DIABETES PREDICTION USING MACHINE LEARNING ALGORITHMS

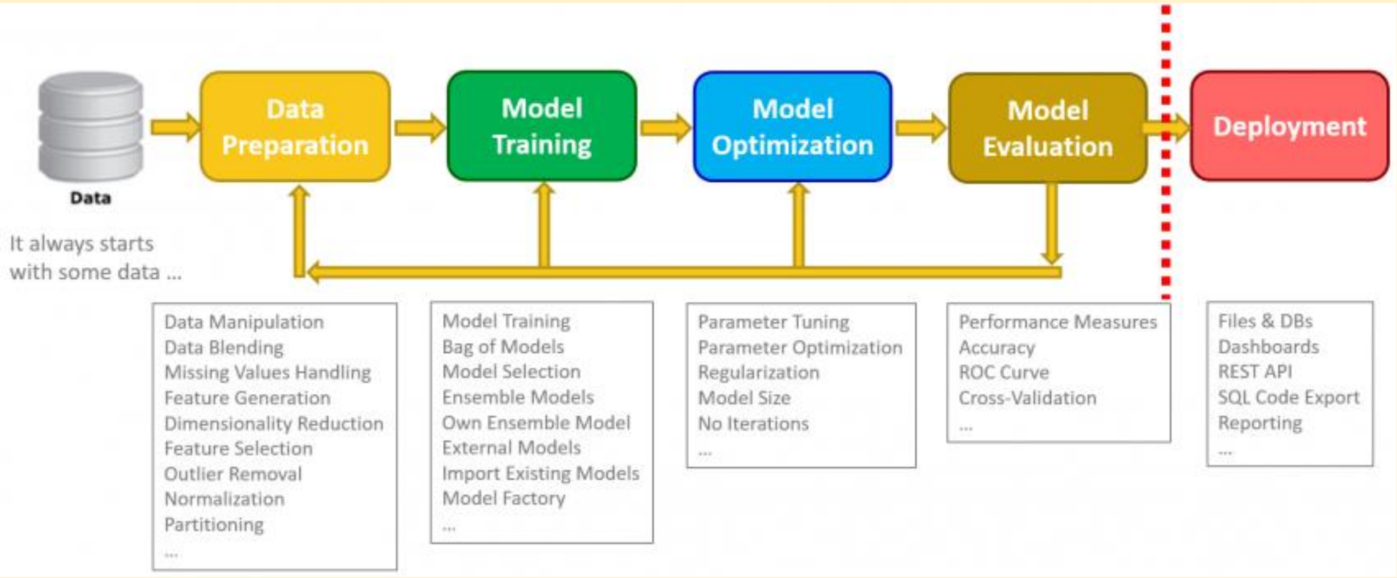
Problem definition:

Diabetes is a dangerous disease that results in too much sugar in the blood. The good news is that the complications from diabetes can often be prevented or delayed with quality medical care and by adopting healthy behaviour's to manage diabetes. Most of the people do not know that they will fall prey to this disease. So we have designed a model to know if a person will have diabetes or not based on a study of 750 people. We get the details of a person like insulin, glucose, age, pedigree function, BP, and predict the vulnerability.

Introduction:

We will predict if a patient has diabetes in future by looking at other patient's with similar medical history by classifying and clustering the dataset using logistic regression decision tree algorithms and K nearest neighbours algorithms finally we calculate the accuracy of our models to check how far our results are valid.

After reading the csv file and importing the dataset, we explore and analyse the data for reducing the error and to improve the performance of our model by dropping the unwanted attributes, removing the outliers, etc. After this, we standardize the dataset to reduce data redundancy and improve data integrity. Next, we cross validate this dataset and train the model to predict the outcomes. Finally, with this predicated values and actual values we calculate the accuracy score, AUC value, etc. We do the same for all three models, calculate their scores to find which model is best and use that model to get the input to predict the outcome.

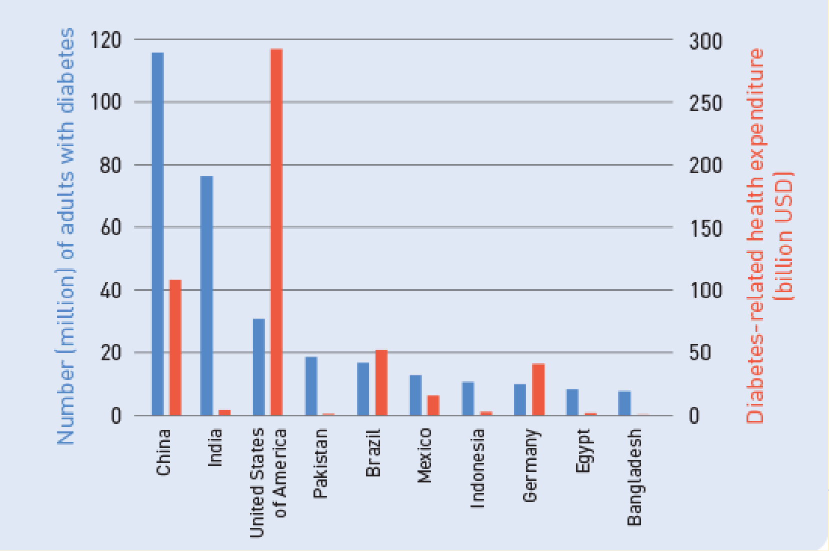


Literature survey:

Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Thanks to advances in machine learning and artificial intelligence, which enables the early detection and diagnosis of DM (Diabetes Mellitus) through an automated process which is more advantageous than a manual diagnosis. Currently, many articles are published on automatic DM detection, diagnosis, and self-management via machine learning and artificial intelligence techniques. This review delivers an analysis of the detection, diagnosis, and self-management techniques of DM from six different facets viz., datasets of DM, pre-processing methods, feature extraction methods, machine learning-based identification, classification, clustering and diagnosis of DM, artificial intelligence-based intelligent DM assistant and performance measures

In our model, we make use of logistic regression, K-Nearest neighbour, and decision trees to classify our data. In logistic regression, we basically decide with a threshold value above which we classify values into Class 1 and if the value goes below the threshold then we classify it in Class 2. whereas, in K-nearest neighbour, we assume the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

In 2018, 8.5% of adults aged 18 years and older had diabetes. In 2019, diabetes was the direct cause of 1.6 million deaths and in 2020 high blood glucose was the cause of another 2.2 million deaths. Between 2009 and 2021, there was a 5% increase in premature mortality from diabetes. In high-income countries the premature mortality rate due to diabetes decreased from 2009 to 2016 but then increased in 2016-2021. In lower-middle-income countries, the premature mortality rate due to diabetes increased across both periods. By contrast, the probability of dying from diabetes between the ages of 30 and 70 decreased by 18% globally between 2015 and 2021, Thanks to advances in machine learning and artificial intelligence, which enabled the early detection and diagnosis of diabetes.



System Description:

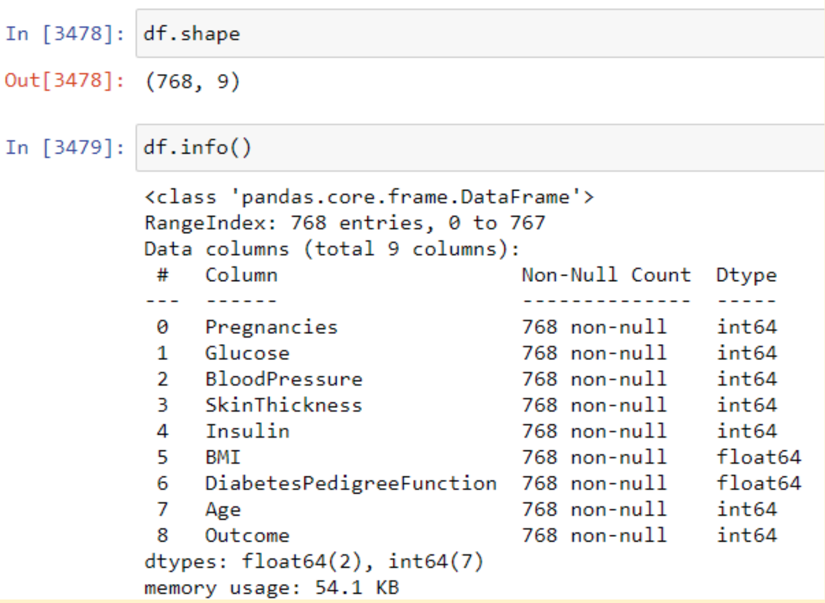
To run this code Basic system requirements are:

- Windows 10 pro
- 4 GB RAM
- Python Version 3.9
- Libraries used
    - **pandas**: To work with CSV files and data frames
    - **matplotlib**: To create charts using pyplot
    - **train_test_split**: To split the dataset into training and testing data
    - **Seaborn:** provides a high-level interface for drawing attractive and informative statistical graphics.
    - **sklearn:** It is used for classification, regression, clustering and dimensionality reduction.
    - **statsmodel:** It provides classes and functions for the estimation of many different statistical models

Results and inferences:

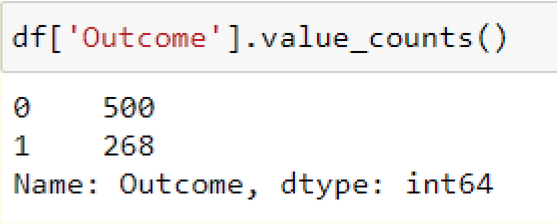
1. Exploratory data analysis:

It is a way of visualizing, summarizing and interpreting the information that is hidden in rows and column format. Once this is complete and insights are drawn, its feature can be used for supervised and unsupervised machine learning modelling.



