BRAC
UNIVERSITY
Inspiring Excellence

CSE422: Artificial Intelligence

Project Report

Diabetes Detection using Machine Learning Algorithms

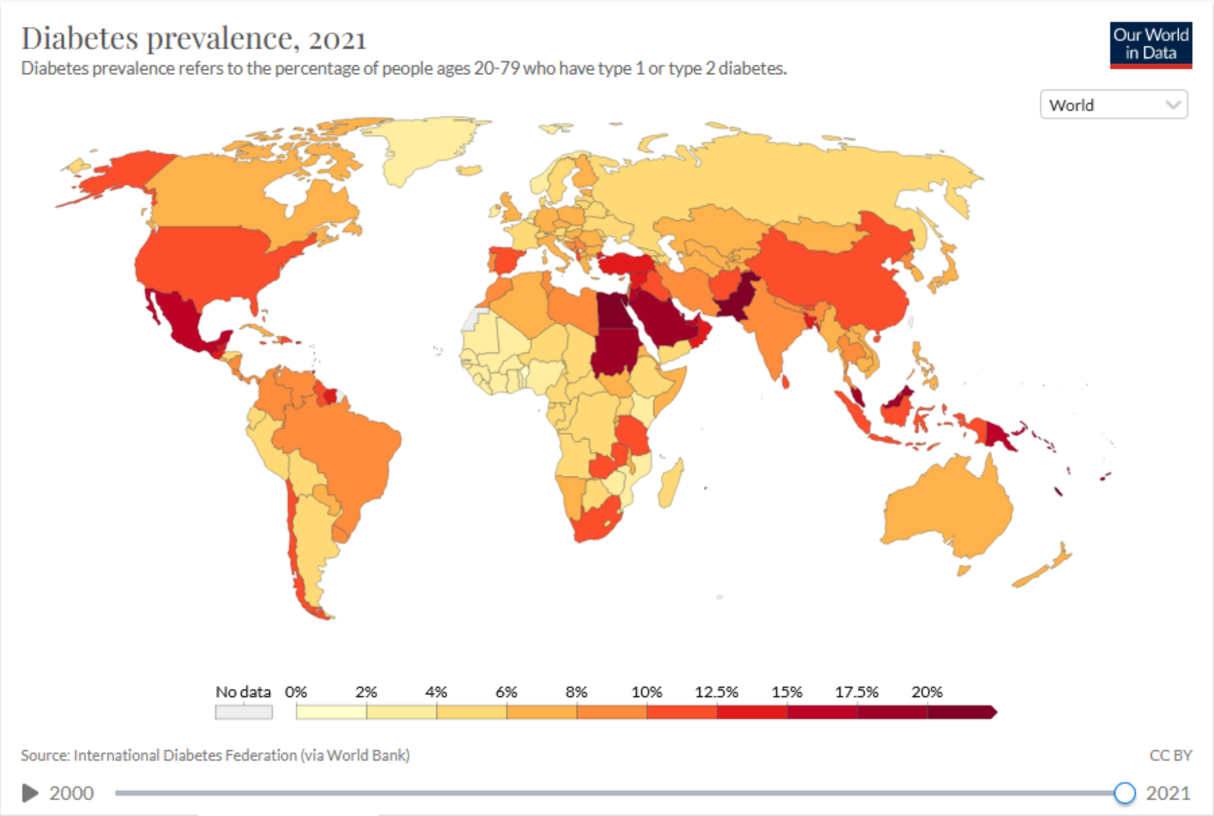
Submitted By

S.N.	Name	ID
1	Kazi Md. Al-Wakil	19301051
2	Sababa Rahman Meem	19101569
3	Arman Zaman	19201005
4	Ahmed Lateef	19241016

Section: 8 | **Group:** 1

Table of contents

Introduction



Diabetes prevalence, 2021

Diabetes prevalence refers to the percentage of people ages 20-79 who have type 1 or type 2 diabetes.

Source: International Diabetes Federation (via World Bank)

Diabetes is a kind of disease which we can call "Silent Killer". It is a chronic and metabolic disease characterized by elevated levels of blood glucose (or blood sugar), which leads over time to serious damage to the heart, blood vessels, eyes, kidneys and nerves. Type-2 diabetes is the most common type of diabetes. This type is widely seen among adults, when their body becomes resistant to insulin or can not produce enough insulin. Although recently the number of Type-1 diabetes has increased dramatically. There is a globally agreed target to halt the rise in diabetes and obesity by 2025.

