# Comparing Accuracy of Various Models in Predicting Diabetes in an Individual

(Sewanti Lahiri 20BEE0007, Siddhanta Mondal 20BEE0010)

**Abstract:** Diabetes is a chronic disease and around 1.5 to 5 million deaths occur every year due to it. In light of this, four neural network models have been developed to predict the likelihood of a person of being a diabetic. These models include Logistic Regression, Backward Propagation, Random Forest Classifier and Decision Tree architecture. All the models are developed and tested in Jupyter notebook. The highest accuracy reported is **77.9%**

## I. INTRODUCTION

Diabetes is a metabolic disease that causes high blood sugar. The hormone insulin moves sugar from the blood into your cells to be stored or used for energy. Untreated high blood sugar from diabetes can damage your nerves,eyes,kidneys, and other organs.The higher your blood sugar is and the longer you live with it, the greater your risk for complications. Complications associated with diabetes include heart disease, heart attack, stroke, neuropathy, nephropathy, retinopathy, vision loss, hearing loss, foot damage such as infections and sores that don't heal, depression, dementia and many more. Over 30 million have now been diagnosed with diabetes in India. The CPR (Crude prevalence rate) in urban areas of India is thought to be 9 per cent. In rural areas, the prevalence is approximately 3 percent of the total population. Women who have never had diabetes can suddenly develop gestational diabetes in pregnancy. About half of women with gestational diabetes will develop type 2 diabetes within 5 to 10 years of delivery according to the International Diabetes Federation (IDF). After this case study, it has become crucial for diabetes to be diagnosed as soon as possible. Diabetes can be predicted by lab tests and other methods but those become painful after continuing for a longer time. So, we have tried to train different models and compare their accuracy.

## II. DATASET

The dataset used for training and testing of these four models consists of 768 female subjects, out of which 268 are diabetic and 500 are non-diabetic. Apart from this, the dataset also contains eight important features: Pregnancies, Glucose level, Skin Thickness, Blood Pressure, Insulin, BMI, Age, Diabetes Pedigree function.

```
In [8]: df.head()
```

Out[8]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

Before using the dataset as training material, it was made to undergo several preprocessing to remove any anomalies whatsoever present. Initially, the database was checked for any zero values present in any column. Several such cases were found in the column of Insulin and skin thickness, which were replaced by the mean of the respective columns.
Then using the in-built library, sklearn.preprocessing, StandardScaler was imported.

```
In [23]: from sklearn.preprocessing import StandardScaler

In [24]: std = StandardScaler()
         X_train_std = std.fit_transform(X_train)
         X_test_std = std.transform(X_test)
```

StandardScaler standardizes the features by removing the mean and scaling to unit variance. The fit_transform() seen above is used for a training set which first calculates the mean and variance of each of the features and then transforms the features using the mean and variance. The transform() is however used for the testing set which performs the same thing as fit_transform().

After this we split the dataset into a training and testing set using the train_test_split() which is present in the sklearn.model_selection() . We have considered a 70:30 ratio for training and testing sets. Also, we applied a random_state attribute with a value of 10.

```
In [18]: from sklearn.model_selection import train_test_split

In [19]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.70, random_state = 10)
```

After this, preprocessing of the dataset is over, and we continue to use the dataset for training and testing our models.

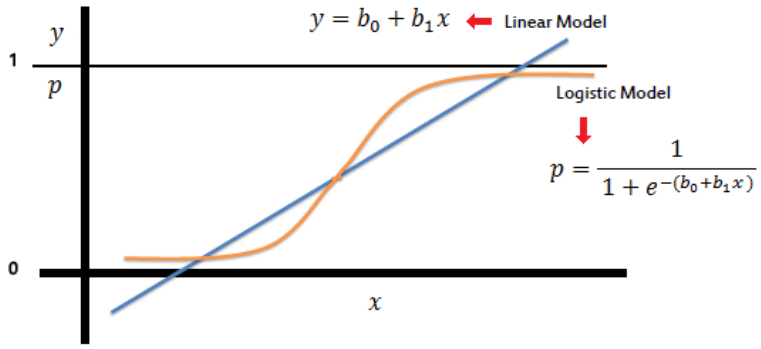
## III. LOGISTIC REGRESSION

The first model used for the prediction of Diabetes using the preprocessed dataset is the Logistic Regression neural network. It is similar to linear regression, instead of using a linear activation function, it uses the sigmoid function as the activation function.
The logistic function has asymptotes at 0 and 1 and it crosses the y-axis at 0.5.