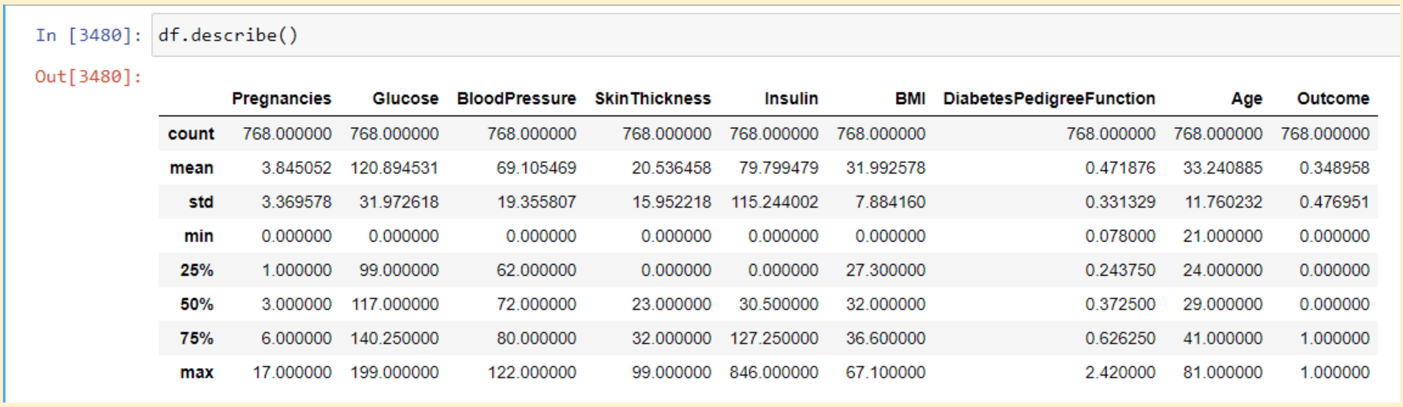
```
In [3478]: df.shape

Out[3478]: (768, 9)

In [3479]: df.info()

           <class 'pandas.core.frame.DataFrame'>
           RangeIndex: 768 entries, 0 to 767
           Data columns (total 9 columns):
            #   Column                    Non-Null Count  Dtype
           ---  ------                    --------------  -----
            0   Pregnancies               768 non-null    int64
            1   Glucose                   768 non-null    int64
            2   BloodPressure             768 non-null    int64
            3   SkinThickness             768 non-null    int64
            4   Insulin                   768 non-null    int64
            5   BMI                       768 non-null    float64
            6   DiabetesPedigreeFunction  768 non-null    float64
            7   Age                       768 non-null    int64
            8   Outcome                   768 non-null    int64
           dtypes: float64(2), int64(7)
           memory usage: 54.1 KB
```

Checking the dimensions, data type of all the features and any null values



```
df['Outcome'].value_counts()

0    500
1    268
Name: Outcome, dtype: int64
```

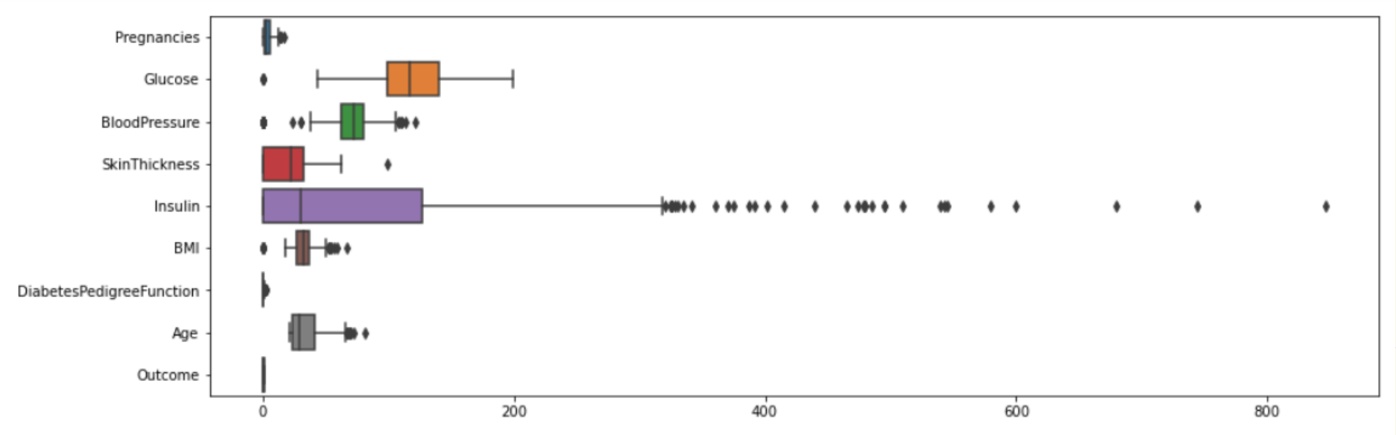
Here we can many outliers like zeros and some vague values:



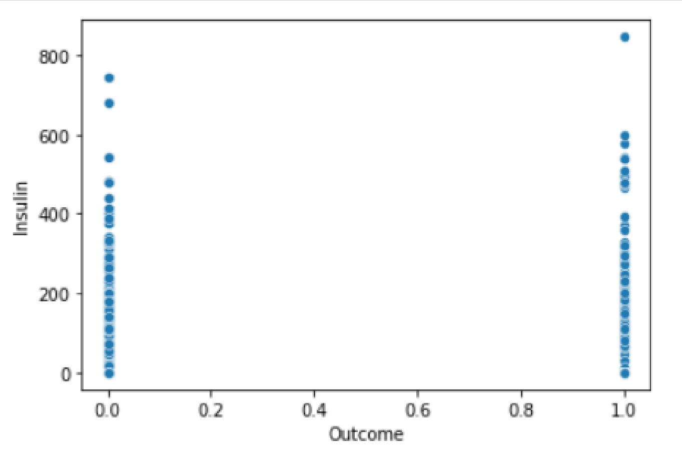
```
In [3480]: df.describe()

Out[3480]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

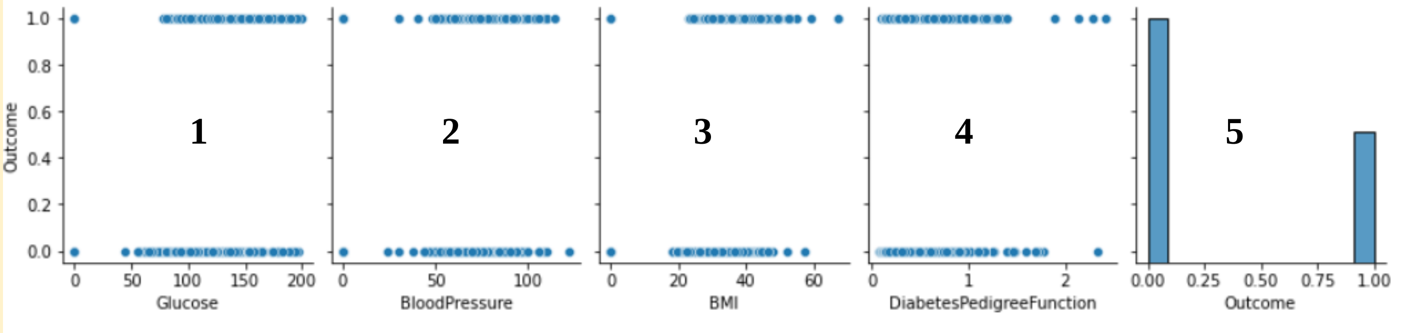
Bar plot of data with unwanted features and outliers:



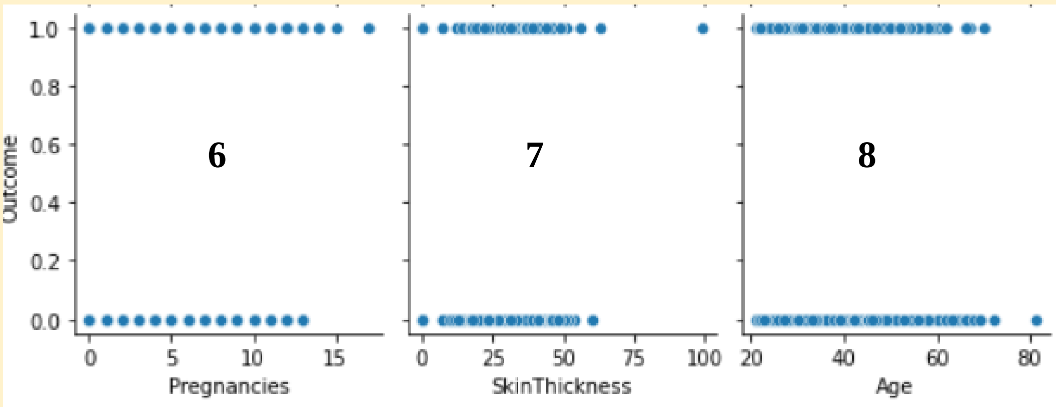
Checking if the features have valid outlier



Here we can see that when insulin level increases there is high probability of getting diabetes, so it's a valid outlier and couldn't be neglected.



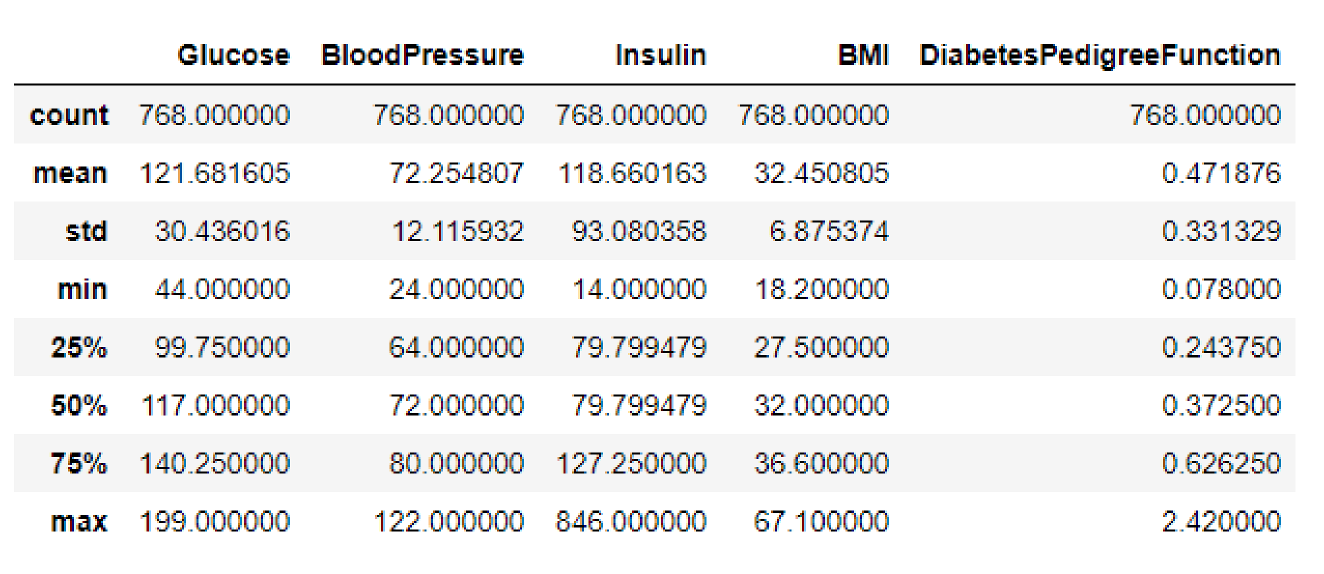
From graph **1, 2, 3, 4** we can infer that when the value of glucose, BMI, BP, DPF is high there is high probability that the person will have diabetes, from graph **5** we can infer that there are more patients doesn't have diabetes than the persons who have diabetes



From graph **6, 7, 8** we can infer that all the parameters doesn't show positive relation with the Outcome, Outcome is independent of pregnancies, Skin