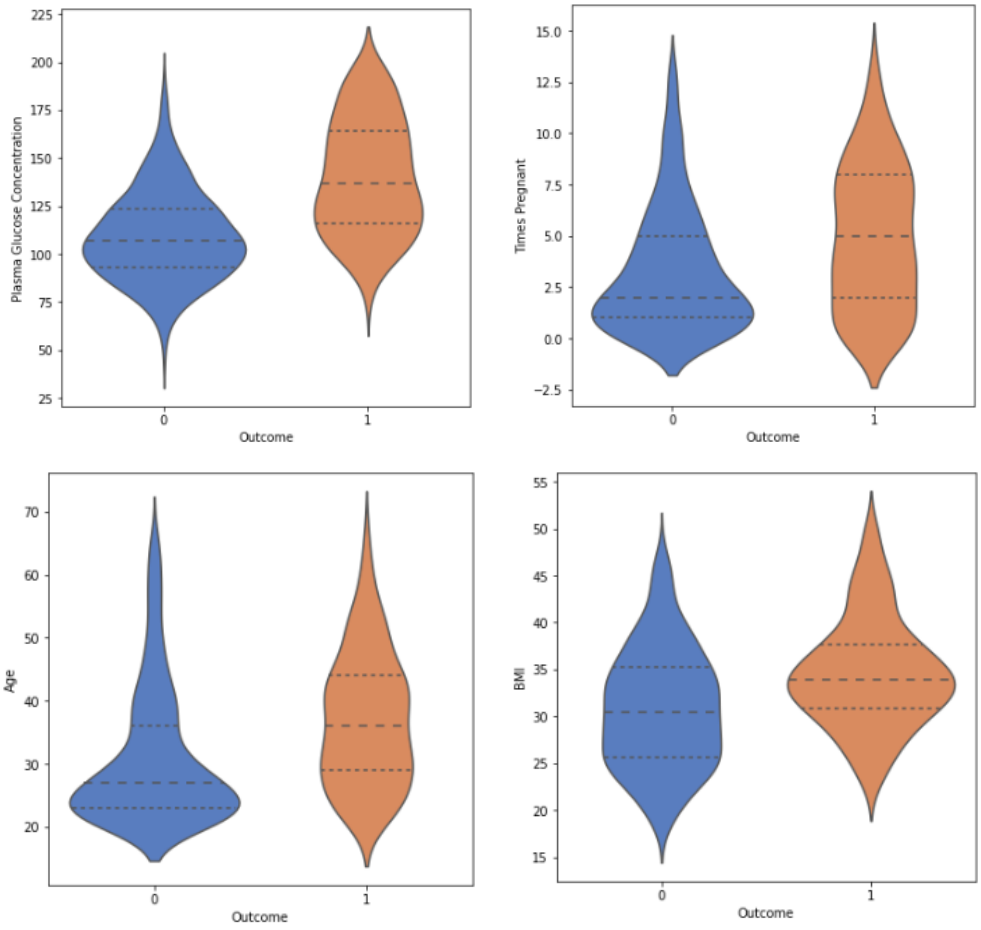


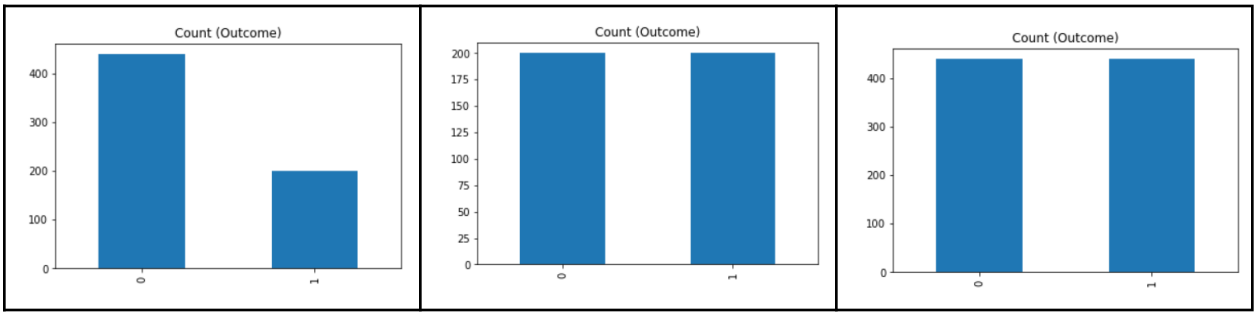
Figure 20: Violin plots to show frequency and distribution

3.1.4 Balancing Outcome

For a classification model to produce a greater accuracy model, higher balanced accuracy, and also a higher detection rate, the data set must be well-balanced. The current cleaned data has 439 negative and 200 positive outcomes. As a result, the classification process will be influenced since machine learning classifiers tend to be more biased towards the majority class, which results in the inaccurate categorization of the minority class. The data collection should be resampled using the Resampling Technique to resolve this problem. This technique under-samples the majority class by removing samples from it or by increasing the number of examples from the minority class (over-sample).

Table 6: Count of Outcome Before and After Balancing

Unbalanced	Undersampled	Oversampled



In order to achieve a balance between the majority and minority classes, random undersampling removes observations from the majority class. Data of the negative outcome was removed from 439 to 200 observations, a total of 239 observations removed. On the other hand, until the number of observations equals that of the majority class (439), copies of the minority class are added to the initial number of observations (200). All three data sets, unbalanced, undersampled, and oversampled will be further tested with the different machine learning models.

3.2 Multi-layer Perceptron

3.2.1 Flow of Tuning Multi-layer Perceptron

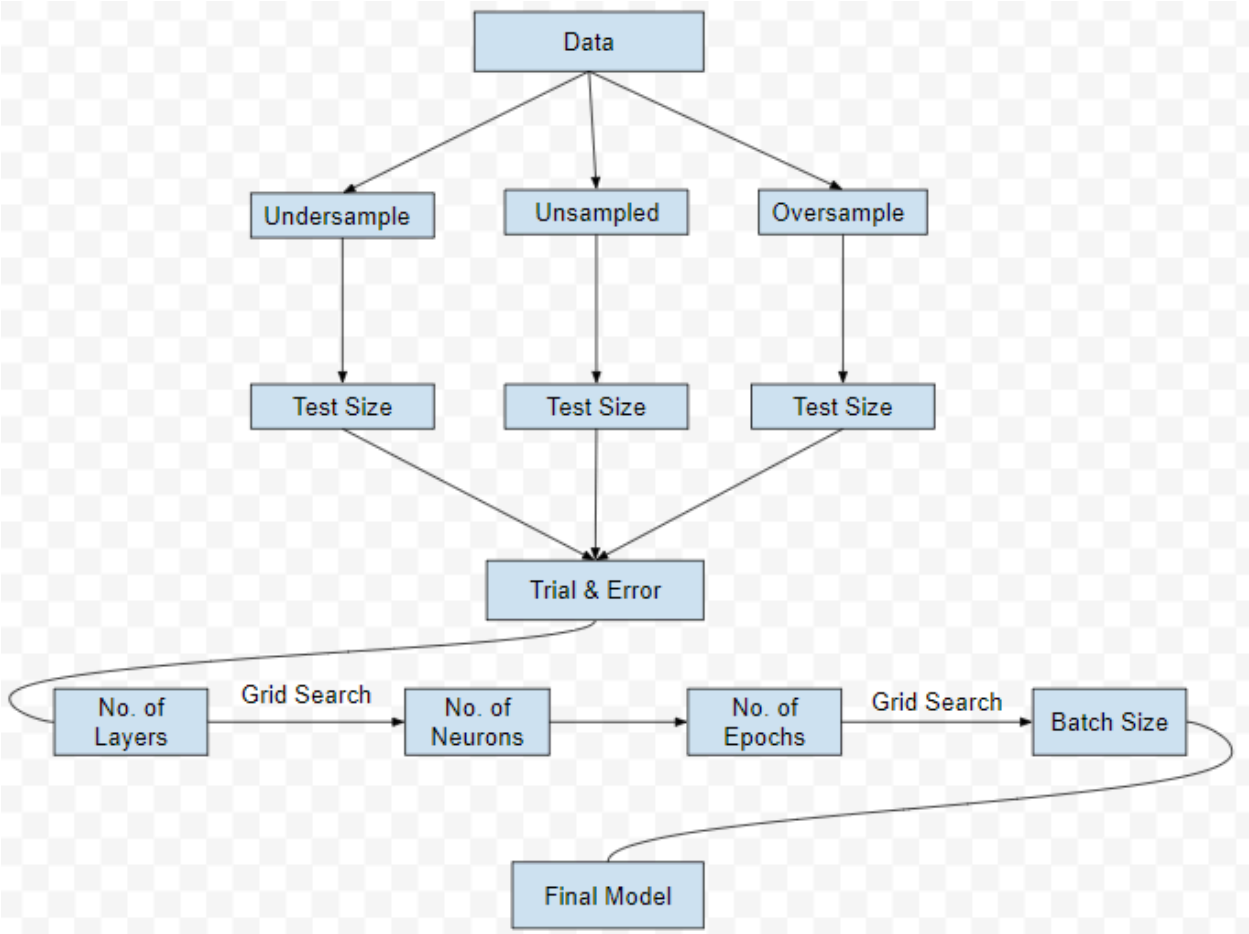


Figure 21: Steps for tuning the Multi-layer perceptron

The data will first be tested across different test sizes and sampling types. Afterwhich, trial and error of the parameters will be conducted. Grid Search is performed to obtain the best hyperparameter values for tuning the model.

3.2.2 Different testing sizes and sampling types

Before the standardisation of data, the selected features are labelled as X while the data for the Outcome is labelled as Y. Next, the data is standardised, and trial and error of the data are performed to determine the best training to test data split ratio. A basic deep learning model with untuned hyperparameters is used and tested across the following ratios of training data to testing data: 75%:25%, 80%:20%, and 85%:15%. The generic model has one input, hidden, and output layer. The input layer has 12 neurons, the hidden layer has 9 neurons, and the output layer is 1 neuron. The optimizer used is the most basic: Stochastic Gradient Descent (SGD), with the epochs set to 150 and a batch size of 16. The above will be carried out for oversampled, unsampled, and undersampled data.

Appendix A, tables 1, 2, and 3 show the training and validation accuracies and losses for the different data split ratios and sampled data. When the data has been oversampled, the overall accuracies are higher. Hence, the training and testing data split ratio of 75%:25% will be used on the oversampled data.

3.2.3 Different number of layers

Next, trial and error will determine the best number of hidden layers for the model. Since the number of input variables is 4, the number of neurons in the input layer will be kept at numbers that are multiples of 4 and powers of 2. The output layer will consist of only one neuron since there is only one target variable. When the number of hidden layers increases, the input layer is updated so that the number of neurons per hidden layer decreases along the neural network. Trial and error were conducted twice for the following optimizers: Stochastic Gradient Descent (SGD) and Adam. The different accuracies and losses can be found in Appendix A, tables 4 and 5. It is observed that when using the optimizer SGD, the average loss difference across the experimented number of layers is less than the average loss difference when the optimizer Adam is used. However, several other factors must be taken into account. It is observed that the training accuracies for the model when the optimizer Adam is used is much higher than the training accuracies of the model when the optimizer SGD is used. Both optimizers will continue to be used for testing the next few hyperparameters.

On the other hand, it is observed that regardless of the optimizer used, the training accuracies increase with an increasing number of hidden layers in the model while the validation accuracies decrease. Hence, the validation losses are larger with an increasing number of hidden layers in the model, indicating overfitting of the validation curves. Fortunately, this issue can be easily fixed by adding regularisation layers in the model.

Grid search will be performed to find the optimal dropout value and rate for each layer with the addition of Max Normalisation weight constraints. The incoming weights for the neurons in each next hidden layer were constrained using the Max Norm class of weight constraints. The weights may rise as backpropagation begins to update them, overfitting the training dataset. When training a neural network, weight regularisation techniques like decay add a penalty to the loss function to motivate the network to adopt modest weights. Weight constraint, a trigger that examines the weights' size or magnitude and scales them so they are all below a predetermined threshold, was chosen as the regularisation approach other than dropout. When used with other regularisation techniques like dropout, weight limitations reduce the generalisation error of the model [23].

The best dropout rates and weight constraints with their respective accuracies can be found in Appendix A, tables 6 and 7. Afterwhich, the models are updated with the best dropout rates and kernel constraints to reduce the overfitting of the models. The models can now be compared to determine the best number of hidden layers. After observing the accuracies and losses given in Appendix A, tables 8 and 9 while considering a balance between minimal losses and good accuracies, Neural Structures A4 and S4 are selected for further testing involving other parameters and hyperparameters.

3.2.4 Different number of neurons

Previously, the number of neurons used per layer was in multiples of 4. The number of neurons per hidden layer is labelled as the model's width. General rules for the width of each layer are [24]:

1. The magnitude of the widths must lie between the widths of the input layer and the output layer.
2. The magnitude of widths is recommended to be ⅔ the width of the input layer plus the output layer.
3. The hidden layers should have narrower widths than twice the width of the input layer.

The diagram illustrates the architecture of the multi-layer perceptron models chosen.

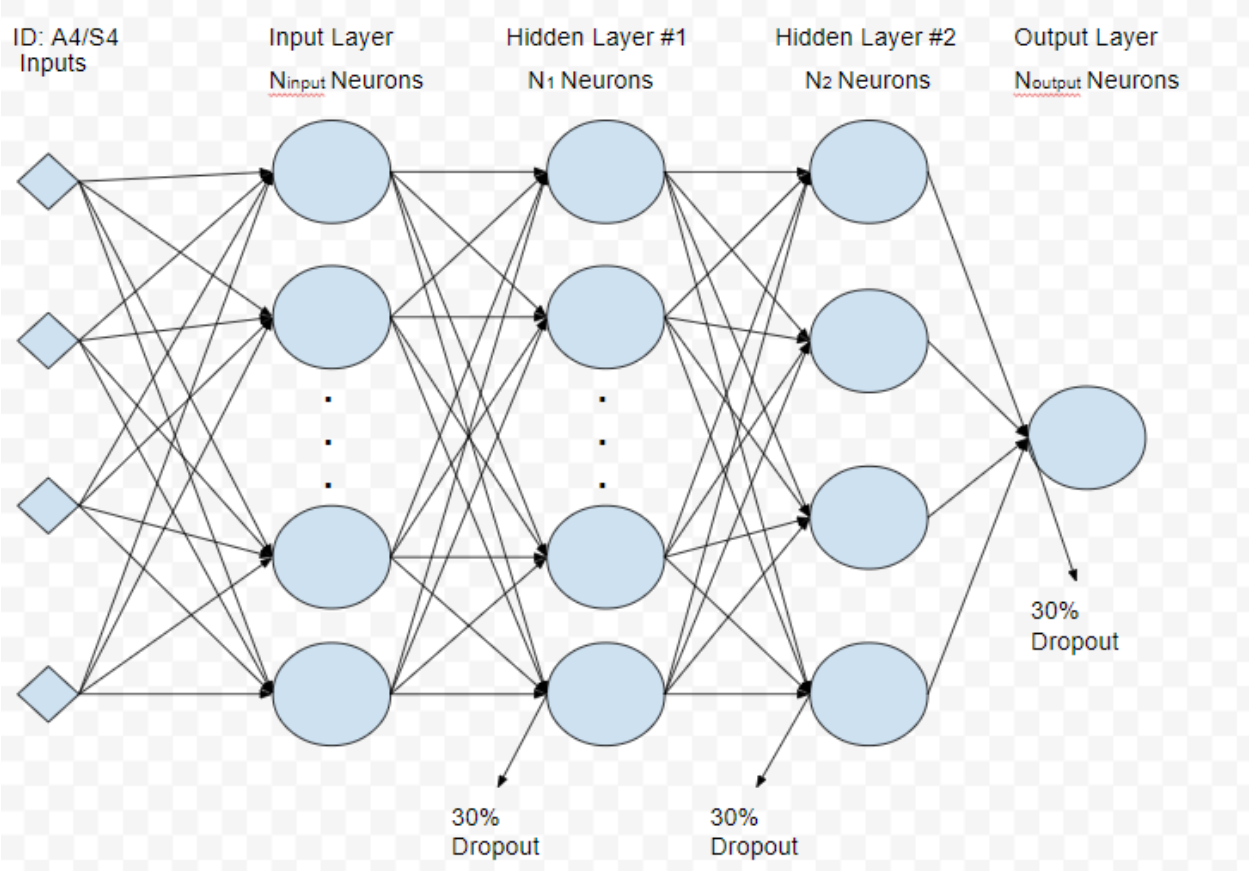


Figure 22: Visualisation of MLP Neural Architecture ID: A4/S4

As mentioned, the number of neurons per layer is in multiples of 4 and powers of 2 and the deeper the hidden layer, the narrower the width. The widths of the architecture are in descending powers of 2 down the neural network. Rule 2 is also applied. If the number of neurons in the input layer is not divisible by 3, the quotient is rounded to the nearest whole number. The range of the number of neurons across the different number of layers will also be set to be large and small, to compare the effect of large and small decrements of neurons between consecutive layers with the increasing depth of the model. These comparisons are listed in Appendix A, tables 10 and 11. The table column ‘number of neurons’ follows the order illustrated in Figure 1 above, where each number is the number of neurons for one layer. The output neuron is kept at 1. All the variations made to the model are also repeated for it when the optimizer SGD is used. For layers tested with a larger number of neurons, the dropout rate was increased from 0.3 to 0.5 with the addition of batch normalisation layers to reduce overfitting while achieving higher training and validation accuracies.