According to the World Health Organization, there are 422 million people worldwide who have diabetes and 1.5 million deaths are directly attributed to diabetes each year. Both the number of cases and the prevalence of diabetes have been steadily increasing over the past few decades.

There is no hard and fast way to tell how to prevent diabetes. After analyzing so many cases, medical researchers are still ambiguous about the source of this disease. They are trying to halt the disease as fast as they can but it seems like a lot of features are directly related to this disease and we can not overcome all the factors overnight.

There is so much data available for us to research and analyze the common pattern by which we might overcome this disease. But as human beings we are bound to have limited amounts of knowledge and only to compute a finite amount of equations. But for a machine it is quite easy to calculate complex functions. If we can code a machine to do the computational part, we can gather so much knowledge about diabetes. Then we can avoid the dominating feature in order to stay safe and healthy.

Therefore, here we have a dataset where we have different features related to diabetes. We need to train our machine in such a way so that it can predict any given data related to diabetes. Given necessary feature information, the machine will predict whether a patient is diabetic or not.

Methodology

Dataset Description

The National Institute of Diabetes and Digestive and Kidney Diseases is the original source of this dataset. Based on specific diagnostic metrics present in the dataset, the dataset's goal is to diagnostically forecast whether a patient has diabetes or not. These instances were chosen from a bigger database under a number of restrictions. Particularly, all patients at this facility are Pima Indian women who are at least 21 years old.

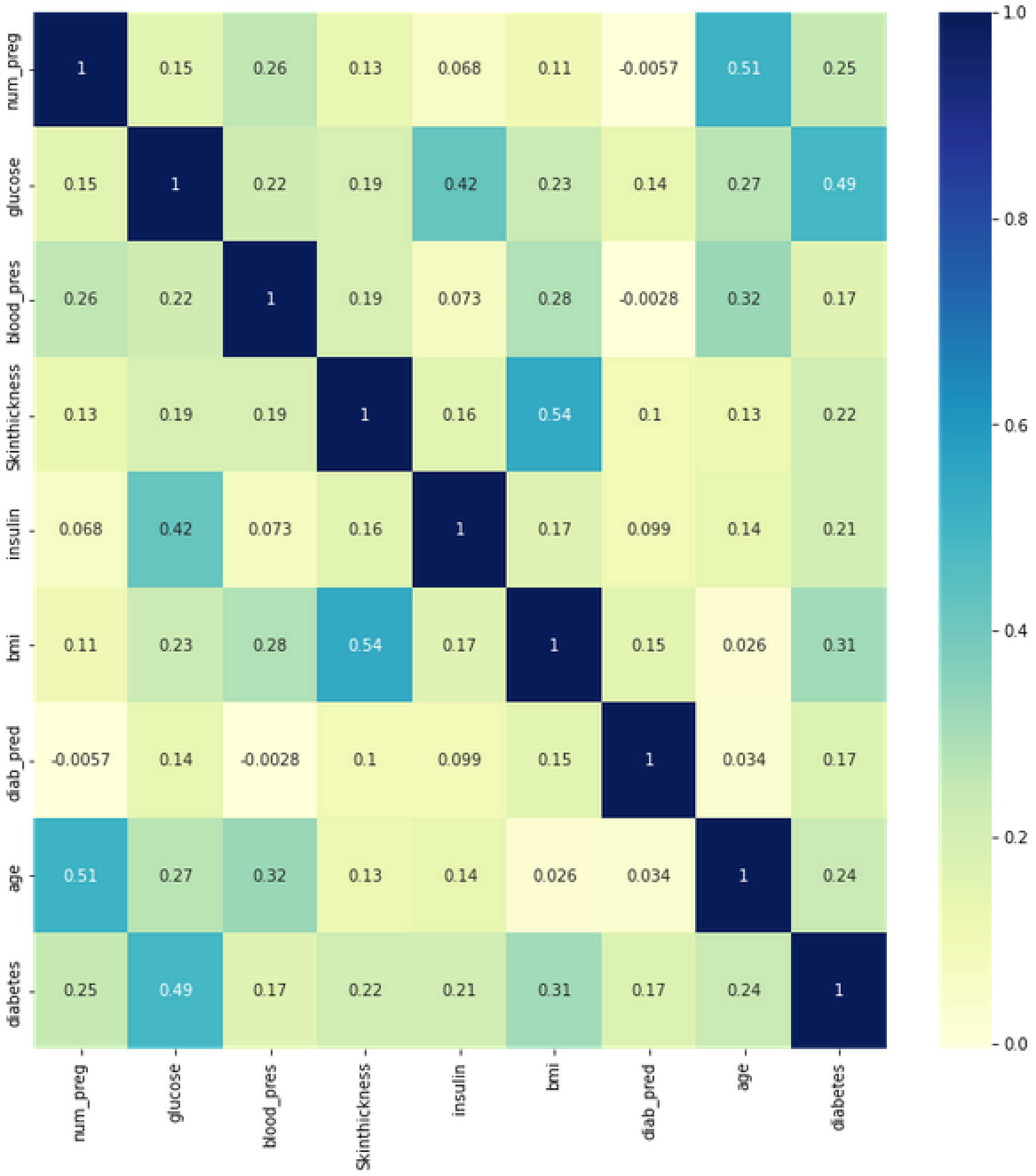
The datasets consist of one target variable or label, Outcome, and a number of medical predictor variables.The patient's BMI, insulin level, age, number of previous pregnancies, and other factors are predictor variables or features.