Thickness and age. These unwanted features decrease training speed, decrease model interpretability, and, most importantly, decrease generalization performance on the test set.

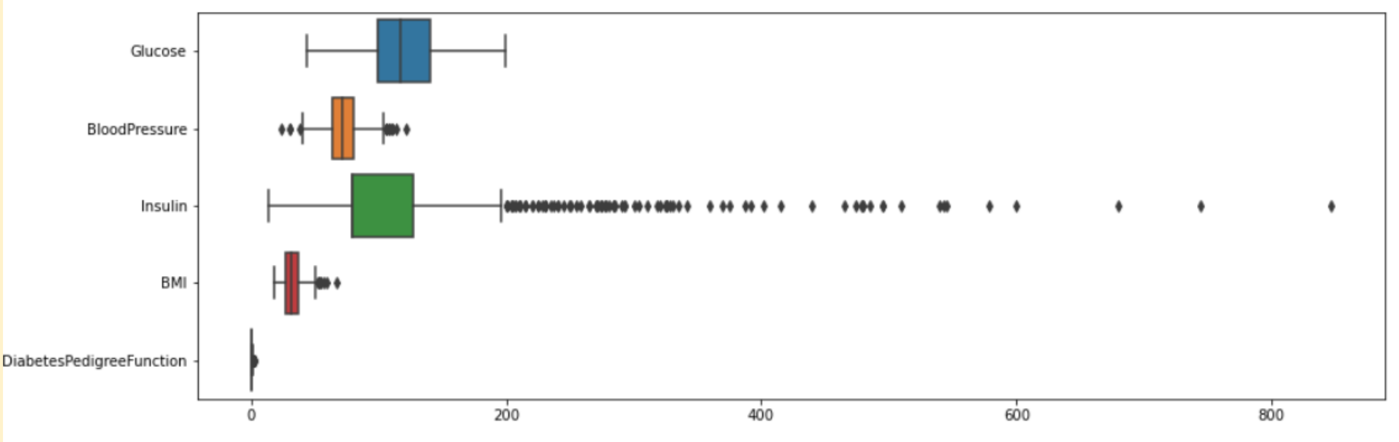
2. Removing unwanted features and outliers

For removing the outliers we just replace them with average of the values of respective features and we just drop the unwanted features like pregnancies, skin thickness and age.



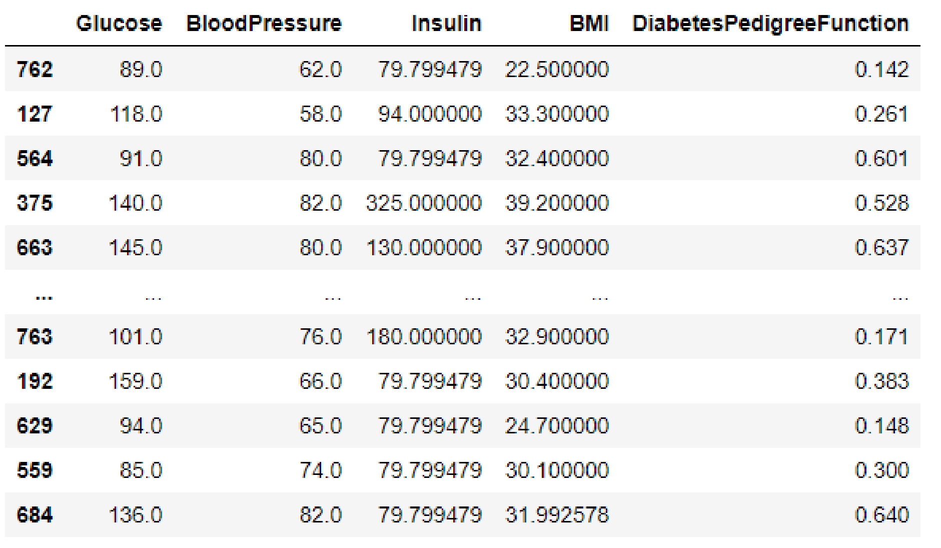
	Glucose	BloodPressure	Insulin	BMI	DiabetesPedigreeFunction
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	121.681605	72.254807	118.660163	32.450805	0.471876
std	30.436016	12.115932	93.080358	6.875374	0.331329
min	44.000000	24.000000	14.000000	18.200000	0.078000
25%	99.750000	64.000000	79.799479	27.500000	0.243750
50%	117.000000	72.000000	79.799479	32.000000	0.372500
75%	140.250000	80.000000	127.250000	36.600000	0.626250