When applied to either the activations of a previous layer or inputs directly, batch normalisation is a method for standardising the inputs to a network. Batch normalisation provides some regularisation, reduces generalisation error, and increases the robustness of the neural network. Batch normalisation

speeds up training. Due to a stronger tendency for the model to overfit and a longer training period as the number of neurons increases, batch normalisation was utilised only when evaluating models consisting of neurons in powers of 2. Batch normalisation can shorten the training period, reducing overfitting while boosting overall computing efficiency [25].

After testing different combinations of layers and neurons per layer, it is easily observed from the tables that when the optimizer SGD is used in the model, the training and validation accuracies are much lower than when the optimizer adam is used. Most graphs for when the optimizer SGD is used have underfitting validation curves when the number of input neurons was set to 256, 512, and 1024. Hence, only the optimizer adam will continue to be used in the model as overall, the training and validation curves and accuracies are better fittings and higher regardless of the number of hidden layers of neurons. Below are the training and validation curves of the 4 highest training and validation accuracies when the optimizer adam is used.

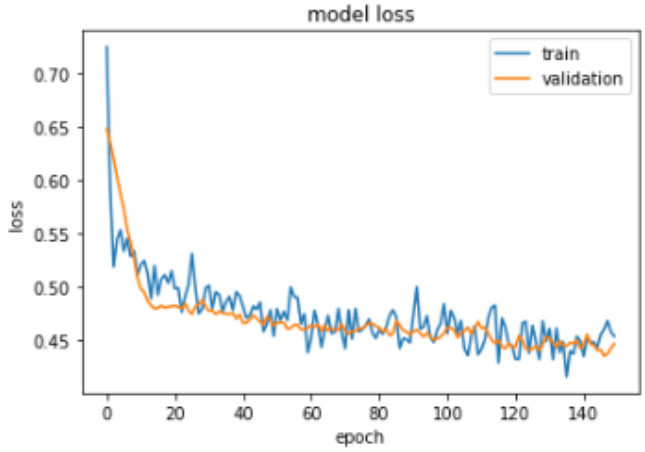


Figure 23: Neural Architecture of 256-256-8 neurons

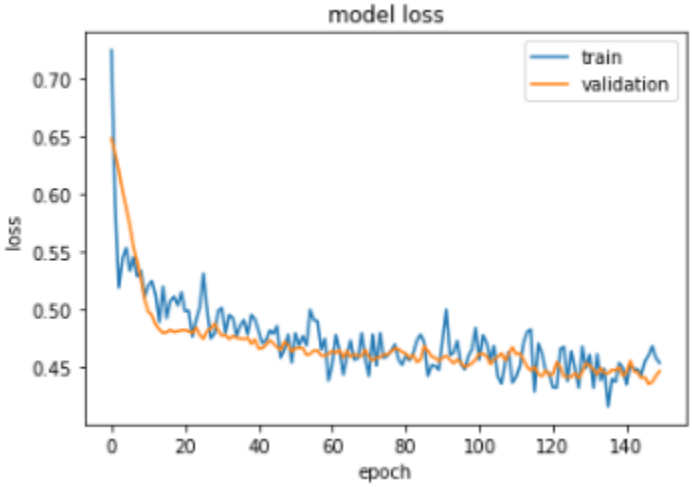


Figure 24: Neural Architecture of 256-164-8 neurons

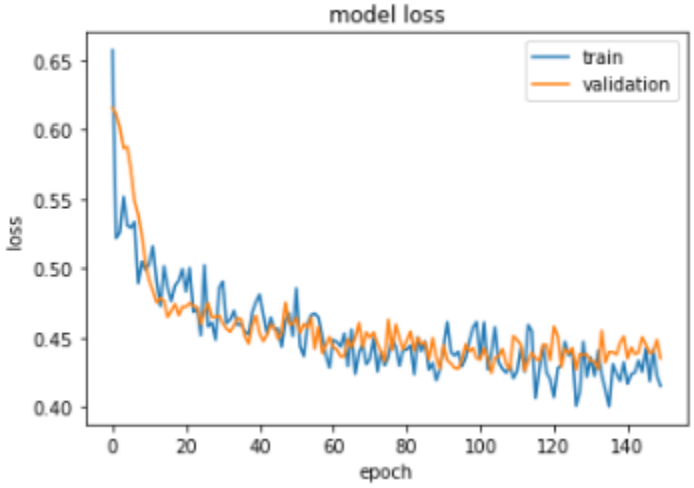


Figure 25: Neural Architecture of 512-512-8 neurons

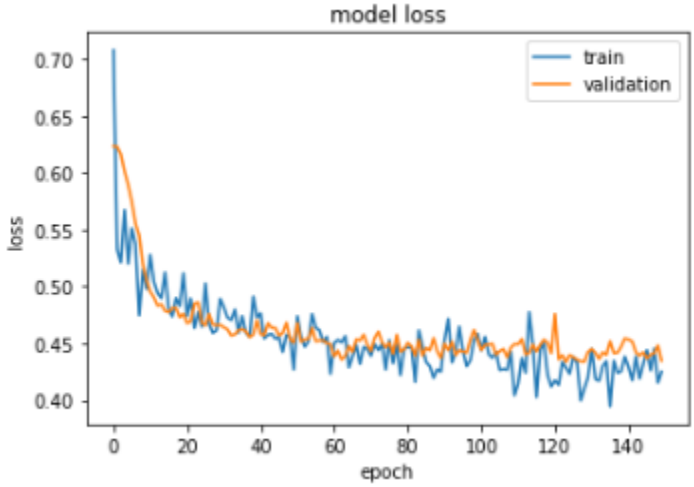


Figure 26: Neural Architecture of 512-334-8 neurons

When the number of neurons in the input layer is 512, these combinations have the most balance between loss differences and accuracies. With reference to the graphs, however, both neural architectures which have good fitting training and validation curves start to overfit after 120 epochs. When the number of neurons in the input layer is 256, these combinations also have a good balance between loss differences and accuracy. It is also observed that when the last hidden layer has 8 neurons, the model gives the best training and validation accuracies.

3.2.5 Different number of epochs, learning rates, and momentums

Next, the number of epochs has to be tuned. In order to do so, the number of epochs, learning rate and momentum of the optimizer Adam are grid searched to find the best combination of these hyperparameters for the 4 neural architectures chosen above. Appendix A, table 12 shows the results of the grid search values while table 13 shows the improved accuracies and losses after the grid-search hyperparameters have been implemented. The graphs below show the training and validation curves of the 4 highest training and validation accuracies after tuning the hyperparameters:

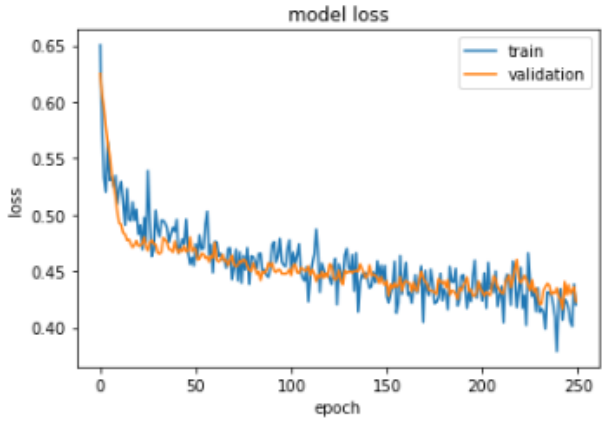


Figure 27: Neural Architecture of 256-256-8 neurons

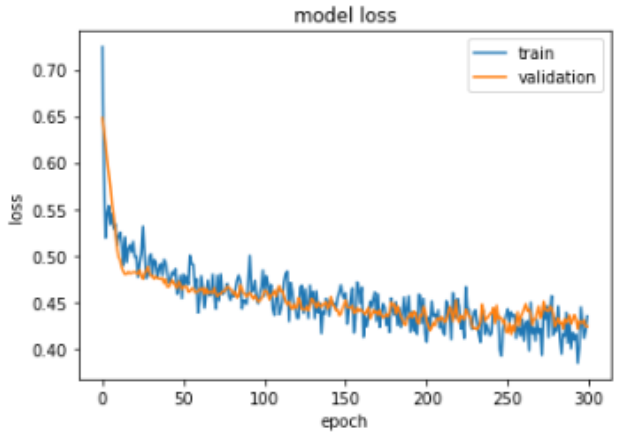


Figure 28: Neural Architecture of 256-164-8 neurons

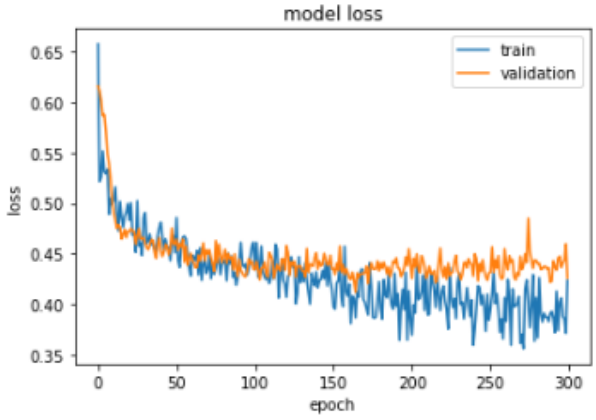


Figure 29: Neural Architecture of 512-512-8 neurons

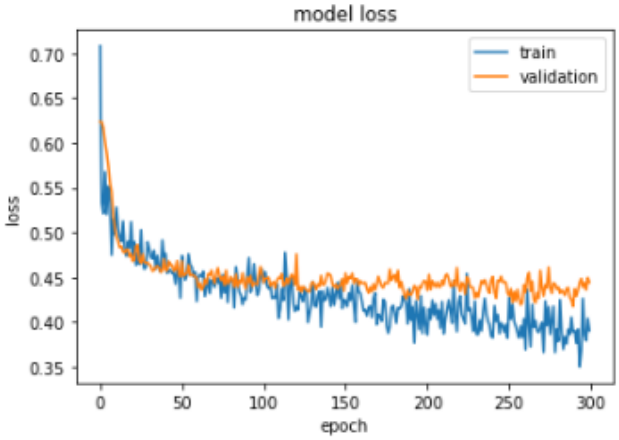


Figure 30: Neural Architecture of 512-334-8 neurons

The optimal number of neurons are 256 in the input layer, 164 in the first hidden layer, 8 neurons in the second hidden layer and one neuron in the output layer. This is because this combination of neurons achieves the highest training and validation accuracy of 88.91% and 83.64% respectively while presenting good-fitting training and validation curves as observed in the graphs above, with reference to Appendix A.

3.2.6 Different Batch Sizes

Lastly, the batch sizes will be tuned to achieve the optimal results for the final model to be selected for predictions.

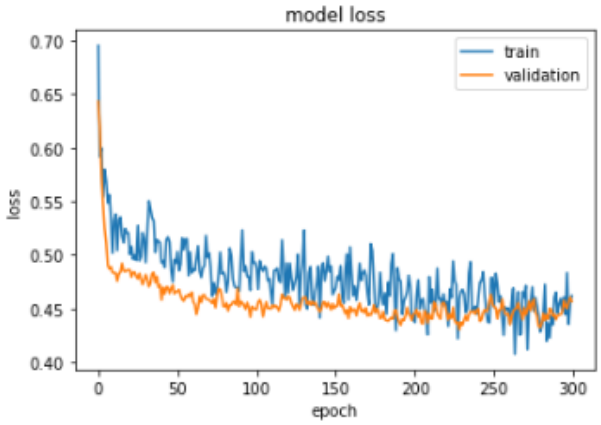


Figure 31: Batch Size of 8

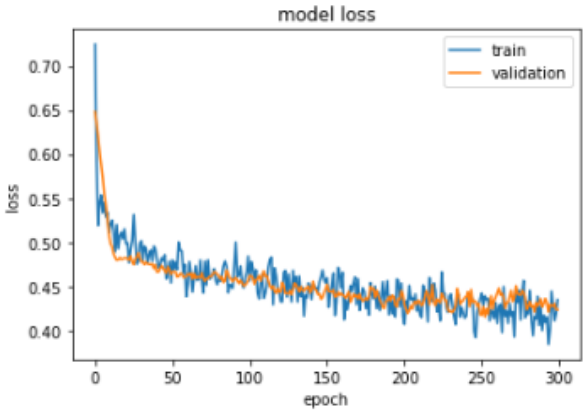


Figure 32: Batch Size of 16

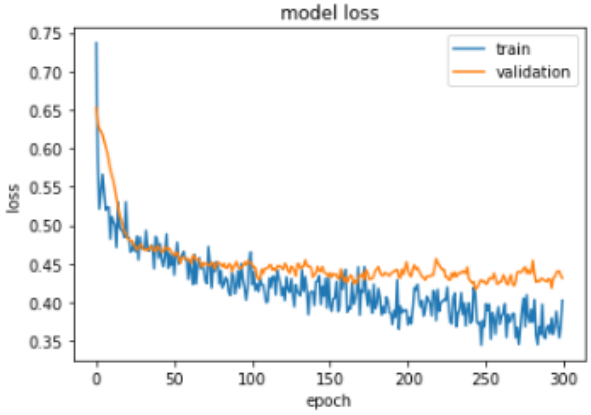


Figure 33: Batch Size of 32

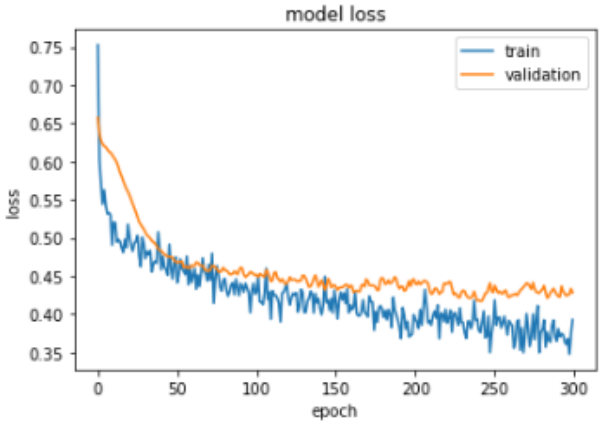


Figure 34: Batch Size of 64

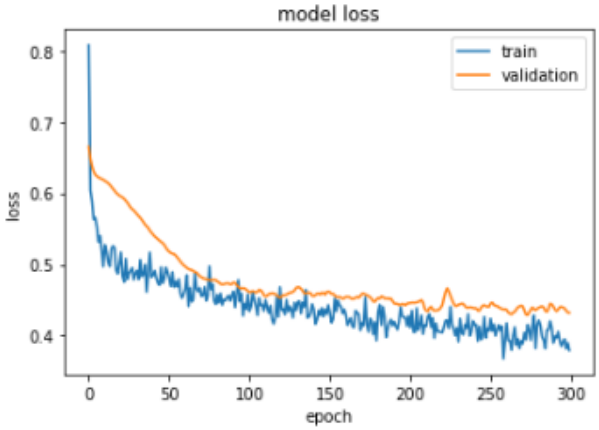


Figure 35: Batch Size of 128

The batch size selected is 16, which gives the best fitting training and validation curves balanced with high training and validation accuracies of 88.91% and 83.64% respectively, with reference to Appendix A, table 14. This model will be the final model selected to be used for predictions.