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

This equation gives the predictive value of the output as 0, if the given inputs after assigning the given weights is a considerable negative value. Similarly, if the given input turns out to be a considerable positive value, we get output as 1.

Using this, a decision boundary is set to predict the class to which the data belongs.



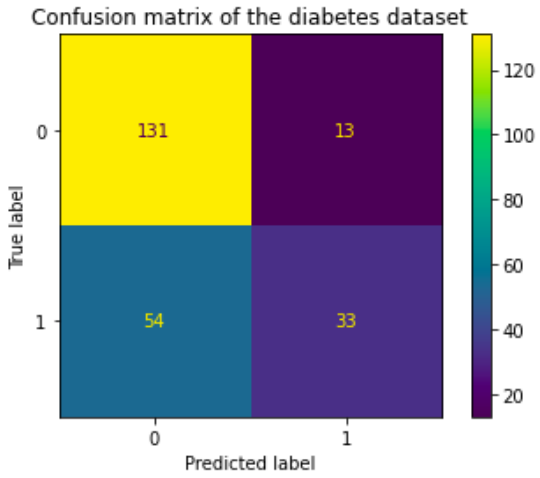

The library for Logistic Regression is present in the sklearn.linear_model, from which it is imported.

```
In [27]: from sklearn.linear_model import LogisticRegression
         lr = LogisticRegression()

In [28]: lr.fit(X_train_std, Y_train)
Out[28]: LogisticRegression()
```

Using the fit(), we tried to get accurate results from the training dataset. Following this, we tested the model on the testing set, and using the accuracy_score, imported from the library sklearn.metrics, we got an accuracy of **74.46%**

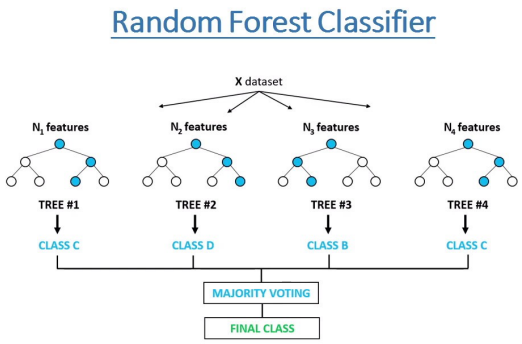
```
In [29]: from sklearn.metrics import accuracy_score

In [30]: Y_pred=lr.predict(X_test_std)

In [31]: accuracy_score(Y_test,Y_pred)
Out[31]: 0.7445887445887446
```

The more detailed result about the accuracy of the predictability is obtained from the following confusion matrix



## IV. **RANDOM FOREST CLASSIFIER:**

Random Forest is an ensemble classifier model using many decision tree models. Ensemble models combine the results from different models. Unlike the decision tree, it does not rely on a single decision. It assembles randomized decisions based on several decisions and makes the final decision based on majority.




The first step is to take a glance at choices and use the foundations of each indiscriminately created decision tree to predict the result and store the anticipated outcome at intervals at the target place. Secondly, we calculate the votes for each predicted target and ultimately, admit the high voted predicted target as a result of the ultimate prediction from the random forest formula. Some of the options of Random Forest do correct predictions results for a spread of applications are offered. Through model coaching the importance of every feature may be measured.

In python, the library for Random Forest Classifier is imported from sklearn.ensemble.

```
In [17]: import numpy as np
         import pandas as pd
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score
         import matplotlib.pyplot as plt
```

For getting the best results, we fixed the number of estimators to 1000, the maximum number of features considered while making a split was fixed to 5 and the minimum samples present in every leaf node was fixed to 3.

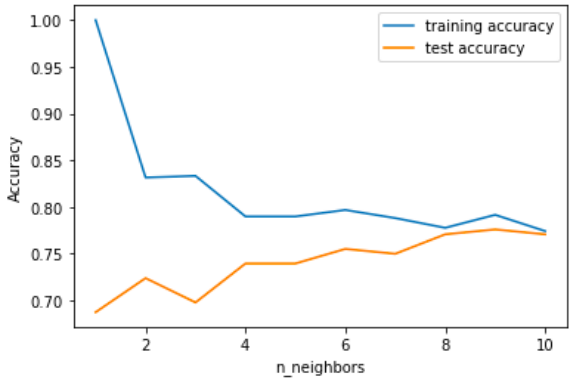
After these modifications, we successfully fitted the model using the fit().