max	199.000000	122.000000	846.000000	67.100000	2.420000

Box plot of the new data set, here we can see the outliers has been removed.

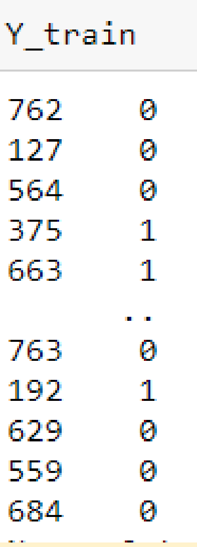


3. Modelling the data set

After cross-validating, we model the dataset by splitting the dataset into testing and training datasets , here are the values or X_train and Y_train , where X is the input features and Y is the outcome.



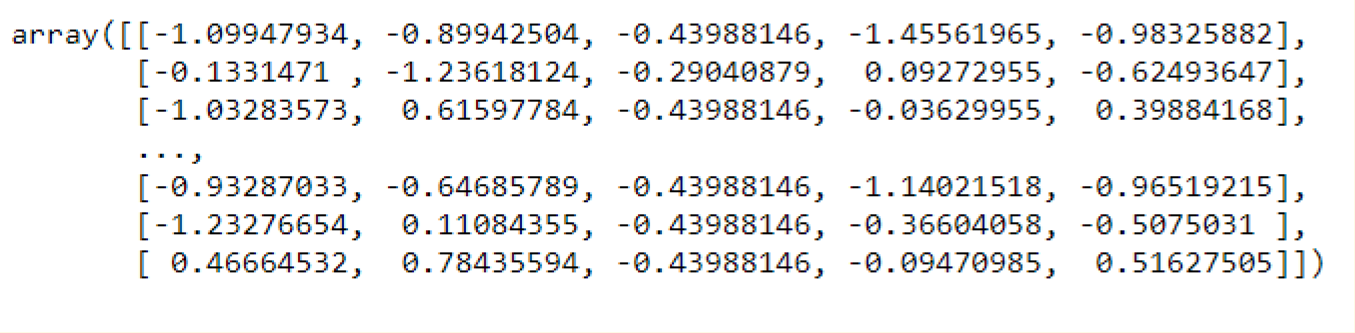
	Glucose	BloodPressure	Insulin	BMI	DiabetesPedigreeFunction
762	89.0	62.0	79.799479	22.500000	0.142
127	118.0	58.0	94.000000	33.300000	0.261
564	91.0	80.0	79.799479	32.400000	0.601
375	140.0	82.0	325.000000	39.200000	0.528
663	145.0	80.0	130.000000	37.900000	0.637
...	...	...	...	...	...
763	101.0	76.0	180.000000	32.900000	0.171
192	159.0	66.0	79.799479	30.400000	0.383
629	94.0	65.0	79.799479	24.700000	0.148
559	85.0	74.0	79.799479	30.100000	0.300
684	136.0	82.0	79.799479	31.992578	0.640