3.3 Support Vector Machines (SVMs)

3.3.1 Flow of Support Vector Machines (SVMs)

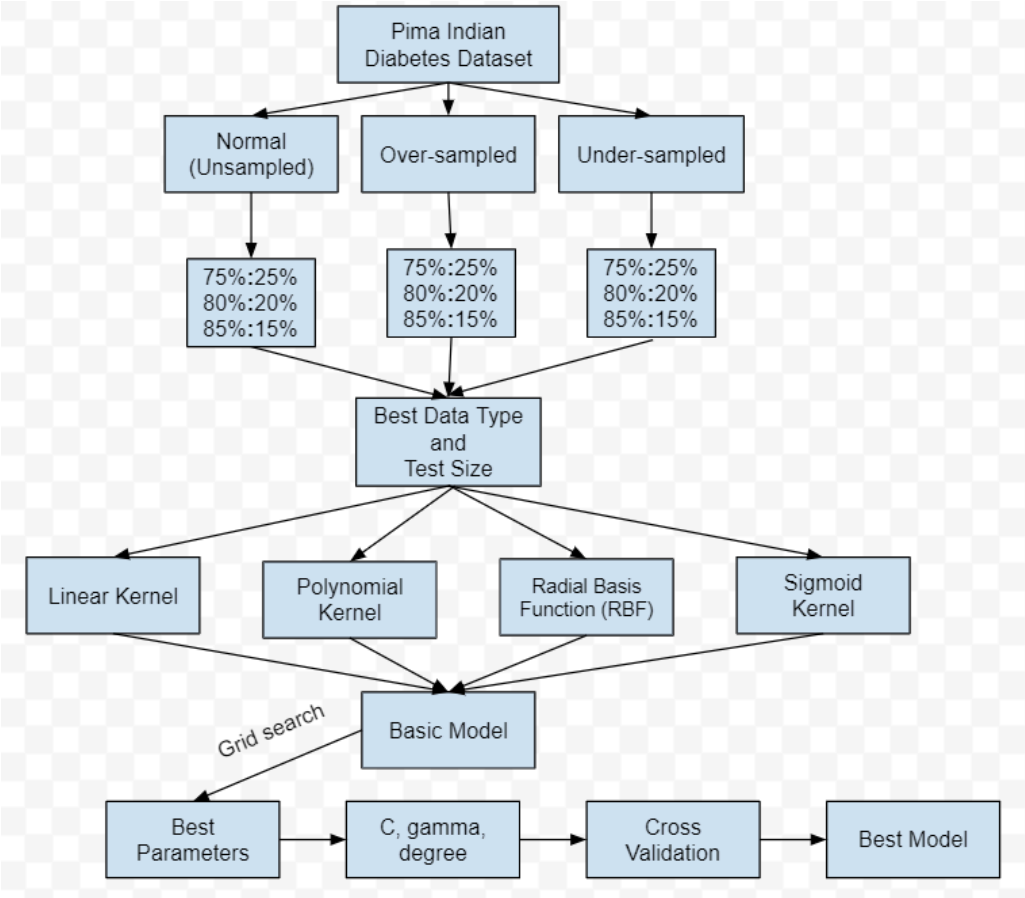


Figure 36: Flow of Support Vector Machine

3.3.2 Different Testing Size and Sampling Types

The train-test split is a method for assessing a machine learning algorithm. The train and test, which is most commonly expressed as a percentage between 0 and 1, will affect the accuracy of the algorithm's performance. Three different train-test ratios were fed into Grid Search to test the most appropriate ratio and corresponding data type (unbalanced, over-sampled, or under-sampled), 75%:25%, 80%:20%, and 85%:15%.

Appendix B, tables 1, 2, and 3 show the prediction accuracy of the unbalanced, oversampled, and undersampled data with the different train-test ratios. Even though the prediction accuracy of the unsampled data is highest among the 3, the unsampled data is not suitable for classification as the Outcome is imbalanced. Hence, comparing the prediction accuracy values of oversampled and undersampled, the accuracy of oversampled data with the train-test ratio of 75%:25%.

3.3.3 Types of Kernels

The focused kernels will be Linear, Polynomial, Radial Basis Function, and Sigmoid. Using 75%:25% as the train-test ratio, the respective parameters for respective kernels will be put into a basic model to check its accuracy. Next, to further evaluate the kernel's performance, a confusion matrix is frequently used to describe the performance of a classification model on a set of test data with known true values. Since the data is symmetrical, a good indicator of accuracy is the ratio of accurately predicted observations to all

observations. Regarding positive observations, precision is the proportion of accurately predicted observations to all predicted positive observations. For recall, any score above 0.5 is considered good for the model's recall because it measures the proportion of correctly predicted positive observations to all observations in the actual class. F1-score is a weighted average of precision and recall, taking both false positive and false negative into account [26]. Next, to find out the best parameters, hyperparameter tuning is a procedure that helps increase the accuracy of the model as it evaluates the parameters needed for the method to perform and gives the best accuracy. Lastly, Cross-validation is a method for evaluating the generalizability of statistical analysis to a different data set. It is a method for assessing machine learning models that comprises training various models on different subsets of the input data and then comparing the results. For SVM, the “cross_val_score” is used. It used KFold/StratifiedKFold, which provides train/test indices and splits the dataset in K consecutive folds, each fold served as validation. A number of performance measures will be printed out. The test score, train score, accuracy (a ratio of correctly predicted observation to the total observations), F1 score, recall, and precision.

3.3.3.1 Linear Kernels

The performance analysis for Linear Support Vector Machine (SVM) is as shown. The linear kernel is used when data can be linearly separable (using a single line) with the C parameter affecting the SVM optimization as it is the amount of misclassification for each training example. A basic model with a basic C value will be used to check the accuracy. The values can be obtained from Appendix B, Tables 4 and 5.

The confusion matrix can be obtained from Appendix B, Table 6. The measurement measures can be obtained from Appendix B, Table 7.

The main parameter is C. hence, a series of numbers will be passed into the model for gird searching. The best score, the values, and the corresponding best C parameters can be found in Appendix B, Table 8.

Next, the best C parameter will be put back into the basic model, increasing the accuracy from before, as seen in Appendix B, Table 9. The confusion matrix, precision, recall, f1-score, and support will be plotted again, values as shown in Appendix B, Tables 10 and 11.

The values of the respective measures found from cross-validation are recorded in Appendix B, Table 12.

3.3.3.2 Polynomial Kernel

Polynomial kernel is a non-linear SVM kernel. Unlike the linear kernels, Non-linear kernels add the decision boundary with the z-axis acting as the third axis for 3-dimensional plotting. Compared to the linear kernel, the non-linear contains more than 1 parameter. For instance, the Polynomial kernel has parameters such as gamma, C value, and degree (flexibility of decision boundary). A basic model with a basic C value and degree will be used to check the accuracy. The values can be obtained from Appendix B, Table 13.

The confusion matrix can be obtained from Appendix B, Table 14. The measurement measures can be obtained from Appendix B, Table 15.

For the Polynomial kernel, the main parameters are C and degree. Hence, a series of numbers will be passed into the model for grid searching. The best score, the values, and the corresponding best C and degree parameters can be found in Appendix B, Table 16.

Next, the best C and degree parameters will be put back into the basic model, increasing the accuracy from before, as seen in Appendix B, Table 17. The confusion matrix, precision, recall, f1-score, and support will be plotted again, values as shown in Appendix B, Tables 18 and 19.

The values of the respective measures can be found from cross-validation are recorded in Appendix B, Table 20.

3.3.3.3 Gaussian Radial Basis Function (RBF) Kernel

Gaussian RBF SVM's main parameter is Gamma and C. For the Gamma value used in the RBF kernel, it is directly proportional to the distance between the two vectors, with low values denoting "far" and large values denoting "near." Models with lower gamma values are equally as accurate as those with higher gamma values. Gamma in the basic RBF model to check the accuracy of the data set is set to "scale" so that Gamma will use this equation: 1 / (n_features * X.var()) to calculate the suitable value. The accuracy value can be obtained in Appendix B, Table 21.

The confusion matrix can be obtained from Appendix B, Table 22. The measurement measures can be obtained from Appendix B, Table 23.

For the RBF kernel, the main parameters are C and gamma. Hence, a series of numbers will be passed into the model for gird searching. The best score, the values, and the corresponding best C and gamma parameters can be found in Appendix B, Table 24.

Next, the best C and gamma parameters will be put back into the basic model, increasing the accuracy from before, as seen in Appendix B, Table 25. The confusion matrix, precision, recall, f1-score, and support will be plotted again, values as shown in Appendix B, Tables 26 and 27.

The values of the respective measures found from cross-validation are recorded in Appendix B, Table 28.

3.3.3.4 Sigmoid Kernel

This function serves as an activation function for artificial neurons and is similar to a two-layer perceptron neural network model. The main parameters of the Sigmoid kernel are C value and gamma. A basic Sigmoid model is being used to find the accuracy rate of the Sigmoid kernel. The accuracy value is shown in Appendix B, Table 29.

The confusion matrix can be obtained from Appendix B, Table 30. The measurement measures can be obtained from Appendix B, Table 31.

For the Sigmoid kernel, the main parameters are C and gamma. Hence, a series of numbers will be passed into the model for gird searching. The best score, the values, and the corresponding best C and gamma parameters can be found in Appendix B, Table 32.

Next, the best C and gamma parameters will be put back into the basic model, increasing the accuracy from before, as seen in Appendix B, Table 33. The confusion matrix, precision, recall, f1-score, and support will be plotted again, values as shown in Appendix B, Tables 34 and 35.

The values of the respective measures found from cross-validation are recorded in Appendix B, Table 36.

3.3.5 Comparing all Kernels

One of the most crucial metrics for evaluating the success of the model is the Area Under The Curve Receiver Operating Characteristics (ROC) curve. The ROC curve and the AUC display the separability metric, which indicates how well the model can differentiate between classes. The classifier is likely to be able to distinguish between class values and detect more true positive and true negative signals when the AUC is between 0.5 and 1 [27].

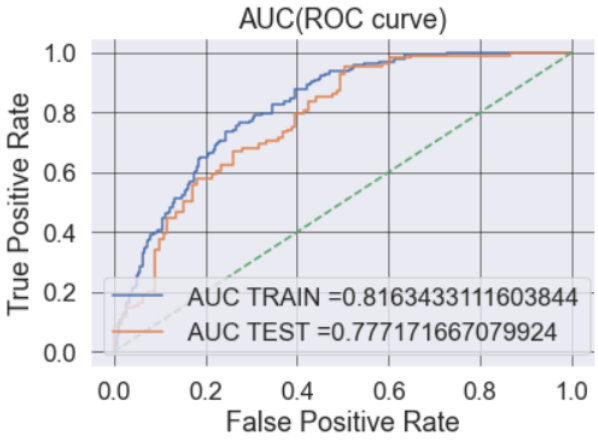


Figure 37: AUC (ROC Curve) for Linear Kernel

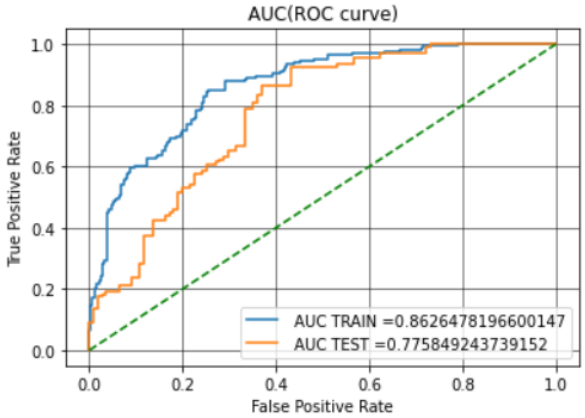


Figure 38: AUC (ROC Curve) for Polynomial Kernel

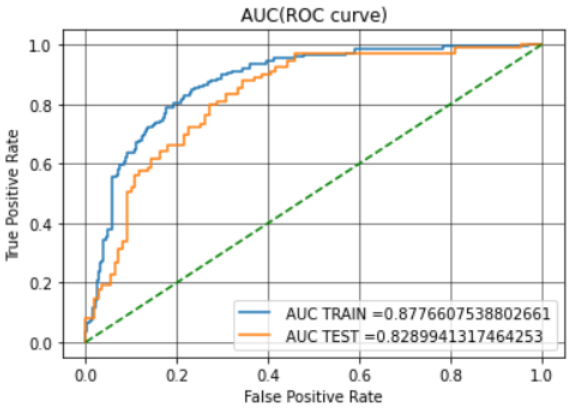


Figure 39: AUC (ROC Curve) for Radial Basis Function Kernel

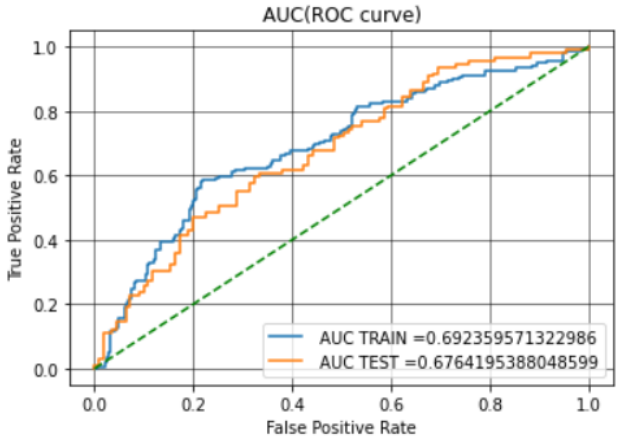


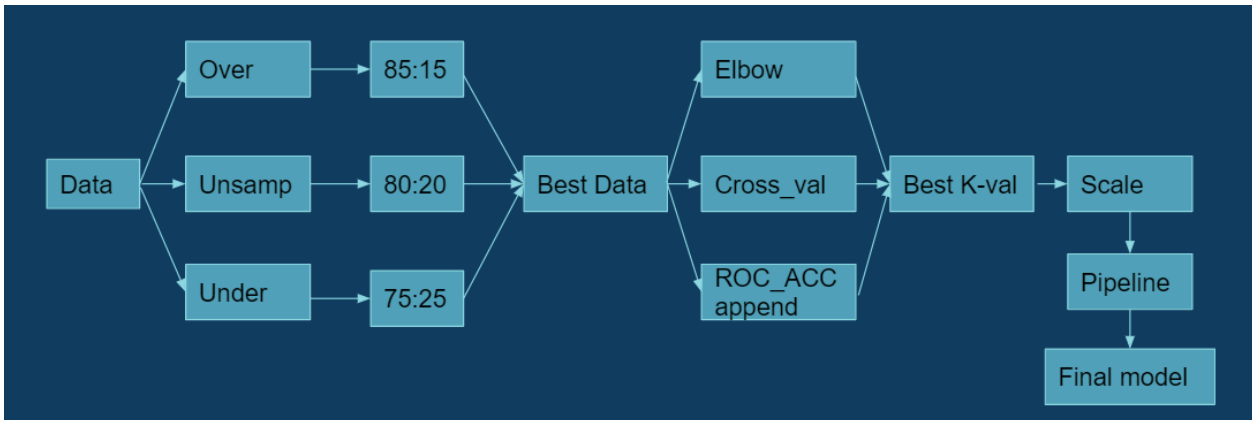
Figure 40: AUC (ROC Curve) for Sigmoid Kernel

From the comparison above, the best kernel that produces the best accuracy and has the highest Area Under the Curve value for both train and test is the RBF kernel.