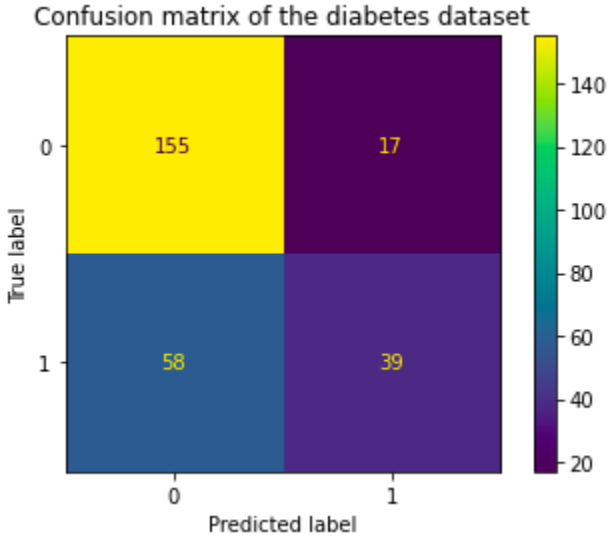
```
In [42]: #Make and fit the random forest to the training data
         randForest = RandomForestClassifier(n_estimators=1000, max_features = 5, min_samples_leaf = 3)
         randForest.fit(X_train, Y_train)
Out[42]: RandomForestClassifier(max_features=5, min_samples_leaf=3, n_estimators=1000)
```

Then using the accuracy_score() from the library sklearn.metrics, the accuracy of the model was found to be 75%.

```
In [43]: randForest.score(X_test,Y_test)
Out[43]: 0.75
```

To get a detailed result of the accuracy level of the model, we plotted a graph between the accuracy level and the nearest n neighbours for the training and the testing set. Along with this, we also plotted the confusion matrix.




## V. **BACKPROPAGATION:**

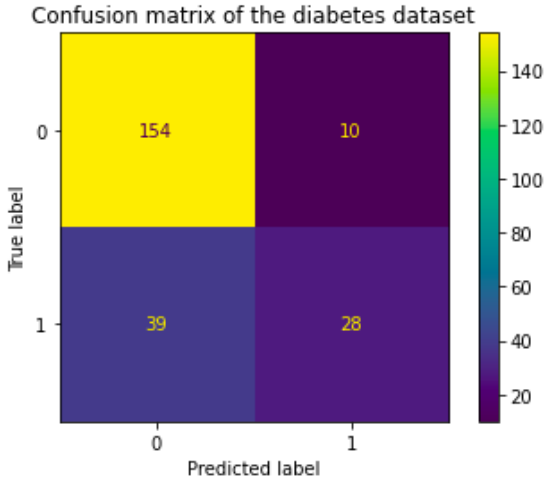
The third architecture that we developed for the prediction of diabetes is backpropagation. It's trained in such a way so that it is able to get the hidden layer errors.
We first provide inputs to the network and carry out the output but the output obtained can be anything because the first weights given to the input network are random numbers.
We then find the error in one of the neurons using 'Error=Output(1-Output)(Target-Output).

Then 'output(1-output)' is used in the equation when the sigmoid function is there. But Target-Output is used if there is only a threshold neuron. Then the weights are updated. After updating the weight errors for the hidden layers, neurons are supposed to be calculated but targets are not available. Now we apply backpropagation from the output layer. After obtaining the error from the hidden layer neurons, the hidden layer weights are updated and the procedure continues.

After implementing the above model in python, we got an accuracy level of **77.9**%

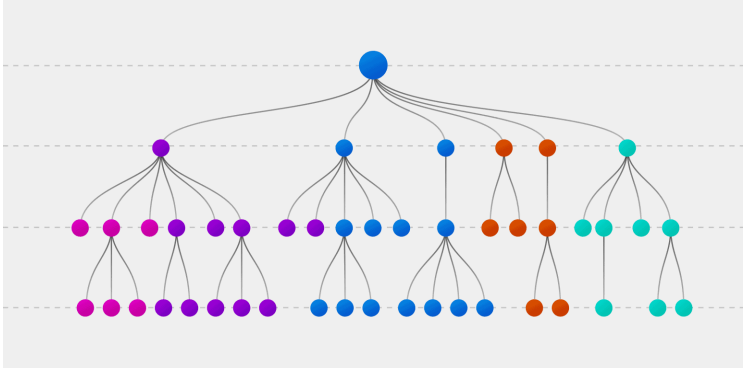
```
In [35]: from sklearn.metrics import accuracy_score
         print(accuracy_score(Y_test,Y_pred))

         0.7792207792207793
```

To get further information about the accuracy level, we plotted the confusion matrix as well.