```
Y_train

762    0
127    0
564    0
375    1
663    1
      ..
763    0
192    1
629    0
559    0
684    0
```

4. Normalisation, standardization of dataset

We, do this step to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges. Standardizing the features around the centre and 0 with a standard deviation of 1 is important when we compare measurements that have different units. Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias.



```
array([[-1.09947934, -0.89942504, -0.43988146, -1.45561965, -0.98325882],
       [-0.1331471 , -1.23618124, -0.29040879,  0.09272955, -0.62493647],
       [-1.03283573,  0.61597784, -0.43988146, -0.03629955,  0.39884168],
       ...,
       [-0.93287033, -0.64685789, -0.43988146, -1.14021518, -0.96519215],
       [-1.23276654,  0.11084355, -0.43988146, -0.36604058, -0.5075031 ],
       [ 0.46664532,  0.78435594, -0.43988146, -0.09470985,  0.51627505]])
```

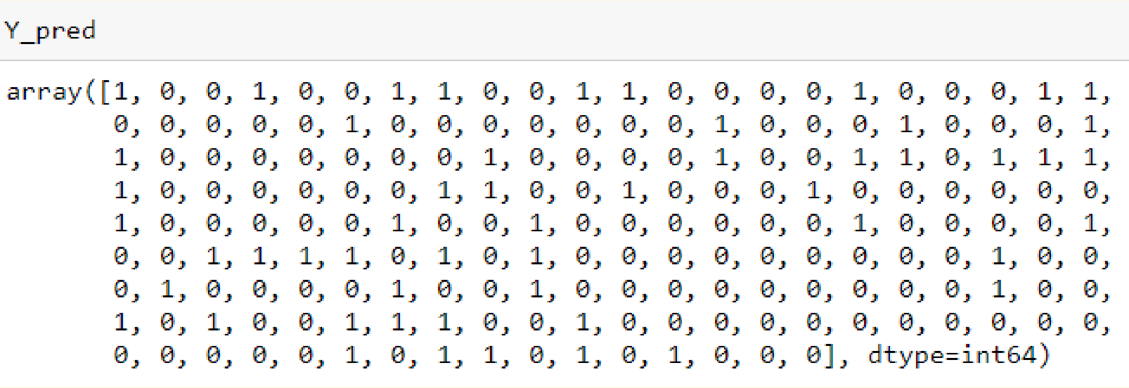
Values of X_train_std after standardizing.

5. Logistic Regression

We, use the Logistic regression modelling algorithm to predict the outcome. It is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability. We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.

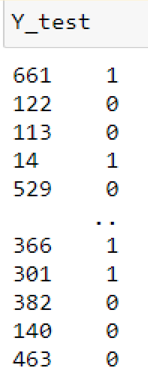
For Example, we have 2 classes, let’s take them like diabetes and no diabetes (1-diabetes, 0- no diabetes). We basically decide with a threshold value above which we classify values into Class 1 and if the value goes below the threshold then we classify it in Class 2.

Predicted values of outcome Y_pred and Y_test



```
Y_pred
```

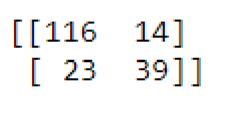
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0], dtype=int64)
```



```
Y_test
```

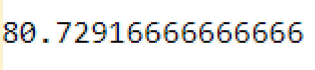
```
661    1
122    0
113    0
14     1
529    0
      ..
366    1
301    1
382    0
140    0
463    0
```

Confusion matrix of Y_pred and Y_test



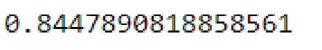
```
[[116  14]
 [ 23  39]]
```

Accuracy score



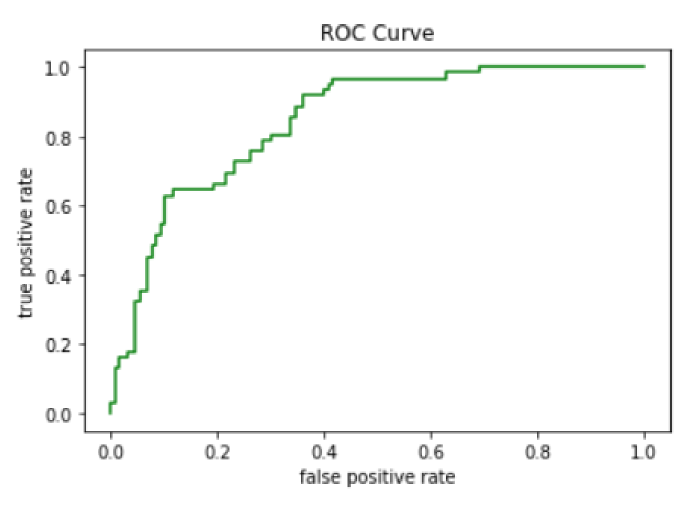
```
80.72916666666666
```

AUC



```
0.8447890818858561
```

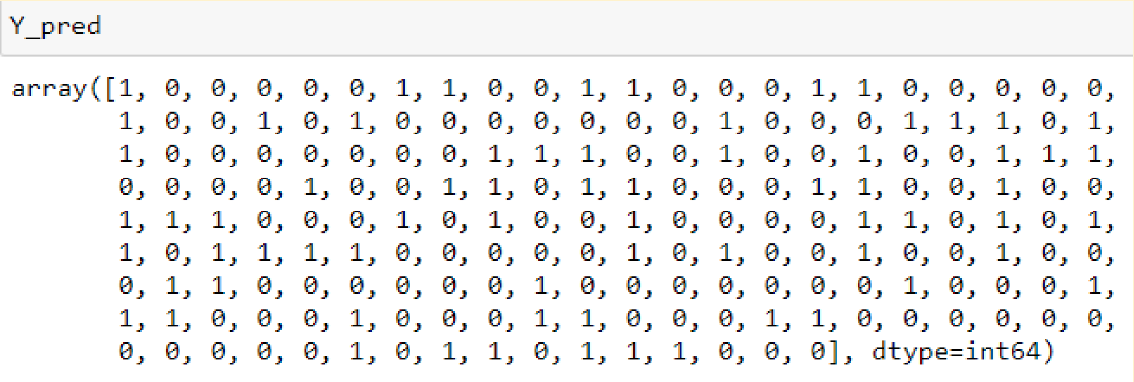
ROC curve



6. Decision tree

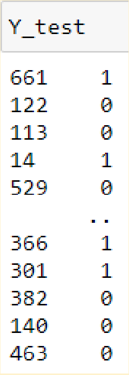
Here, we use the decision tree modelling algorithm to predict the outcome. In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value