The dataset has 768 rows and 9 columns

Column’s Description

1. Pregnancies: Number of times pregnant
2. Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. BloodPressure: Diastolic blood pressure (mm Hg)
4. SkinThickness: Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml)
6. BMI: Body mass index (weight in kg/(height in m)^2)
7. DiabetesPedigreeFunction: Indicates the function which scores likelihood of diabetes based on family history
8. Age: Patient’s age (years)
9. Outcome: Target feature

We can visualize the whole dataset and their relation with each other through the Correlation Matrix given below:



Pre-Processing Techniques applied

Pre-processing in Machine Learning refers to cleaning and organizing the raw data at one's disposal in order to make it suitable for building and training machine learning models. This can be accomplished by the following steps:

- acquiring the dataset(first step)
- importing crucial libraries i.e. NumPy, Pandas, Matplotlib etc.

- Importing the dataset
- Identifying and handling the missing values
- Encoding the categorical data
- Splitting the dataset(into ratios i.e. training data and testing data such as 70:30 or 80:20)
- Feature Scaling(end of preprocessing, method to standardize independent variables of a dataset within a specific range.)

Firstly, we acquired the dataset from Kaggle, a platform consisting of a myriad of datasets for machine learning enthusiasts to utilize and it is also a community for data scientists. Then, we imported the necessary libraries needed such as NumPy, Pandas, Matplotlib(for generating the accuracy and loss curves and other graphs), and seaborn(to visualize random distribution/statistical data).

After importing the dataset we tried to get an overview of the dataset by seeing the row and column numbers. There are 768 rows and 9 columns in our dataset and all the columns except for the target column are of numerical values. Only the target column contains categorical values (Yes, No). After visualizing the dataset, we got an idea of which features are important for our target (Outcome column).

After getting enough information on each column, we tend to see that each column is directly or indirectly related to our target. So, we did not have any unnecessary columns.

To start the pre-processing, firstly we saw if there are any null values available in our dataset. There was none, so we moved on to see if there are any "0" values in our dataset. We saw that a couple of columns contain "0" values. This could result in a false prediction by the machine. So we imputed the values by using the "SimpleImputer" function and our strategy was to fill the "0" values with the "mean" values of the respective column.

As machines can not learn well with categorical values, we changed the values of the "Outcome" column by using "LabelEncoder". In this way, all the "Yes" became "1", and all the "No" became "0". Then we saw how many 1's and 0's are in our label column. There are 268 1's and 500 0's. The dataset is slightly biased towards 0's. But as our dataset is relatively small, it will not matter that much.

Then we divided the dataset into two separate datasets. One is for "Features" another is for "Label". Now it is time to split the dataset into train, test dataset. For our case, we chose to keep the ratio 75:25. Meaning that, 75% of the dataset we will be trained, and 25% of the dataset will be tested. Here we had two options to choose our values for splitting. One, we can let the machine choose the values randomly or we can stratify the system and keep the actual ratio of

1's and 0's. In this case, we wanted our machine to learn from everywhere of the dataset, so we let the machine, to choose values randomly.