## VI. **DECISION TREE:**

The fourth and the last architecture developed for the prediction of diabetes is the Decision tree. It consists of a flowchart like structure in which each internal node refers to a test, each branch refers to the outcome of the test. It is a supervised learning model that predicts the target by learning simple decision rules inferred from the data features. The main objective of using Decision Tree in this research work is the prediction of the target class using decision rules taken from prior data. It uses nodes and internodes for the prediction and classification. Root nodes classify the instances with different features. Root nodes can have two or more branches while the leaf nodes represent classification. In every stage, Decision tree chooses each node by evaluating the highest information gain among all the attributes



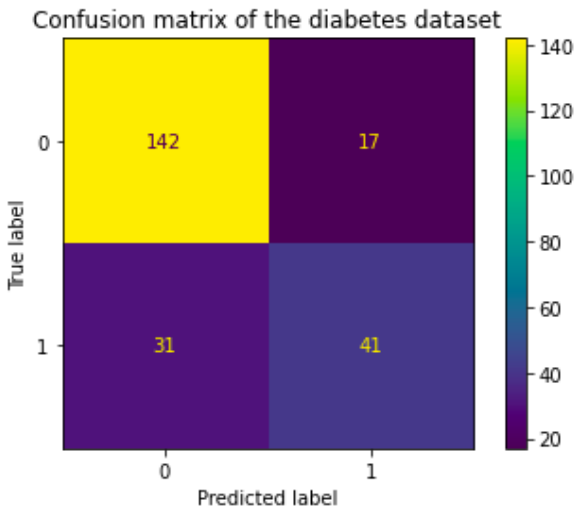
The library for Decision Tree Classifier is imported from sklearn.tree.
Followed by, we used the fit() upon the training sets and then tested the model on the testing set.

```
In [48]: from sklearn.tree import DecisionTreeClassifier

In [49]: dt = DecisionTreeClassifier()

In [50]: dt.fit(X_train_std, Y_train)
Out[50]: DecisionTreeClassifier()
```

After testing the model, the accuracy score was generated using the accuracy_score() imported from the library sklearn.metrics. It reported an accuracy of **72.29**%.

```
In [30]: from sklearn.metrics import accuracy_score
         accuracy_score(Y_test, Y_pred)
Out[30]: 0.7229437229437229
```

Finally, for getting the better understanding of the results, the confusion matrix of the Decision tree classifier was plotted:




## VII. RESULT:

This work presented four neural network models with different architectures. These models are used to predict the probability of diabetes in an individual using 8 important attributes. The models, Logistic Regression, Random Forest Classifier, Backpropagation and Decision Tree reported accuracies of 74.46%, 75%, 77.92% and 72.29% respectively. So, from these, we can infer that the Backpropagation model is the best among these four that can be used to predict diabetes in an individual with high accuracy.

## VIII. ACKNOWLEDGEMENT

We would like to thank our guide Dr. Himadri Lala for his valuable suggestion, guidance and advice for the completion of this project.

## IX. REFERENCES:

[1]2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE) Detection and Prediction of Diabetes Mellitus Using Back-Propagation Neural Network. DOI Bookmark:10.1109/ICMETE.2016.11

[2] Detection and Prediction of Diabetes Mellitus Using Back-Propagation Neural Network. DOI:10.1109/ICMETE.2016.11

[3]https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674

[4] Random Forest Algorithm for the Prediction of Diabetes. DOI: 10.1109/ICSCAN.2019.8878802

[5]On Diabetes Classification and Prediction Using Artificial Neural Networks Maha S. Diab, Saddam Husain and Anwar Jarndal **DOI:** 10.1109/CCCI49893.2020.9256621

[6]Diabetics Prediction using Logistic Regression in Python. https://medium.com/@pragya_paudyal/diabetics-prediction-using-logistic-regression-in-python-e51b90630f2f

[7] Diabetics prediction using logistic regression predicting whether the person is having diabetics or not. https://www.kaggle.com/kandij/diabetes-dataset

[8]Machine Learning based Diabetes Prediction using Decision Tree J48 **DOI:** 10.1109/ICISS49785.2020.9316001

[9]A hybrid prediction model for type 2 diabetes using K-means and decision tree. **DOI:** 10.1109/ICSESS.2017.8342938