Predicted values of outcome Y_pred and Y_test



```
Y_pred
```

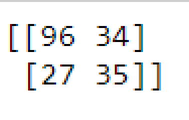
```
array([1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0], dtype=int64)
```



```
Y_test
```

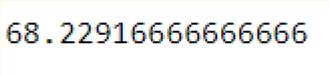
```
661    1
122    0
113    0
14     1
529    0
      ..
366    1
301    1
382    0
140    0
463    0
```

Confusion matrix of Y_pred and Y_test



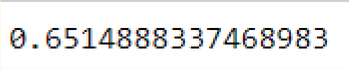
```
[[96 34]
 [27 35]]
```

Accuracy score



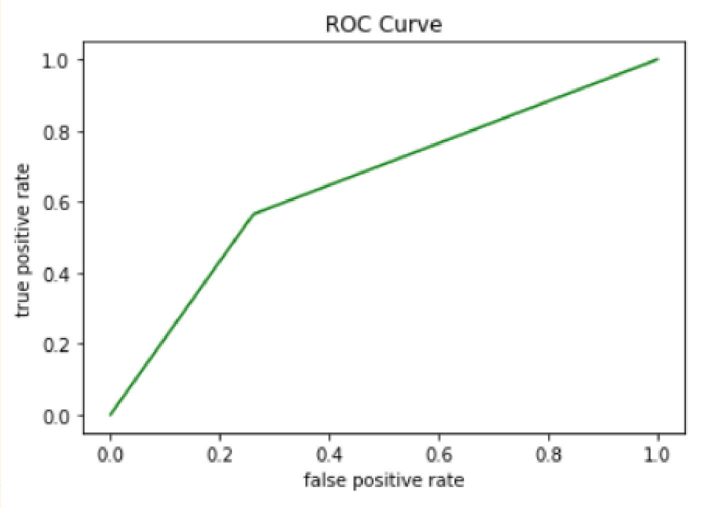
```
68.22916666666666
```

AUC



```
0.6514888337468983
```

ROC curve



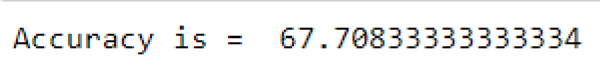
7. K nearest neighbours

Here, we use K nearest neighbour's algorithm to predict the outcome. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

In our model, we make use of logistic regression, K-Nearest neighbour, and decision trees to classify our data. In logistic regression, we basically decide with a threshold value above which we classify values into Class 1 and if the value goes below the threshold then we classify it in Class 2. whereas, in K-nearest neighbour, we assume the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

Accuracy score



```
Accuracy is =  67.70833333333334
```

8. Conclusion

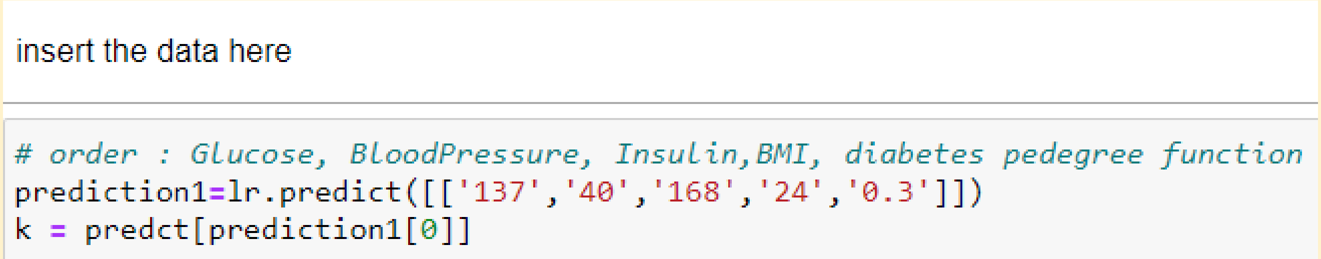
The project involved analysis of the diabetic patient dataset with proper data processing. Then, 3 models were trained and tested with maximum scores as follows:

- Logistic regression: 80.7%
- Decision Tree Classifier: 68.2%
- K Neighbours Classifier: 67.7%

So, we can see the accuracy value of logistics regression is much higher than that of KNN and decision tree classifiers. So we can infer that the Logistics regression is best in classifying the problems with 2 class labels (0 or 1 in our case).so we choose that for predicting our outcome

Predicting the outcome,

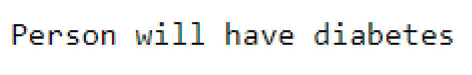
Input:



insert the data here

```
# order : Glucose, BloodPressure, Insulin,BMI, diabetes pedegree function
prediction1=lr.predict([['137','40','168','24','0.3']])
k = predct[prediction1[0]]
```

Output:



```
Person will have diabetes
```

References

- **Diabetes dataset :** https://drive.google.com/file/d/1buHIE9TgIKPTdJ_-KJrlPZfBf9LiATbp/view?usp=sharing
- https://www.who.int/
- https://www.sciencedirect.com
- Book: Artificial Intelligence: A Modern Approach by Stuart J. Russell and Peter Norvig