3.4 KNN Model

KNN or k-nearest neighbors is a supervised learning classifier that makes classifications and predictions based on the proximity of individual data points. It is often used as a classification algorithm, working off the assumption that similar individual data points can be found close to each other.

3.4.1 Flow of KNN



3.4.2 Testing different batch sizes

F1 -score is a good metric when comparing imbalanced data to balanced data as it combines precision and recall into a single metric, making it suitable for such applications

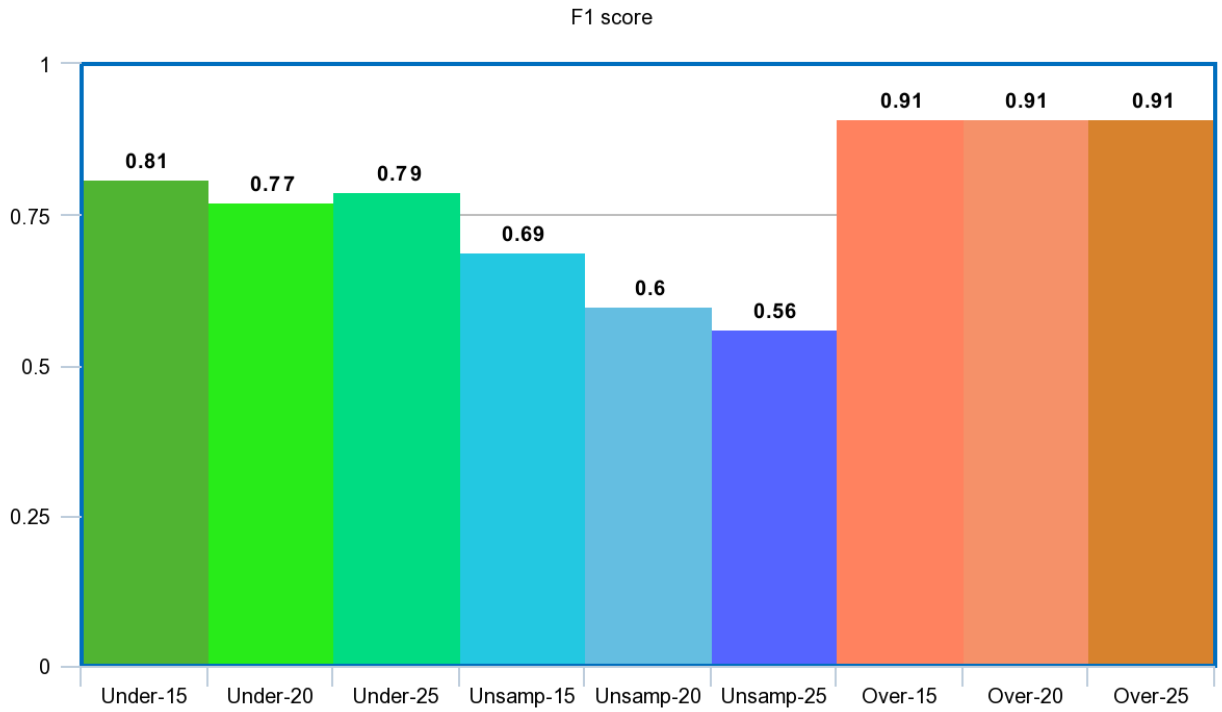


Figure 41: F1 scores of Sampling sizes

As shown above, KNN gets affected by imbalanced classes easily because an imbalanced dataset will often lead to the probability that the K-nearest neighbor of any random data point will belong to the class

with more samples. This results in more bias towards the class with more samples and makes the data more inaccurate when making predictions [28].
Each test size is then trained and tested with the KNN model, using the three different methods to identify the best K-value. As most classifiers''s best results have stemmed from Oversampled, test size 25%, the KNN model has also used it as as it's f1-score, prediction accuracy and ROC_AUC score is similar to the the 15% and 20% test size.

3.4.1 KNN Elbow Method

This method plots the test error rates against their corresponding k-value to determine the best K-value. This method is easy to implement however has many downsides, one of which is that the user must make their own inference on which k value is the best and where to cut it off to prevent large variance in the data, making it very acceptable to human error. The algorithm is also the most 'naive' as it assigns all k values to k clusters even if that is not the right k-value for the dataset [29].

For this particular dataset, as we are doing binary classification, the value k=1, which would usually lead to overfitting, is a better choice than the other k values, as the classes are balanced and easily separable given its characteristic of 0 and 1 [30].

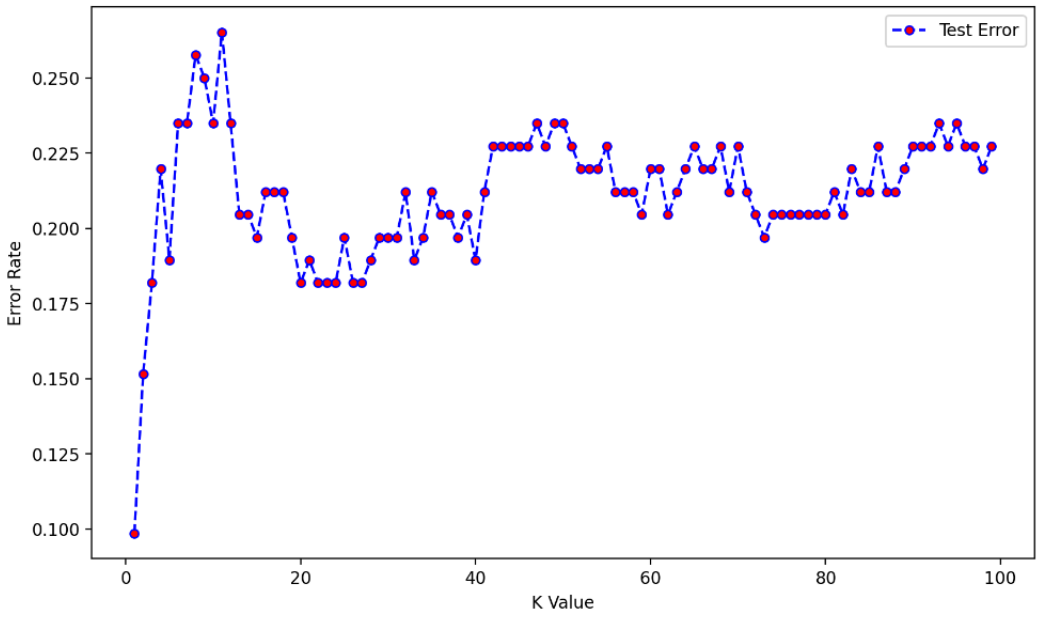


Figure 42: Elbow method graph: Error rate vs K-value

From this graph, we can see that the error rate at k=1 is the lowest, and values after 1 lead to wide variations which do not decrease below the initial error rate, therefore it is best to pick k=1 for this method [31].

3.4.2 KNN Cross Validation with GridSearch

Cross-validation is a reliable method as it lets the model test on multiple splits and allows it to make better predictions on unseen data. We also hypertuned our parameters using GridSearchCV to find the optimal

value for K-nearest neighbors based on the parameter that we had specified (1:100). Gridsearch will then go through every k value to identify which K-value will give the best results. This result is then cross-validated to increase the reliability. In addition we ran it through a pipeline that assembles the steps that can be cross-validated together while setting different parameters to get the best accuracy [32].

Training data	Test data

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 4
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 5

Figure 43: Cross-validation illustration

```
In [44]: full_cv_classifier.best_estimator_.get_params()

Out[44]: {'memory': None,
          'steps': [('scaler', StandardScaler()),
           ('knn', KNeighborsClassifier(n_neighbors=1))],
          'verbose': False,
          'scaler': StandardScaler(),
          'knn': KNeighborsClassifier(n_neighbors=1),
          'scaler__copy': True,
          'scaler__with_mean': True,
          'scaler__with_std': True,
          'knn__algorithm': 'auto',
          'knn__leaf_size': 30,
          'knn__metric': 'minkowski',
          'knn__metric_params': None,
          'knn__n_jobs': None,
          'knn__n_neighbors': 1,
          'knn__p': 2,
          'knn__weights': 'uniform'}
```

This presents another optimal k value, 1

Figure 44: Results Obtained from Cross-validation

To read the result from gridsearching, we look at the value obtained from ‘knn__n neighbours’, in this case it would be 1.

3.4.3 KNN ROC_ACC append method

This method is based on calculating the best AUC-ROC (Area under the curve - Receiving Operating Characteristic) score and appending it to an empty list with the corresponding k-value, in theory, this should yield the best K-value based off AUC-ROC score. It measures performance for classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree of separability. The score shows how well the model can identify the difference between the classes. The higher the score the better it is [27].

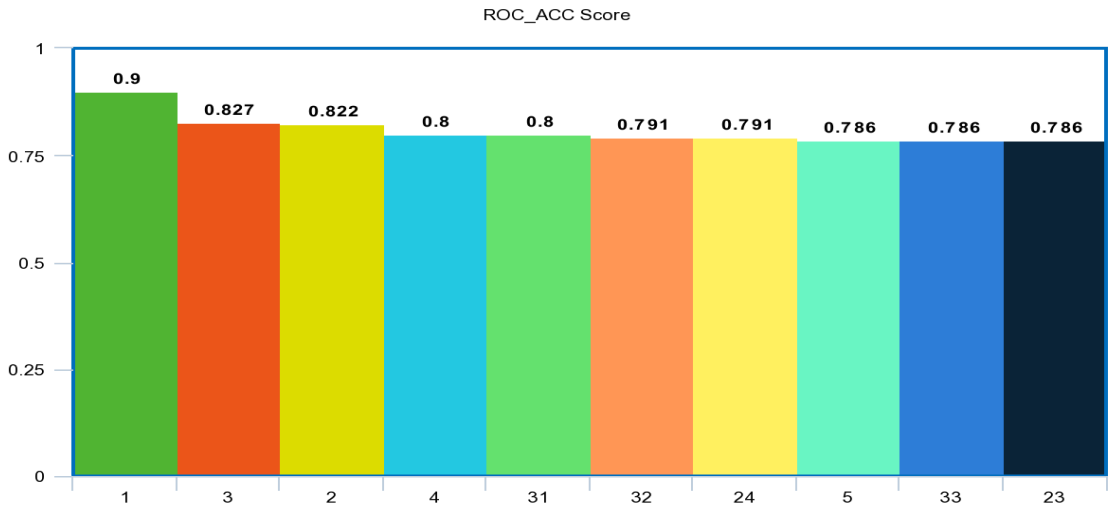


Figure 45: Histogram of ROC Curve

Table 7: ROC Accuracy of Different K-Values

K Value	ROC Accuracy
1	0.9008
3	0.8277
2	0.8223
4	0.8086
31	0.8005
32	0.8004
24	0.7913
5	0.7911
33	0.7869
23	0.7868

In this case, it confirms again that K=1 is the best value for this dataset

3.4.4 KNN Train and Test value method

This method appends the KNN score for testing and training into separate empty lists and gives the index of it (aka k value number) and gives the best score.

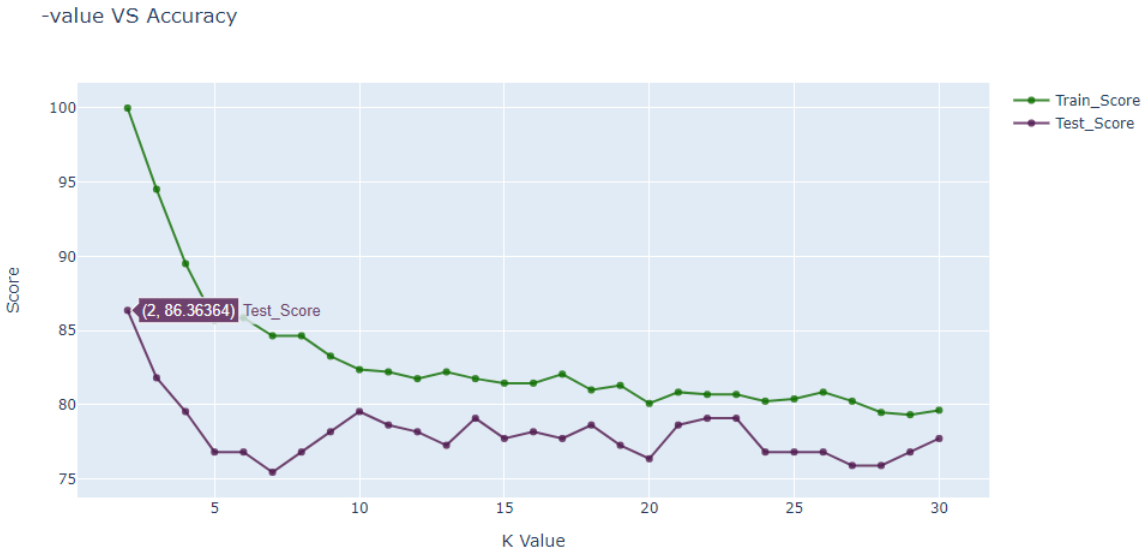


Figure 47: Line Graph Comparing K Value and Accuracy

As we can see from the graph, both train and test scores begin to decrease after k=1, thus confirming again that K=1 is best for this dataset

3.4.5 Results

As we can see from Appendix C, table 2,,3 ,4, 5. Cross validation with gridsearching is generally the best method for identifying the best K-value, identifying the best K-value for 4 out of the 9 combinations

3.5 Logistic Regression

3.5.1 Traditional method

Logistic regression is a supervised learning classification algorithm used to predict and assign observations to a discrete set of classes. The logistic regression uses a complex cost function known as the Sigmoid function. This function maps any real value into a new value between 0 and 1, and can be used to map predictions to probabilities [33].

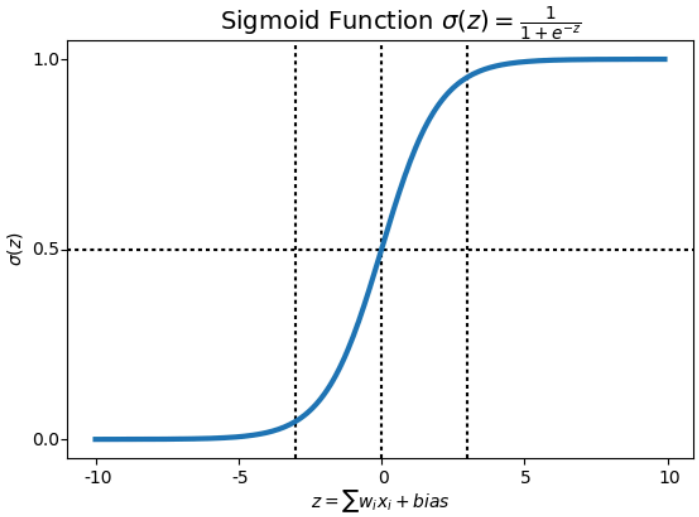


Figure 51: Sigmoid Function Curve

$$h\Theta(x) = 1/1 + e - (\beta o + \beta 1X)$$

'hΘ(x)' *is output of logistic function , where 0 ≤ hΘ(x) ≥ 1*

*'**β1**' is the slope*
*'**βo**' is the y-intercept*
***'X'** is the independent variable*

*(βo + β1*x) - derived from equation of a line Y(predicted) = (βo + β1*x) + Error value*

Figure 52: Linear Regression Equation [34]

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Figure 53: Getting Logistic Regression

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$ **; 0 for y= 0, and infinity for y=1**

Figure 54: Getting Logistic Regression

But we need range between -[infinity] to +[infinity], then take the logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Figure 55: Logistic Regression Equation

The above equation is the final equation for Logistic Regression.