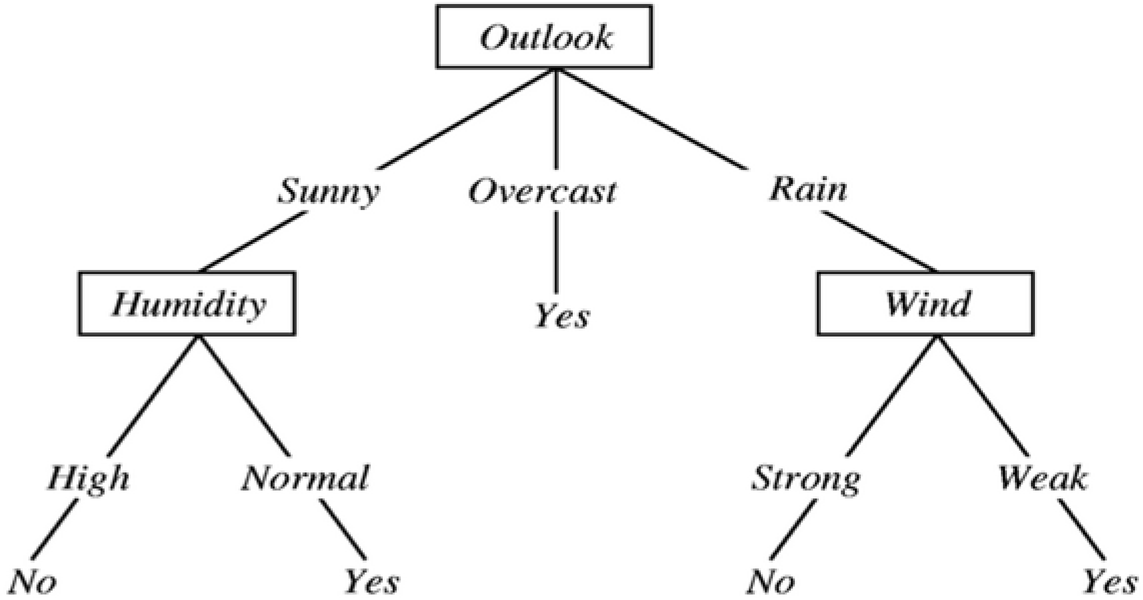
After getting done with the splitting, we scaled our feature columns so that all the values in the dataset come into one single range. We did this using "MinMaxScaler()". We scaled our X_train and X_test dataset.

At this point, we are done with the pre-processing and ready to run some models on our dataset.

Models applied

Decision Tree Classifier

A decision tree is basically a machine learning flow-chart which facilitates in predicting a decision from a set of attributes or features. As the name suggests, the tree will have a number of nodes or vertices with connecting edges or branches between them.



Let us take a look at the diagram above. The base of the decision tree is the root node. Other sub-nodes split out from this root node and diverge into decision nodes, connected by edges. The nodes which cannot be split further are the leaf nodes or the bottom nodes of the decision tree. The process of removing sub-nodes from a decision tree is called pruning. The edges or the branches of the tree lead in a route which is a consequence of the decision made prior to that edge. A decision tree leads us to a prediction of yes or no based on a number of decisions or nodes traversed.

If the dataset consists of N attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy.

We face a dilemma when we try to decide which feature will be at the root. Also, after expanding, which feature should we select. These are the difficulties we face while making the decision tree. If we randomly select any features, it may not be a good trained model and might lead us to a false path and the accuracy will also be low.

To solve this attribute selection problem, researchers worked and devised some solutions. They suggested using some criteria like :

- Entropy,
- Information gain,
- Gini index,
- Gain Ratio,
- Reduction in Variance
- Chi-Square

In our model, we worked with entropy.

Now what is entropy?

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.

Mathematically Entropy for 1 attribute is represented as:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Decision tree models are one of the predictive modeling approaches in machine learning and data science. This runs on specific algorithms which implement routes via which the dataset can be split, based on various criteria. Regression and classification are both crucial uses of the non-parametric decision tree. The target variable is basically the decision, that is, it is the dependent variable. The target variable can be assigned a discrete set of values to form a classification tree. On the other hand, if we assign a continuous set of real values to the target value, we form a regression tree. The other variables are just features of the tree.