3.5.2 Iteration Method

Iterative stepwise regression adds or removes one independent variable at a time to or from the multiple linear regression equation. It involves selecting independent variables to be used in a final model. Each iteration adds or removes potential explanatory variables in succession and tests for statistical significance [35].
However, this model has a downside in that some real explanatory variables that have causal effects on the dependent variable may not be statistically significant, while nuisance variables may be coincidentally significant. As a result, the model may fit the data well in-sample but poorly out-of-sample [36].

3.5.3 GridSearch Method

There are 2 different methods of doing GridSearch for logistic regression, one is through manual setting of parameters, and the other is by using a prebuilt model LogisticRegressinCV. Logspace refers to the manual setting of dimensions such as the start, end, and a number of samples to generate [37]. The log space will then be fed into the GridSearch to find the best parameters based on the dimensions provided and be cross-validated to ensure reliability. Penalties (l1 and l2) may also be introduced to reduce overfitting by biasing data to certain values [38]. The other method involves using the prebuilt model to fit the data and find the best possible parameters. This method is better if the user does not know the data well. [39] For both methods, the ‘liblinear’ solver is used as it can accommodate both penalties of l1 and l2 instead of just one and allows for a more accurate model

3.5.4 Results

With reference to appendix d, table 1,2,3,4, We can see that logistic regression gridsearch gives the best results for balanced data, such as undersampled and oversampled, test size 15 oversampled performed the best but test size 25 oversampled also performed similarly, therefore it is suitable to be used as the final model to suit the other classifier as well.

3.6 Comparing Classifiers

Finally, the best machine learning model must be selected to be used in the Graphical User Interface. Below are the classification reports, confusion matrices, and ROC curves with labelled area-under-curves for comparing the classifiers. The deciding factors for the final classifier to use will be precision accuracy, the area under the ROC curve, precision, recall, and f1-score.

Table 10: Classification Report for MLP

	Precision	Recall	F1 Score	Support

0	0.91	0.75	0.82	111
1	0.78	0.93	0.85	109
Accuracy			0.84	220
Macro Average	0.85	0.84	0.84	220

Table 12: Confusion Matrix for MLP

True	**0**	83	28
	1	8	101
	Binary Label	**0**	**1**
-	**Predicted**		

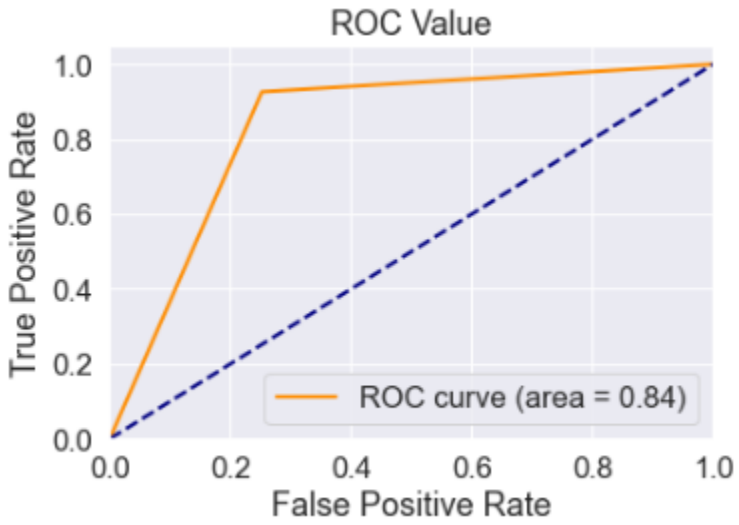


Figure 56: ROC Curve for Multi-layer Perceptron

Table 13: Classification Report for SVM

	Precision	**Recall**	**F1 Score**	**Support**
0	0.83	1.00	0.91	111
1	1.00	0.80	0.89	109
Accuracy			0.90	220
Macro Average	0.92	0.90	0.90	220

Table 14: Confusion Matrix for SVM

True	**0**	111	0
	1	22	87
	Binary Label	**0**	**1**
-	**Predicted**		

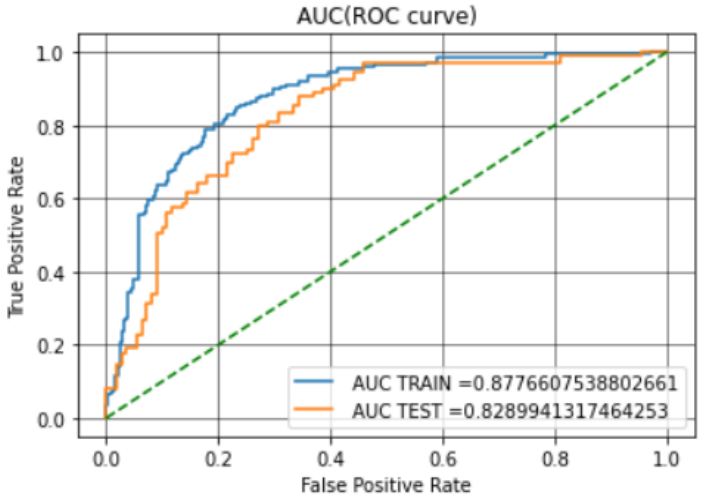


Figure 57: ROC Curve for Radian Basis Function Kernel

Table 15: Classification Report for KNN

	Precision	**Recall**	**F1 Score**	**Support**
0	0.99	0.81	0.89	111
1	0.84	0.99	0.91	109
Accuracy			0.90	220
Macro Average	0.91	0.90	0.90	220

Table 16: Confusion Matrix for KNN

True	**0**	90	21
	1	1	108
	Binary Label	**0**	**1**
-	**Predicted**		

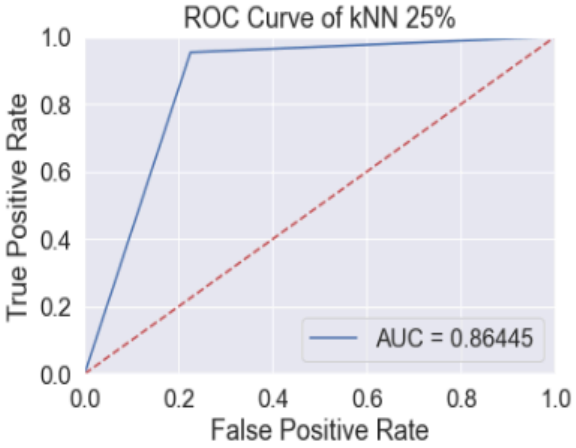


Figure 58: ROC Curve for K Nearest Neighbors

Table 18: Classification Report for Log regression

	Precision	Recall	F1 Score	Support
0	0.77	0.77	0.77	111
1	0.77	0.76	0.76	109
Accuracy			0.75	220
Macro Average	0.77	0.77	0.77	220

Table 19: Confusion Matrix for Log regression

True	**0**	86	25
	1	26	83
	Binary Label	**0**	**1**
	Predicted		

Figure 59: ROC Curve for Logistic Regression

Figures 52 to 57 are histogram plots of the different performance measures, for easier reference and comparison. Firstly, comparing the prediction accuracy of the 4 classifiers through figure 52, SVM has the highest prediction accuracy of 90%. Next is the area under the curve in figure 53. comparing the 4 classifiers, KNN has the highest area of 86.45% capability of distinguishing the two classes correctly. Next, is a comparison of precision in figure 54. precision is the ratio of correctly predicted positive observations to the total predicted observations. high precision relates to a low false positive rate. as shown in the graph, KNN has the highest precision for class 0 of 99% while SVM has the highest precision for class 1, 100%. next, recall, figure 55. the recall is the ratio of correctly predicted positive observations to all the observations in the actual class. a good recall value is higher than 50%. for the recall of class 0, SVM has the highest recall percentage of 100% and KNN has the highest recall percentage of 99% for class 1. lastly, f1-score in figure 56. it is the weighted average of precision and recall. hence, this score takes both false positives and false negatives into account. as shown in the graph, SVM has the highest F1-score for class 0 while KNN has the highest F1-score for class 1, 91%

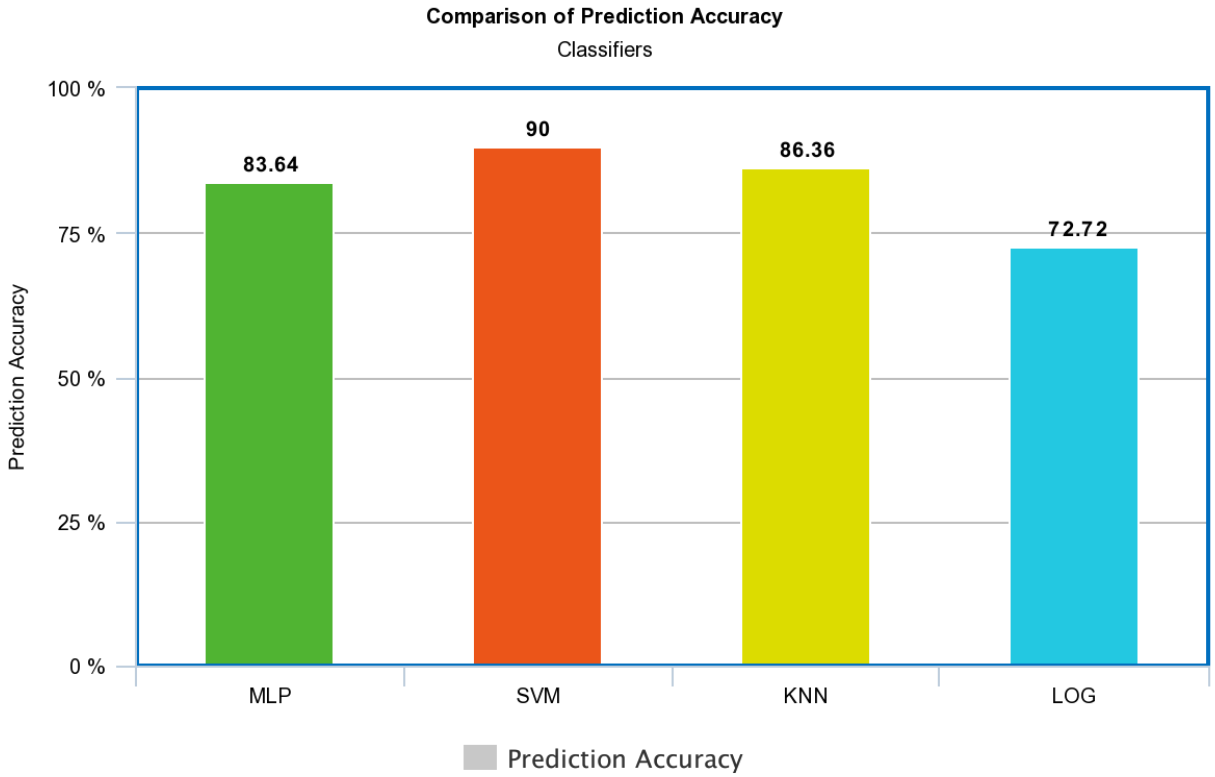


Figure 60: Histogram of Prediction Accuracy

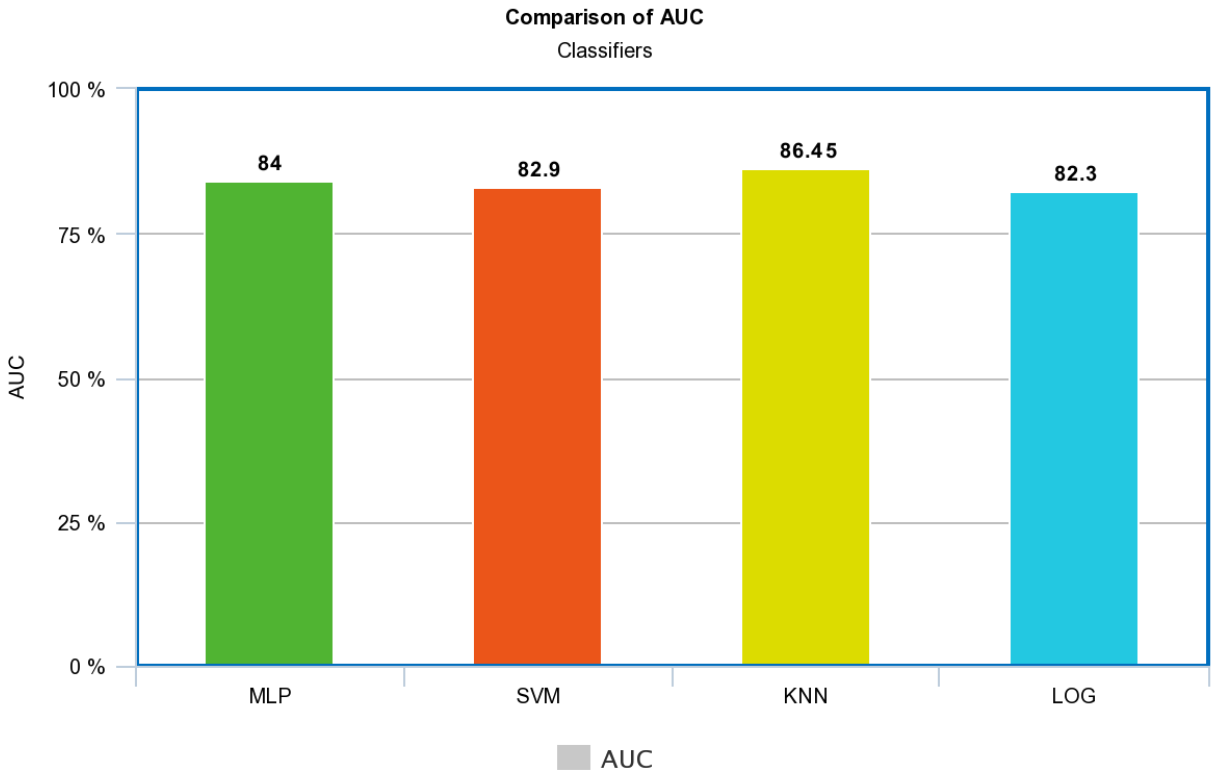


Figure 61: Histogram of Area Under Curve (AUC)

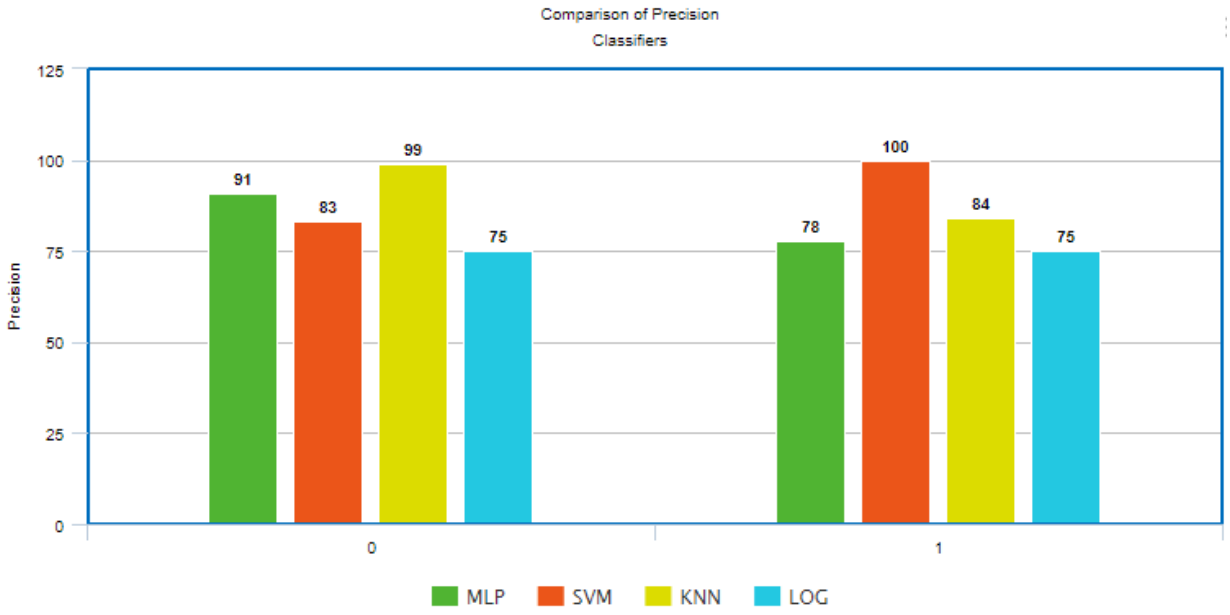


Figure 62: Histogram of Precision

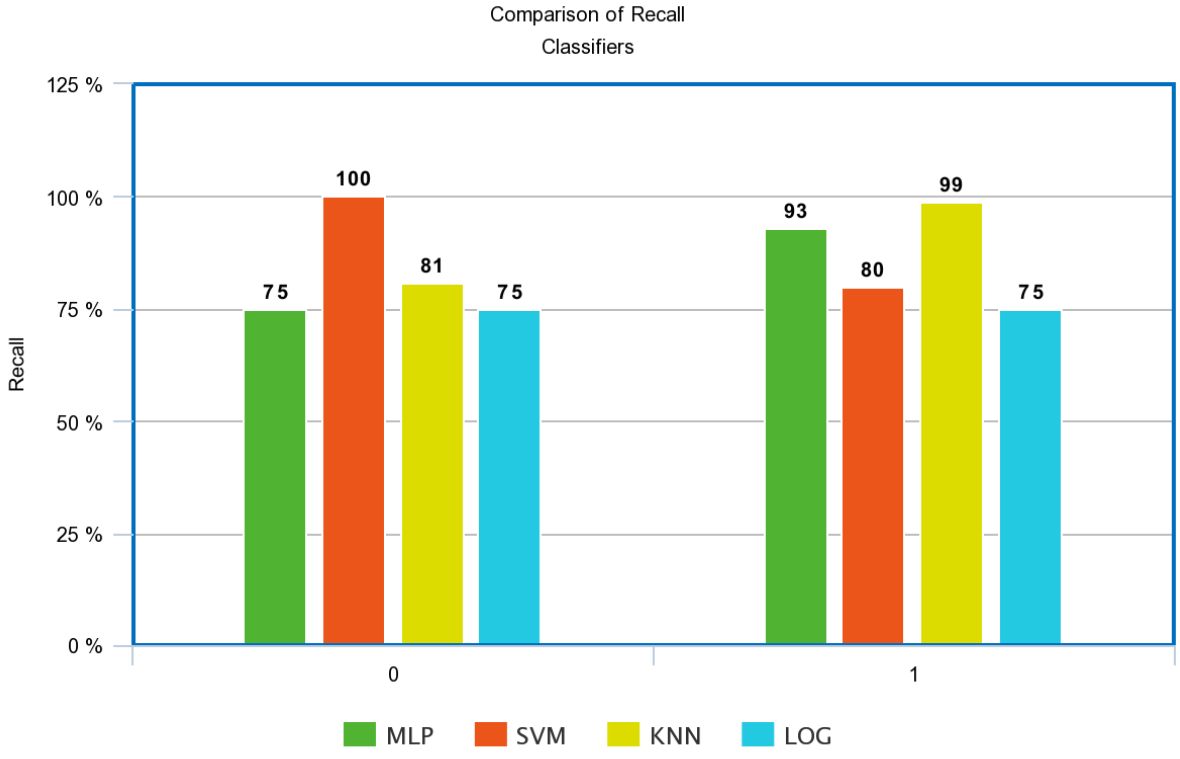


Figure 63: Histogram of Recall

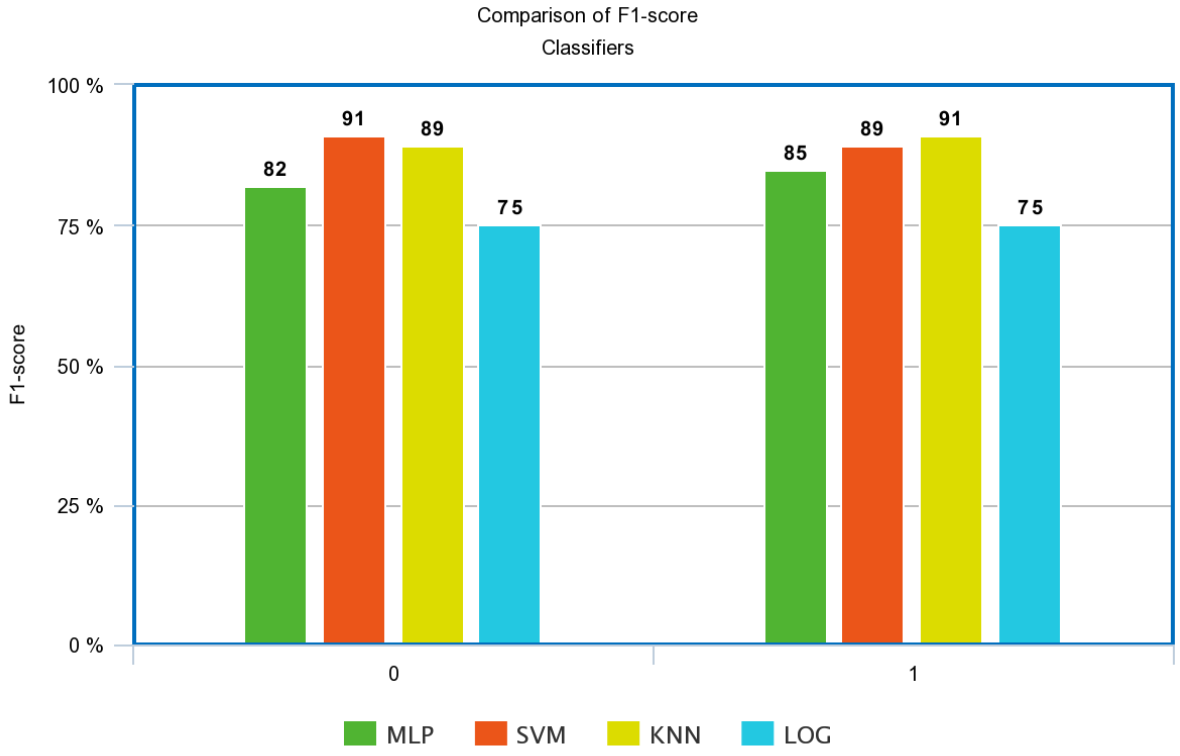


Figure 64: Histogram of F1-score

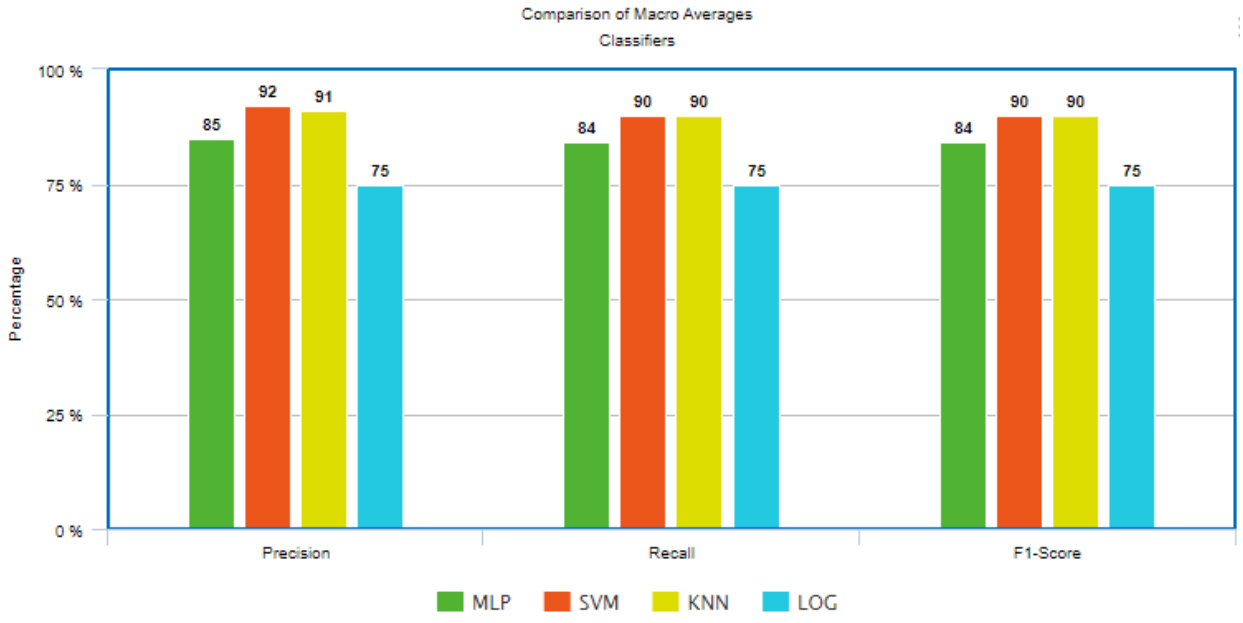


Figure 65: Histogram of Macro-Averages

Since both KNN and SVM appeared the highest number of times out of the 4 classifiers, we plotted the macro averages of the precision, recall, and f1-score to confirm our comparison. as seen from the graph, SVM and KNN has the same for recall and f1-score, leaving precision to compare the two classifiers.

SVMhas the highest precision of 92% while KNN is behind by 1%. hence, the SVM classifier will be used for this project.

Discussion

4.1 Balancing Data

There were two different sampling types that were taken into consideration, firstly, the oversampling of data. To achieve oversampling, the data of the minority class will be exactly copied at random, until it reaches the same number as the majority class. Hence, one disadvantage of oversampling will be the high chance of overfitting the model. The next option was undersampling, where data from the majority class will be removed at random, resulting in the potential of discarding any useful data. However, both methods will provide us with balanced data.

4.2 Multi-layer Perceptron

After testing different numbers of layers of the multi-layer perceptron, it can be concluded that the higher the number of layers, the longer the time to train the model but the training accuracy increases. However, overfitting tends to occur. On the contrary, fewer layers would mean less time for the model to train but may cause underfitting of the model when training accuracy decreases.

Moving on to the parameters and hyperparameters of the multi-layer perceptron, when the activation function ReLu is used, the firing of neurons either outputs 0 or 1, which can cause the dead ReLu problem, where any summation value that is negative is immediately outputted as 0. This can return many 0 values within the activation function. However, ReLu activation functions introduce sparsity of the data, meaning the matrices of the activations have many 0s, making the multi-layer perceptron computationally cheaper [40]. This provides room to add more layers and neurons which may benefit the model. Based on the Appendix A, table 13 as well as the comparison between loss graphs in 3.2.4, it is observed that when the hidden layer is approximately ⅔ of the input layer plus the output layer, the performance of the model seems to be better.

Lastly, multi-layer perceptrons have advantages and other disadvantages. Some of the latter include the issue with backpropagation, where the model tries to find a local minimum in the error function output. If the model finds the wrong minimum, the results can be very inaccurate. It is also not known to what extent each independent variable affects the outcome, so feature engineering and selection must be done, which puts this model type at an even bigger disadvantage since the proper functioning of the model depends heavily on the quality of cleaning, training, and tuning. On the flipside, multi-layer perceptrons utilise adaptive learning, which is the ability of task-learning based on the training data. Additionally, there are no biases or assumptions made by this model type in relation to the underlying probability density functions or any probabilistic data about the classes. Last but not least, multi-layer perceptrons can be used for difficult and non-linear problem solving [41].

4.2 Support Vector Machine

One benefit of SVM is that it performs well when there is a distinct boundary, or hyperplane, between the classes. It is also memory-efficient because SVM kernel models need to store a part of training data in their memory, known as the support vectors. SVM is also extremely stable so even a minor change in the

data would not have a significant impact on the hyperplane. Using the kernel method, SVM can handle non-linear data effectively as well. One of SVM's drawbacks is that choosing the appropriate SVM kernel is a time-consuming operation, increasing the likelihood that the improper kernel would be chosen. Next, SVM performs badly for imbalanced data because, when imbalanced data is used for training, hyperplanes will be biased toward the minority class. Finally, it is also challenging to adjust the SVM model's parameters [42].

4.3 K-nearest Neighbours

K-nearest neighbour's main advantage is that it is simple to implement, there are only 2 parameters that users will need to change, distance metric and K-value. It also does not have a training period as it is an instance-based learner, making it much faster than algorithms that require training such as SVM and Logistic regression. New data can also be added seamlessly without impacting the algorithm's accuracy as everything is done during prediction. There is also only one hyperparameter, K-value, making hyperparameter tuning easy and fast. Distance metrics can also be modified to suit the dataset better [43].

However its accuracy suffers when working with large datasets, as the costs of calculating the distance between the new query and each existing point is huge which degrades the performance of the algorithm [44], imbalanced datasets will also affect KNN accuracy as the probability that the K-value of any random query point will belong to the class with more example becomes higher [45]. It also becomes harder for the algorithm to calculate the distance in higher dimensions. Feature scaling is also required to make reliable predictions and this algorithm is very sensitive to noise outliers and missing data

4.4 Logistic Regression

The main advantages of Logistic regression are that it is simple to implement, fast at classifying unknown data, can use penalties to reduce overfitting and can give additional information about the predictor such as its director of the association. It also works particularly well with simple datasets and data that are linearly separable.
However, the disadvantage is that if the number of observations in the dataset is less than the number of features, overfitting may occur. Logistic regression also assumes linearity between dependent and independent variables, which more often than not is not the case. It also cannot solve nonlinear problems due to its linear decision surface. And the model is very simple, thus making more complex algorithms outperform it when it comes to more complex data and prediction accuracy [45].

4.5 Grid Search vs Random Search

Grid Search and Randomised Search are the two most popular methods for hyper-parameter optimization of any model. In both cases, the aim is to test a set of parameters whose range has been specified by the users and find the best combination of parameters.

GridSearch is very easy to implement but goes through every possible combination of parameters before finding the best, this uses up a lot of computation time which makes it inefficient but reliable. Due to how it works, it cannot be used for a large number of hyperparameters, for example, finding 20 different parameter values for 5 parameters amounts to 100000 combinations to be tested, and that does not even account for cross-validation.

RandomSearch, on the other hand, selects random combinations of hyperparameters to train the model

and score. Thus it has a shorter computation time and is more suitable for higher dimension data. However, the most optimal hyperparameter combination may not be found due to the way it works.

Both hyper-parameter search engines can be used together to achieve the best result. Random Search can be used first to reduce parameter space, and Gridsearch can select the best combination of optimal features within this space [46].

Conclusion

5.1 Summary

To sum up, various preprocessing techniques were needed to extract useful data from the Pima Indian Diabetes dataset. These feature engineering techniques included removing outliers using the Interquartile Range method, replacing missing values in the data with the median, and choosing the best predictor variables using the Pearson's Correlation Coefficient, Univariate Statistical Test, and violin plots. Afterwhich, four machine learning models namely: Multi-layer Perceptron, Support Vector Machine, K-nearest Neighbours, and Logistic Regression were trained, evaluated, and tuned based on the cleaned and processed data. After obtaining and comparing the best performing models for each of the four machine learning models, the Support Vector Machine model was discovered to be the best machine learning model in terms of its overall prediction accuracy and performance. This model was integrated with the graphical user interface to make predictions.