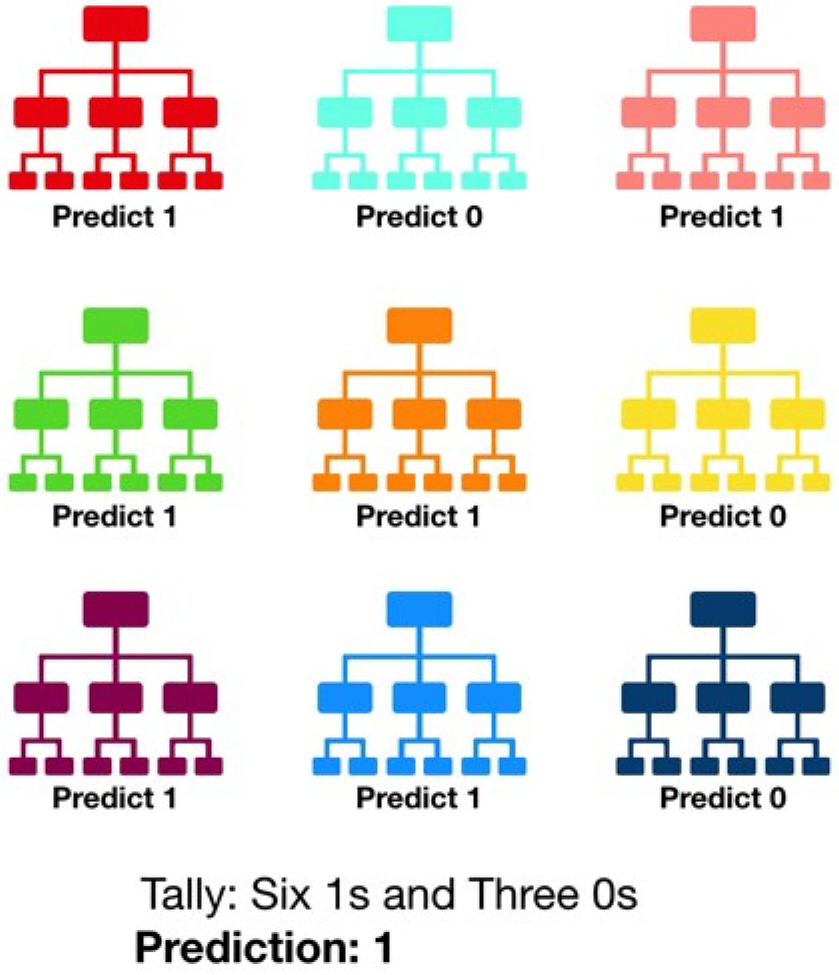
The splitting of nodes of a tree is conducted based on entropy and information gain. Entropy or information is the measured amount of impurity. The feature with the highest information gain will be split right after the root node and so on.

We use a top-down algorithm to partition the dataset into rows or columns of true and false. We take a random portion of the dataset or a random set of entries from the dataset to train it, keeping in mind that the ratio of yes to no of the portion has to equal that of the entire dataset. Next, we take another portion as the test model. Going with the flow, we perform feature scaling and use the confusion matrix to make predictions and check the accuracy of the model. If the accuracy is better than the initial accuracy, we can conclude that the pruned model is less complex and easier to understand than the initial model.

A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees. This algorithm is suitable for our dataset because the outcome of our dataset is also based on Yes/No for predicting diabetes.

Random Forest Classifier

Like its name suggests, a random forest is made up of numerous independent decision trees that work together as an ensemble. Every tree in the random forest spits out a class forecast, and the classification that receives the most votes becomes the prediction made by our model.



Random Forest has been one of the most popular research methods in the data mining area and information to the biological field. Since our dataset contains targeted values as Yes/No i.e 1/0 for diabetic and not diabetic respectively, the Random Forest classifier model is one of the best fit for our dataset.

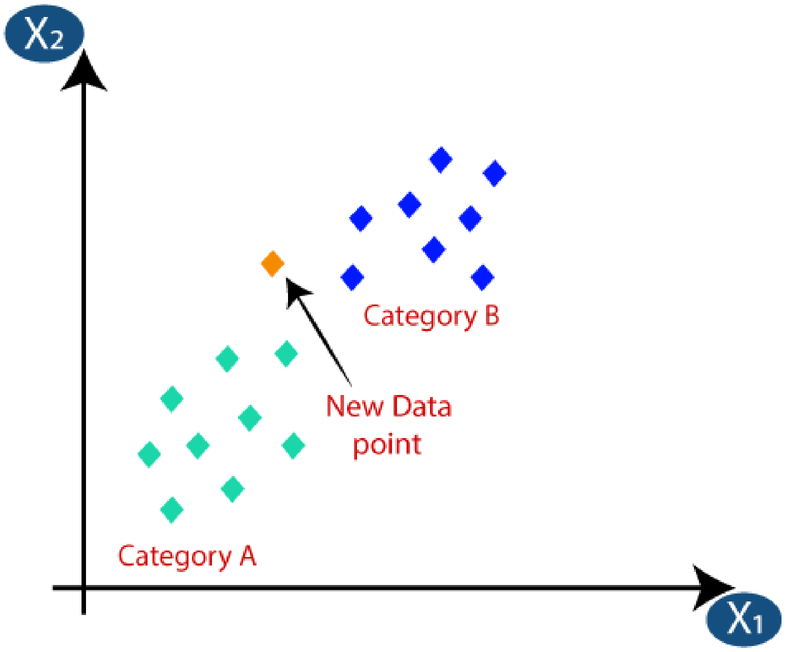
K-Nearest Neighbor(KNN) Classifier

The KNN algorithm is a simple algorithm that takes known clusters of data to predict how an unknown data will fit into it. It operates under the assumption that data near each other in the clusters are similar to each other. It uses Euclidean distance to calculate the distance.