5.2 Potential Direction / Further Development

Instead of using only one method to balance the data, combining over and under-sampling will be a better method of balancing the data. The risk of overfitting and the likelihood of losing useful data will both be decreased by the proposed combined oversampling and undersampling strategy.

To improve the reliability of the Graphical User Interface, multiple well-trained models can be implemented. Even though the accuracy of the chosen SVM model is high, it will still be possible for the model to predict results wrongly. Hence, introducing multiple well-trained and tuned machine learning models will increase the prediction reliability of the interface as there will be predictions from multiple classifiers displayed once the user keys in the data.

Grid Search and Randomised Search are the two most popular methods for hyper-parameter optimization of any model. In both cases, the aim is to test a set of parameters whose range has been specified by the users and find the best combination of parameters. GridSearch is very easy to implement but goes through every possible combination of parameters before finding the best, this uses up a lot of computation time and power which makes it inefficient but reliable. RandomSearch on the other hand selects random combinations of hyperparameters to train the model and score. Thus it has a shorter computation time and is more suitable for higher dimension data, however, the most optimal hyperparameter combination may not be found. A combination of Grid Search and Randomised Search could be used to obtain a wider variety of the best hyperparameters, which also provides more insights into the effectiveness of each search type individually.

Appendices

Appendix A: Testing parameters/hyperparameters in Multi-Layer Perceptron

Table 1: Oversampled Data

Train/Test ratio	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference
75%:25%	0.4406	79.18	0.5033	75.00	4.18	-0.0627
80%:20%	0.4336	78.92	0.5314	71.59	7.33	-0.0978
85%:15%	0.4397	78.42	0.5114	74.24	4.18	-0.0717

Table 2: Unsampled Data

Train/Test ratio	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference
75%:25%	0.4301	78.50	0.4519	78.12	0.38	-0.0218
80%:20%	0.4261	78.67	0.4639	77.34	1.33	-0.0378
85%:15%	0.4272	79.01	0.4557	79.17	-0.16	-0.0285

Table 3: Undersampled Data

Train/Test ratio	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference
75%:25%	0.4703	77.67	0.4369	77.00	0.67	0.0334
80%:20%	0.4610	77.50	0.4475	75.00	2.50	0.0135
85%:15%	0.4619	77.65	0.4283	75.00	2.65	0.0336

Table 4: Testing Number of hidden layers with optimizer Adam (Before Dropout Regularisation)

Neural Structure ID	Number of Hidden Layers	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference	Average Loss Difference
A1	5	0.0334	98.78	1.2072	85.00	13.78	-1.1738	-0.4608
A2	4	0.1646	95.59	0.8153	80.00	15.59	-0.6507	

A3	3	0.1556	93.92	0.6667	80.45	13.47	-0.5111	
A4	2	0.3231	85.41	0.5807	78.64	6.77	-0.2576	
A5	1	0.3775	84.04	0.4929	73.64	10.4	-0.1154	
A6	0	0.4528	78.27	0.5094	71.82	6.45	-0.0566	

Table 5: Testing Number of hidden layers with optimizer SGD (Before Dropout Regularisation)

Neural Structure ID	Number of Hidden Layers	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference	Average Loss Difference
S1	5	0.2328	91.95	0.5415	81.81	10.14	-0.3086	-0.1221
S2	4	0.3547	86.32	0.4951	77.73	8.59	-0.1404	
S3	3	0.3707	84.65	0.4608	78.18	6.47	-0.0901	
S4	2	0.4092	81.76	0.4868	76.36	5.4	-0.0776	
S5	1	0.4316	81.16	0.4895	74.55	6.61	-0.0579	
S6	0	0.4541	77.81	0.5121	73.18	4.63	-0.0580	

Table 6: Grid Search Dropout Regularisation and Kernel Constraints for optimizer Adam

Neural Structure ID	Number of Hidden Layers	Best Dropout Rate	Best Weight Constraints	Best Mean Cross-Validation Score
A1	5	0.4	1.0	0.7963
A2	4	0.5	1.0	0.7979
A3	3	0.3	1.0	0.8025
A4	2	0.3	1.0	0.7964
A5	1	0.1	2.0	0.7797
A6	0	0.1	2.0	0.7751

Table 7: Grid Search Dropout Regularisation and Kernel Constraints for optimizer SGD

Neural Structure ID	Number of Hidden	Best Dropout Rate	Best Weight Constraints	Best Mean Cross-Validation Score

	Layers			
S1	5	0.0	1.0	0.8025
S2	4	0.3	1.0	0.7979
S3	3	0.1	1.0	0.7979
S4	2	0.2	1.0	0.7888
S5	1	0.1	2.0	0.7645
S6	0	0.1	1.0	0.7599

Table 8: Adding Dropout and Kernel Constraints for optimizer Adam

Neural Structure ID	Number of Hidden Layers	Number of Dropout Layer	Number of Weight Constraints	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Loss Difference	Average Loss Difference
A1	5	6	2	0.3452	86.47	0.4465	78.64	-0.1013	-0.0657
A2	4	4	3	0.4318	85.11	0.5484	80.00	-0.1166	
A3	3	4	1	0.3693	83.89	0.4380	79.09	-0.0687	
A4	2	3	3	0.4143	81.31	0.4565	78.18	-0.0422	
A5	1	2	0	0.4240	80.85	0.4608	76.36	-0.0368	
A6	0	1	0	0.4602	76.60	0.4888	72.27	-0.0286	

Table 9: Adding Dropout and Kernel Constraints for optimizer SGD

Neural Structure ID	Number of Hidden Layers	Number of Dropout Layer	Number of Weight Constraints	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Loss Difference	Average Loss Difference
S1	5	0	0	0.2520	89.82	0.4835	80.45	-0.2315	-0.0676
S2	4	3	1	0.4584	81.00	0.4951	76.82	-0.0367	
S3	3	4	1	0.4080	82.07	0.4455	78.18	-0.0375	
S4	2	2	0	0.4308	80.85	0.4642	77.73	-0.0334	
S5	1	2	0	0.4496	79.48	0.4863	75.45	-0.0367	
S6	0	1	0	0.4649	75.38	0.4949	70.91	-0.0300	

Table 10: Testing number of neurons for optimizer Adam

Neural Structure ID	Number of Hidden Layers	Number of neurons in order (Left to Right)	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference	Average Loss Difference
A4	2	16, 8, 4	0.4367	81.61	0.4919	75.45	6.16	-0.0552	-0.0887
A4	2	32, 16, 4	0.4110	82.98	0.4691	77.73	5.25	-0.0581	
A4	2	24, 12, 4	0.4290	81.00	0.4946	73.64	7.36	-0.0656	
A4	2	24, 10, 6	0.4238	80.55	0.4828	72.27	8.28	-0.0590	
A4	2	256, 256, 8	0.3449	85.87	0.4376	79.09	6.48	-0.0927	
A4	2	256, 256, 16	0.3449	85.56	0.4401	77.27	8.29	-0.0952	
A4	2	256, 164, 8	0.3626	84.35	0.4469	77.73	6.62	-0.0843	
A4	2	512, 512, 8	0.3219	87.23	0.4352	80.00	7.23	-0.1133	
A4	2	512, 512, 16	0.3351	86.32	0.4384	79.09	7.23	-0.1033	
A4	2	512, 334, 8	0.3218	87.23	0.4348	78.64	8.59	-0.113	
A4	2	1024, 1024, 8	0.3358	86.63	0.4488	77.73	8.9	-0.113	
A4	2	1024, 676, 8	0.3161	87.23	0.4290	80.91	6.32	-0.1129	

Table 11: Testing number of neurons for optimizer SGD

Neural Structure ID	Number of Hidden Layers	Number of neurons in order (Left to Right)	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)	Accuracy Difference (%)	Loss Difference	Average Loss Difference
S4	2	16, 8, 4	0.4702	77.96	0.4979	74.55	3.41	-0.0277	-0.0527
S4	2	32, 16, 4	0.4554	79.64	0.4878	74.09	5.55	-0.0324	
S4	2	24, 12, 4	0.4644	78.42	0.5178	72.73	5.69	-0.0534	
S4	2	24, 10, 6	0.4602	80.24	0.4961	74.55	5.69	-0.0359	
S4	2	256, 256, 8	0.4157	82.67	0.4741	76.36	6.31	-0.0584	
S4	2	256, 256, 16	0.4269	81.91	0.4758	76.36	5.55	-0.0489	
S4	2	256, 164, 8	0.4210	82.52	0.4726	76.36	6.16	-0.0516	
S4	2	512, 512, 8	0.3980	82.98	0.4666	77.27	5.71	-0.0686	
S4	2	512, 512, 16	0.4085	82.22	0.4669	76.82	5.4	-0.0584	
S4	2	512, 334, 8	0.4284	81.31	0.4810	77.73	3.58	-0.053	
S4	2	1024, 1024, 8	0.4062	81.61	0.4753	73.64	7.97	-0.0691	
S4	2	1024, 676, 8	0.3941	83.28	0.4696	75.91	7.37	-0.0755	

Table 12: Grid Search for Epochs, Learning Rate and Momentum

Number of neurons in order (Left to Right)	Number of Epochs	Best Learning Rate	Best Momentum Value	Best Mean Cross-Validation Score
256, 256, 8	250	0.001	0.0	0.8025

256, 164, 8	300	0.001	0.0	0.7994
512, 512, 8	300	0.001	0.0	0.8009
512, 334, 8	300	0.001	0.0	0.8009

Table 13: After Implementing Grid Search hyperparameters

Number of neurons in order (Left to Right)	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)
256, 256, 8	0.3167	87.99	0.4230	80.00
256, 164, 8	0.3198	88.91	0.4234	83.64
512, 512, 8	0.2754	89.97	0.4262	80.45
512, 334, 8	0.2819	90.27	0.4444	81.36

Table 14: Testing Different Batch Sizes

Batch Size	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)
8	0.3412	86.93	0.4567	79.55
16	0.3198	88.91	0.4234	83.64
32	0.2681	90.27	0.4316	82.27
64	0.2814	89.36	0.4283	80.00
128	0.3239	86.63	0.4322	79.55

Appendix B: Testing parameters/hyperparameters in Support Vector Machine

Table 1: Accuracy for Normal (Unbalanced Data)

Train/Test ratio	**Prediction Accuracy**
75%:25%	0.7562

80%:20%	0.7968
85%:15%	0.7916

Table 2: Accuracy for Over-sample

Train/Test ratio	Prediction Accuracy
75%:25%	0.7545
80%:20%	0.7159
85%:15%	0.7121

Table 3: Accuracy for Under-sample

Train/Test ratio	Prediction Accuracy
75%:25%	0.73
80%:20%	0.725
85%:15%	0.7166

Table 4: Accuracy with Corresponding Parameters Before Grid Search (Linear SVM)

Accuracy	C Value
69.0909%	0.5

Table 5: C Parameters and Corresponding Accuracy

Accuracy of Linear SVM	Corresponding C parameter
0.6864	0.10
0.6818	0.20
0.6955	0.30
0.6909	0.40
0.6909	0.50
0.6909	0.60
0.6909	0.70
0.6909	0.80

0.6909	0.90

Table 6: Confusion Matrix Before Grid Search (Linear SVM)

True Negative	80
True Positive	72
False Negative	37
False Positive	31

Table 7: Precision, Recall and Support Before Grid Search (Linear SVM)

	Precision	**Recall**	**F1-score**	**Support**
0	0.68	0.72	0.70	111
1	0.70	0.66	0.68	109
Accuracy			0.69	220

Table 8: Grid Search (Linear SVM)

C Parameter	**Best Score**
0.001	0.7447
0.01	
0.1	
1	
5	
10	

Table 9: Accuracy with Corresponding Parameters After Grid Search (Linear SVM)

Accuracy	**C Value**
70.9090%	0.001

Table 10: Confusion Matrix After Grid Search (Linear SVM)

True Negative	84
True Positive	72

False Negative	37
False Positive	27

Table 11: Precision, Recall, F1-score and Support After Grid Search (Linear SVM)

	Precision	Recall	F1-score	Support
0	0.69	0.76	0.72	111
1	0.73	0.66	0.69	109
Accuracy			0.71	220

Table 12: Cross Validation Results (Linear SVM)

Test Score	Train Score	Accuracy	F1 Score	Recall	Precision
0.7272	0.7193	0.73 (+/- 0.05)	0.7039	0.6477	0.7702
0.7045	0.7378		0.6941	0.6704	0.7195
0.75	0.7279		0.7317	0.6818	0.7894
0.7142	0.7254		0.6875	0.6321	0.7534
0.7714	0.7283		0.7560	0.7045	0.8157

Table 13: Accuracy with Corresponding Parameters Before Grid Search (Polynomial SVM)

Accuracy	Gamma	C Value	Degree
64.5454%	“scale”	1.0	2.0

Table 14: Confusion Matrix Before Grid Search (Polynomial SVM)

True Negative	89
True Positive	53
False Negative	56
False Positive	22

Table 15: Precision, Recall and Support Before Grid Search (Polynomial SVM)

	Precision	Recall	F1-score	Support
0	0.61	0.80	0.70	111
1	0.71	0.49	0.58	109

Accuracy		0.61	0.65	220

Table 16: Grid Search (Polynomial SVM)

C Parameter	Degree	Best Score
0.001	**1**	0.7439
0.01	2	
0.1	3	
1	4	
5	5	
10	6	

Table 17: Accuracy with Corresponding Parameters After Grid Search (Polynomial SVM)

Accuracy	C Value	Degree
70.4545%	0.01	1

Table 18: Confusion Matrix After Grid Search (Polynomial SVM)

True Negative	82
True Positive	29
False Negative	36
False Positive	73

Table 19: Precision, Recall and Support After Grid Search (Polynomial SVM)

	Precision	Recall	F1-score	Support
0	0.69	0.74	0.72	111
1	0.72	0.67	0.68	109
Accuracy			0.70	220

Table 20: Cross Validation Results (Polynomial SVM)

Test Score	Train Score	Accuracy	F1 Score	Recall	Precision
0.7159	0.7464	0.73 (+/- 0.04)	0.6951	0.6477	0.75
0.75	0.7834		0.7777	0.875	0.7

0.7272	0.7521		0.7073	0.6590	0.7631
0.6914	0.7325		0.6301	0.5287	0.7796
0.7485	0.7809		0.7659	0.8181	0.72

Table 21: Accuracy with Corresponding Parameters Before Grid Search (RBF SVM)

Accuracy	Gamma	C Value
75.9090%	“scale”	0.5

Table 22: Confusion Matrix Before Grid Search (RBF SVM)

True Negative	76
True Positive	91
False Negative	18
False Positive	35

Table 23: Precision, Recall and Support Before Grid Search (RBF SVM)

	Precision	Recall	F1-score	Support
0	0.81	0.68	0.74	111
1	0.72	0.83	0.77	109
Accuracy			0.76	220

Table 24: Grid Search (RBF SVM)

C Parameter	Gamma	Best Score
0.001	0.001	0.8768
0.01	0.01	
0.1	0.1	
1	1	
10	10	
100	**100**	

Table 25: Accuracy with Corresponding Parameters After Grid Search (RBF SVM)

Accuracy	Gamma	C Value

90%	100	1

Table 26: Confusion Matrix After Grid Search (RBF SVM)

True Negative	111
True Positive	87
False Negative	22
False Positive	0

Table 27: Precision, Recall and Support After Grid Search (RBF SVM)

	Precision	**Recall**	**F1-score**	**Support**
0	0.83	1	0.91	111
1	1	0.80	0.89	109
Accuracy			0.90	220

Table 28: Cross Validation Results (RBF SVM)