Firstly, the training data is grouped into clusters on a 2d grid according to their classes. Then a new data point is added to the graph depending on its features. The number of nearest neighbors, k, to calculate is decided, in our case since we only have 2 classes diabetic and not, we used k =

1. By default the distance metric sklearn uses is the Euclidean distance, so the Euclidean distance to the first nearest neighbor is calculated and finally, that class is applied to the new data point. Using this we can classify all the test points in our data and check how accurate the algorithm was.

This model solely depends on the euclidean distance. But our dataset has many different features of which our target is dependent. Depending on the distance, the machine predicts some false outcome. That's why the K-nearest neighbor is not the best fitted model for our dataset. We kept this model to compare its accuracy score with the other models.

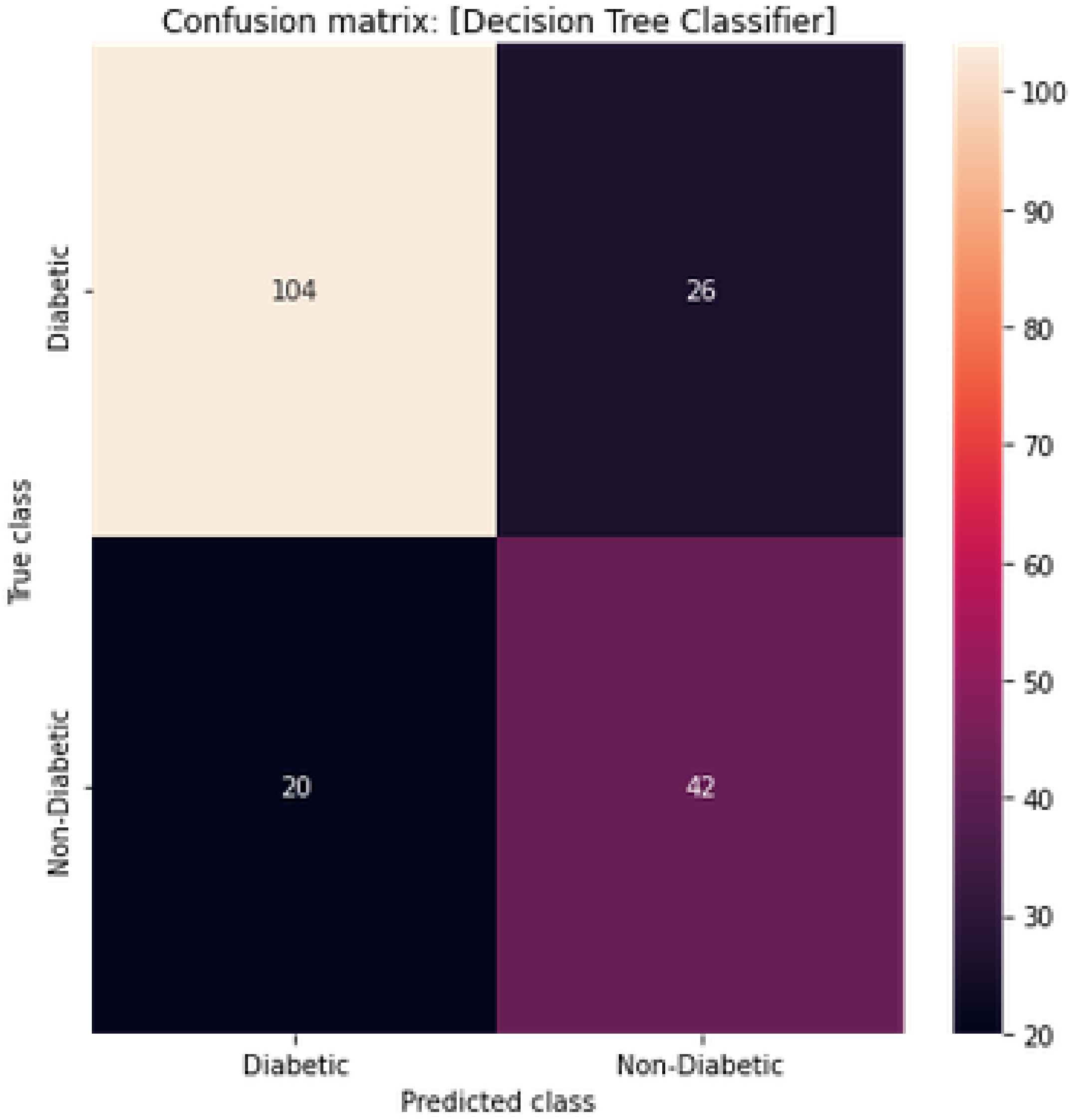


Results

As we have previously discussed, we have implemented three models on this dataset and have derived results such as accuracy for each of the models to compare and contrast amongst the three models. We can also see which model did the best by observing the confusion matrix.
The three models and their results have been given below:

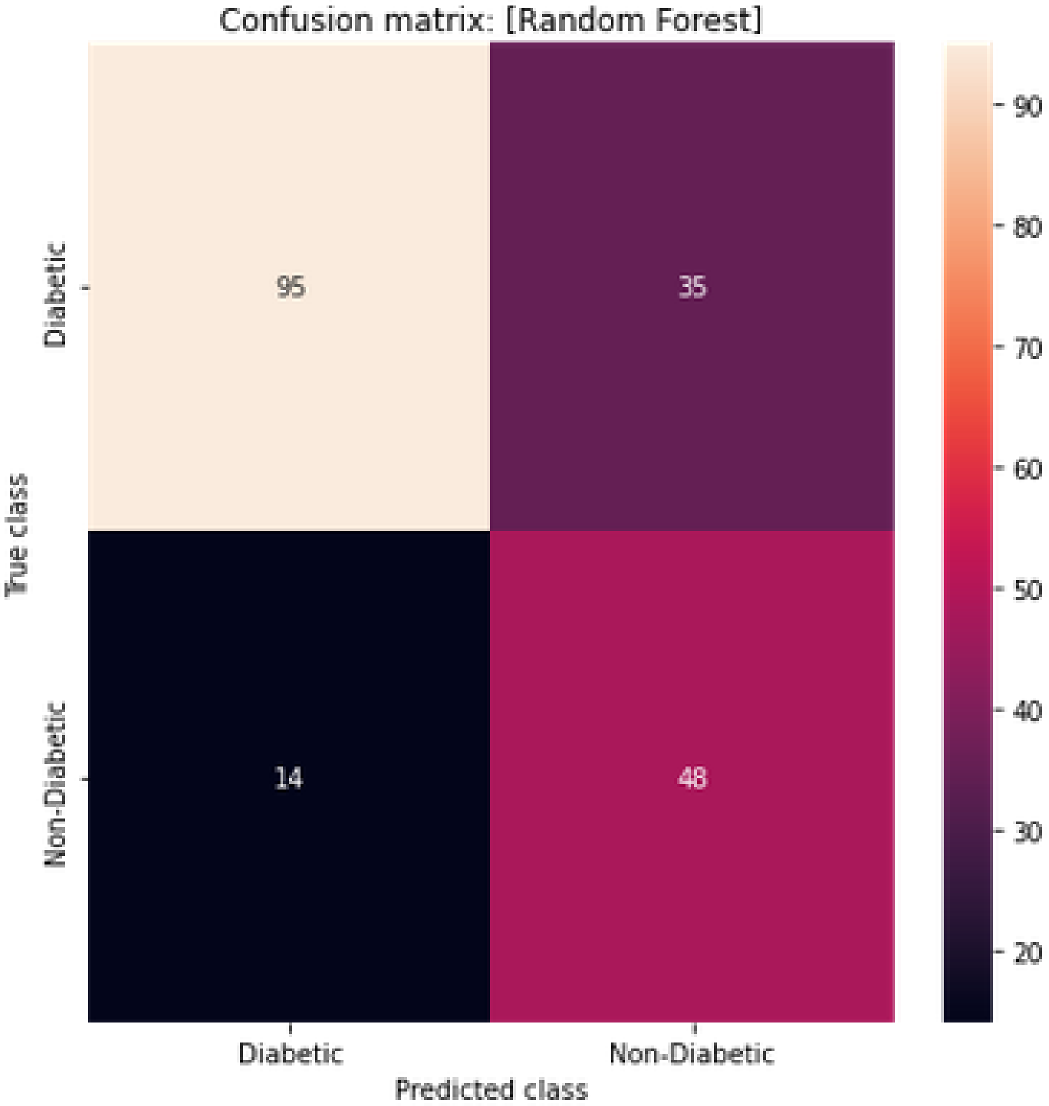
Decision Tree Classifier

- Accuracy: 60.42%
- Error: 39.58%



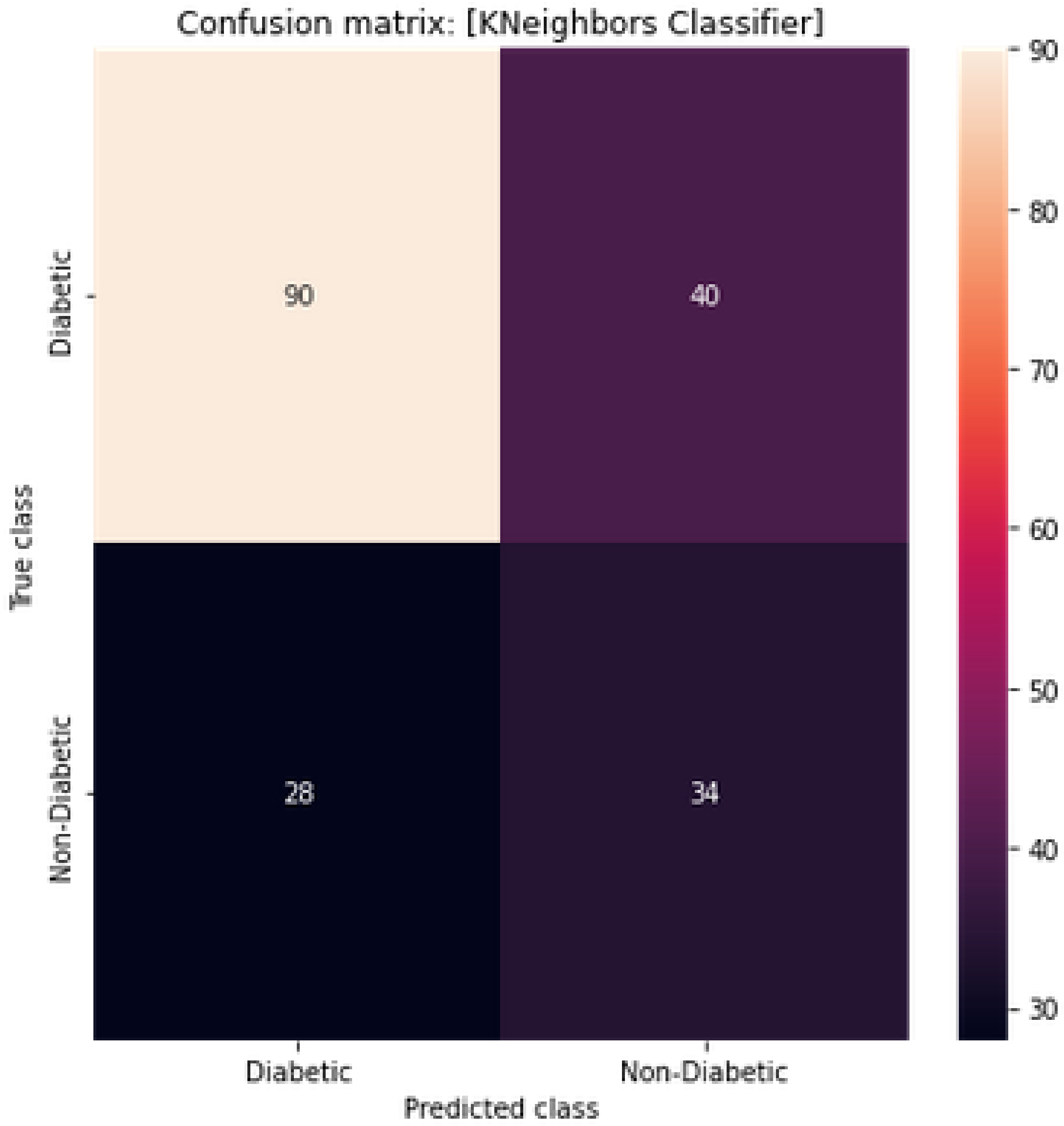
Random Forest Classifier

- Accuracy: 74.48%
- Error: 25.52%



K-Nearest Neighbor(KNN) Classifier

- Accuracy: 64.58%
- Error: 35.42%



Amongst the three models used, the one with the highest accuracy is Random Forest which makes it the best model amongst the three.

References

- https://www.who.int/health-topics/diabetes#tab=tab_2
- https://ourworldindata.org/grapher/diabetes-prevalence?time=2021
- https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
- https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
- https://towardsdatascience.com/understanding-random-forest-58381e0602d2