Test Score	**Train Score**	**Accuracy**	**F1 Score**	**Recall**	**Precision**
0.9090	1	0.94 (+/- 0.04)	0.9	0.8181	1
0.9318	1		0.9268	0.8636	1
0.9375	1		0.9333	0.875	1
0.96	1		0.9580	0.9195	1
0.9371	1		0.9333	0.875	1

Table 29: Accuracy with Corresponding Parameters Before Grid Search (Sigmoid SVM)

Accuracy	**Gamma**	**C Value**
58.6363%	0.5	1

Table 30: Confusion Matrix Before Grid Search (Sigmoid SVM)

True Negative	62
True Positive	67
False Negative	42

False Positive	49

Table 31: Precision, Recall and Support Before Grid Search (Sigmoid SVM)

	Precision	Recall	F1-score	Support
0	0.60	0.56	0.58	111
1	0.58	0.61	0.60	109
Accuracy			0.59	220

Table 32: Grid Search (Sigmoid SVM)

C Parameter	Gamma	Best Score
0.001	0	0.7326
0.0035	**0.1**	
0.0129	0.2	
0.0464	0.3	
0.1668	0.4	
0.5994	0.5	
2.1544	0.6	
7.7426	0.7	
27.8255	0.8	
100	0.9	

Table 33: Accuracy with Corresponding Parameters After Grid Search (Sigmoid SVM)

Accuracy	Gamma	C Value
66.8181%	0.1	0.046415888336127795

Table 34: Confusion Matrix After Grid Search (Sigmoid SVM)

True Negative	70
True Positive	77
False Negative	32
False Positive	41

Table 35: Precision, Recall and Support After Grid Search (Sigmoid SVM)

	Precision	Recall	F1-score	Support
0	0.69	0.63	0.66	111
1	0.65	0.71	0.68	109
Accuracy			0.67	220

Table 36: Cross Validation Results (Sigmoid SVM)

Test Score	Train Score	Accuracy	F1 Score	Recall	Precision
0.7272	0.7364	0.73 (+/- 0.05)	0.7176	0.6931	0.7439
0.7045	0.7364		0.7045	0.7045	0.7045
0.7159	0.7350		0.7093	0.6931	0.7261
0.7314	0.7240		0.7116	0.6666	0.7631
0.7828	0.7083		0.7790	0.7613	0.7976

Appendix C: Testing parameters/hyperparameters in KNN

Table 1: K values gathered, best k-val with star

Test Size	15%	20%	25%
Under-Sampled	1,4,9*	1,5,84,99*	1,3,15,21*
Unsampled	1*,17	1,6,15,22*	1,6,13,47*
Over-Sampled	1*	1*	1*

Table 2: best method of identifying K-val

Test Size	15%	20%	25%
Under-Sampled	Train and Test	ROC_append	Train and Test
Unsampled	Cross_val	ROC_append	Elbow
Over-Sampled	Cross_val	Cross_val	Cross_val

Table 3: Best K-val f1 score (positive)

Test Size	15%	20%	25%
Under-Sampled	0.82	0.77	0.8
Unsampled	0.69	0.55	0.55
Over-Sampled	0.91	0.91	0.91

Table 4: Best prediction accuracy

Test Size	15%	20%	25%
Under-Sampled	0.816	0.7625	0.76
Unsampled	0.8125	0.75	0.78
Over-Sampled	0.886	0.875	0.9

Table 5: Best ROC_AUC score

Test Size	15%	20%	25%
Under-Sampled	0.863	0.86	0.851
Unsampled	0.808	0.797	0.816
Over-Sampled	0.901	0.897	0.901

Table 6: Best model's classification report (k=1, 25% , oversampled)

	Precision	**Recall**	**F1-score**	**Support**
0	0.99	0.81	0.89	111
1	0.84	0.99	0.91	109
Accuracy			0.90	220

Appendix D: Testing parameters/hyperparameters in Logistic Regression

Table 1: Best method

Test Size	15%	20%	25%
Under-Sampled	GridSearch	GridSearch	GridSearch
Unsampled	Traditional	Traditional	Traditional

Over-Sampled	GridSearch	GridSearch	GridSearch

Table 2: Test size and sampling size best f1 score

Test Size	15%	20%	25%
Under-Sampled	0.8166	0.81	0.77
Unsampled	0.60	0.59	0.59
Over-Sampled	0.82	0.78	0.75

Table 3: Best prediction accuracy

Test Size	15%	20%	25%
Under-Sampled	0.75	0.825	0.78
Unsampled	0.80	0.79	0.79
Over-Sampled	0.825	0.7272	0.75

Table 4: ROC_AUC score

Test Size	15%	20%	25%
Under-Sampled	0.888	0.899	0.841
Unsampled	0.789	0.801	0.808
Over-Sampled	0.869	0.842	0.833

References

[1] P. Saeedi, P. Salpea, S. Karuranga, I. Petersohn, B. Malanda, E. W. Gregg, N. Unwin, S. H. Wild, and R. Williams, “Mortality attributable to diabetes in 20-79 years old adults, 2019 estimates: Results from the International Diabetes Federation Diabetes Atlas, 9 TH edition,” *Diabetes research and clinical practice*, 15-Feb-2020. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32068099/#:~:text=Diabetes%20is%20estimated%20to%20contribute,Middle%20East%20and%20North%20Africa. [Accessed: 20-Jul-2022].

[2]“Global report on diabetes,” *World Health Organization*, 21-Apr-2016. [Online]. Available: https://www.who.int/publications/i/item/9789241565257. [Accessed: 15-Jul-2022].

[3]E. F. Brutsaert, “Diabetes mellitus (DM) - hormonal and metabolic disorders,” *MSD Manual Consumer Version*, Sep-2020. [Online]. Available: https://www.msdmanuals.com/en-sg/home/hormonal-and-metabolic-disorders/diabetes-mellitus-dm-and-disorders-of-blood-sugar-metabolism/diabetes-mellitus-dm. [Accessed: 15-Jul-2022].

[4] “What is diabetes?,” *Centers for Disease Control and Prevention*, 07-Jul-2022. [Online]. Available: https://www.cdc.gov/diabetes/basics/diabetes.html. [Accessed: 26-Jul-2022].

[5]“Polycystic ovary syndrome (PCOS),” *Mayo Clinic*, 03-Oct-2020. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/pcos/symptoms-causes/syc-20353439#:~:text=Polycystic%20ovary%20syndrome%20(PCOS)%20is,fail%20to%20regularly%20release%20eggs. [Accessed: 26-Jul-2022].

[6]“Diabetes,” *Mayo Clinic*, 30-Oct-2020. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/diabetes/symptoms-causes/syc-20371444. [Accessed: 26-Jul-2022].

[7] By: IBM Cloud Education, “What is machine learning?,” *IBM*. [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning. [Accessed: 10-Jun-2022]

[8]“Diabetes,” *Mayo Clinic*, 30-Oct-2020. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/diabetes/diagnosis-treatment/drc-20371451#. [Accessed: 20-Jul-2022].

[9]“The A1C test & diabetes,” *National Institute of Diabetes and Digestive and Kidney Diseases*, Apr-2018. [Online]. Available: https://www.niddk.nih.gov/health-information/diagnostic-tests/a1c-test. [Accessed: 20-Jul-2022].

[10]“A comparative study to predict diabetes using machine learning techniques,” Aug-2021. [Online]. Available: https://www.researchgate.net/publication/357221683_A_Comparative_Study_to_Predict_Diabetes_using_Machine_Learning_Techniques. [Accessed: 15-Jul-2022].

[11]V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, “Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms - neural computing and applications,” *SpringerLink*, 24-Mar-2022. [Online]. Available: https://link.springer.com/article/10.1007/s00521-022-07049-z. [Accessed: 10-Jul-2022].

[12]Jbrownlee, “Datasets/pima-indians-diabetes.data.csv at master · jbrownlee/datasets,” *GitHub*, 11-Mar-2018. [Online]. Available: https://github.com/jbrownlee/Datasets/blob/master/pima-indians-diabetes.data.csv. [Accessed: 12-Jul-2022].

[13]“Python - GUI programming (TKINTER),” *Tutorials Point*. [Online]. Available: https://www.tutorialspoint.com/python/python_gui_programming.htm#. [Accessed: 10-Jul-2022].

[14]*Python programming tutorials*. [Online]. Available: https://pythonprogramming.net/python-pickle-module-save-objects-serialization/. [Accessed: 11-Jul-2022].

[15]A. Kumar, “Python - replace missing values with mean, median & mode,” *Data Analytics*, 03-Oct-2021. [Online]. Available: https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/#:~:text=You%20can%20use%2mean%20value,to%20replace%20the%20missing%20values. [Accessed: 15-Jul-2022].

[16]DeepAI. (2019, May 17). *Multilayer Perceptron*. DeepAI. Retrieved July 23, 2022, from https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron

[17]Praharsha, V. (2022, January 2). *Relu (rectified linear unit) activation function*. OpenGenus IQ: Computing Expertise & Legacy. Retrieved July 23, 2022, from https://iq.opengenus.org/relu-activation/

[18] The sigmoid activation function - python implementation. JournalDev. (2022, March 8). Retrieved July 23, 2022, from https://www.journaldev.com/47533/sigmoid-activation-function-python

[19] Martinek, V. (2020, June 19). *Cross-entropy for classification*. Medium. Retrieved July 23, 2022, from https://towardsdatascience.com/cross-entropy-for-classification-d98e7f974451

[20] O. Harrison, “Machine learning basics with the K-nearest neighbors algorithm,” *Medium*, 11-Sep-2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761# . [Accessed: 20-Jul-2022].

[21] “Most popular distance metrics used in KNN and when to use them,” *KDnuggets*. [Online]. Available: https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html . [Accessed: 20-Jul-2022].

[22]Hansen, C. (2019, December 19). *Nested cross-validation python code*. Machine Learning From Scratch. Retrieved July 23, 2022, from https://mlfromscratch.com/nested-cross-validation-python-code/#what-is-cross-validation

[23]: Brownlee, J. (2019, August 6). *A gentle introduction to weight constraints in deep learning*. Machine Learning Mastery. Retrieved May 30, 2022, from https://machinelearningmastery.com/introduction-to-weight-constraints-to-reduce-generalization-error-in-deep-learning/

[24]: P. Gaurang, Y. P. Kosta, A. Ganatra, and D. Panchal, “Behaviour analysis of multilayer perceptrons with multiple hidden ...,” Jan-2011. [Online]. Available: https://www.researchgate.net/profile/Y-Kosta/publication/272912659_Behaviour_Analysis_of_Multilayer_Perceptronswith_Multiple_Hidden_Neurons_and_Hidden_Layers/links/594cfd0045851543382a66d6/Behaviour-Analysis-of-Multilayer-Perceptronswith-Multiple-Hidden-Neurons-and-Hidden-Layers.pdf. [Accessed: 11-Jul-2022].

[25]: Brownlee, J. (2019, December 3). *A gentle introduction to batch normalization for Deep Neural Networks*. Machine Learning Mastery. Retrieved May 30, 2022, from https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[26] “Accuracy, precision, Recall & F1 score: Interpretation of performance measures,” *Exsilio Blog*, 09-Sep-2016. [Online]. Available:

https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/. [Accessed: 26-Jul-2022].

[27] S. Narkhede, “Understanding AUC - roc curve,” *Medium*, 27-Jun-2021. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. [Accessed: 16-Jul-2022].

[28] “Why does Knn get effected by the class imbalance?,” *Quora*. [Online]. Available: https://www.quora.com/Why-does-knn-get-effected-by-the-class-imbalance. [Accessed: 10-Jul-2022].

[29] M. Doumbia, “Elbow method in supervised learning(optimal K value),” *Medium*, 24-Aug-2019. [Online]. Available: https://medium.com/@moussadoumbia_90919/elbow-method-in-supervised-learning-optimal-k-value-99d425f229e7#:~:text=Elbow%20Method%20in%20Supervised%20Machine%20Learning(Optimal%20K%20Value),-The%20most%20important&text=Elbow%20method%20helps%20data%20scientists,this%20optimal%20value%20of%20K . [Accessed: 10-Jul-2022].

[30] E. Allibhai, “Building a K-nearest-neighbors (K-nn) model with Scikit-Learn,” *Medium*, 27-Sep-2018. [Online]. Available: https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a. [Accessed: 10-Jul-2022].

[31]**:** “Learn R, Python & Data Science Online,” *DataCamp*. [Online]. Available: https://www.datacamp.com/. [Accessed: 30-May-2022]

[32] “Sklearn.pipeline.pipeline,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline. [Accessed: 10-Jul-2022].

[33] A. Pant, “Introduction to logistic regression,” *Medium*, 22-Jan-2019. [Online]. Available: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148. [Accessed: 12-Jul-2022].

[34]“Logistic regression in machine learning - javatpoint,” *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/logistic-regression-in-machine-learning#:~:text=Logistic%20regression%20is%20one%20of,of%20a%20categorical%20dependent%20variable. [Accessed: 12-Jul-2022].

[35] A. Hayes, “Stepwise regression,” *Investopedia*, 10-Jan-2022. [Online]. Available: https://www.investopedia.com/terms/s/stepwise-regression.asp. [Accessed: 12-Jul-2022].

[36] G. Smith, “Step away from stepwise - journal of big data,” *SpringerOpen*, 15-Sep-2018. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-018-0143-6. [Accessed: 26-Jul-2022].

[37] “Numpy.logspace¶,” *numpy.logspace - NumPy v1.6 Manual (DRAFT)*. [Online]. Available: https://docs.scipy.org/doc/numpy-1.6.0/reference/generated/numpy.logspace.html#numpy.logspace. [Accessed: 20-Jul-2022].

[38] Stephanie, “Regularization: Simple definition, L1 & L2 penalties,” *Statistics How To*, 27-Oct-2016. [Online]. Available: https://www.statisticshowto.com/regularization/. [Accessed: 20-Jul-2022].

[39]“Sklearn.linear_model.LOGISTICREGRESSIONCV,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html. [Accessed: 20-Jul-2022].

[40] Hansen, C. (2020, June 8). *Activation functions explained - Gelu, Selu, Elu, relu and more*. Machine Learning From Scratch. Retrieved July 23, 2022, from https://mlfromscratch.com/activation-functions-explained/#/

[41] *Why Multilayer Perceptron - Massachusetts Institute of Technology*. (n.d.). Retrieved July 24, 2022, from https://courses.media.mit.edu/2006fall/mas622j/Projects/manu-rita-MAS_Proj/MLP.pdf

[42] BotBark, “Top 5 advantages and disadvantages of Support Vector Machine algorithm,” *Bot Bark*, 19-Dec-2019. [Online]. Available: https://botbark.com/2019/12/19/top-5-advantages-and-disadvantages-of-support-vector-machine-algorithm/. [Accessed: 30-Jul-2022].

[43] MLNerds, “How does KNN algorithm work ? what are the advantages and disadvantages of Knn ?,” *Machine Learning Interviews*, 14-Feb-2019. [Online]. Available: https://machinelearninginterview.com/topics/machine-learning/how-does-knn-algorithm-work-what-are-the-advantages-and-disadvantages-of-knn/. [Accessed: 20-Jul-2022].

[44] N. Kumar, “Advantages and disadvantages of KNN algorithm in Machine Learning,” *Advantages and Disadvantages of KNN Algorithm in Machine Learning*, 23-Feb-2019. [Online]. Available: http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-knn.html?_sm_au_=isVDMbsF7jV1PKNjBkGLcKQRFMcTt. [Accessed: 20-Jul-2022].

[45] “Advantages and disadvantages of logistic regression,” *GeeksforGeeks*, 02-Sep-2020. [Online]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/. [Accessed: 20-Jul-2022].

[46] “Grid search vs. Randomized Search,” *Welcome -*, 01-Jul-2019. [Online]. Available: https://maelfabien.github.io/machinelearning/GridRand/#randomized-search. [Accessed: 25-Jul-2022].