DIABETES PREDICTION USING MACHINE LEARNING

Submitted by :- Tanushree Kanojiya

To :- Prof. Irfan Shaikh

Index

Introduction

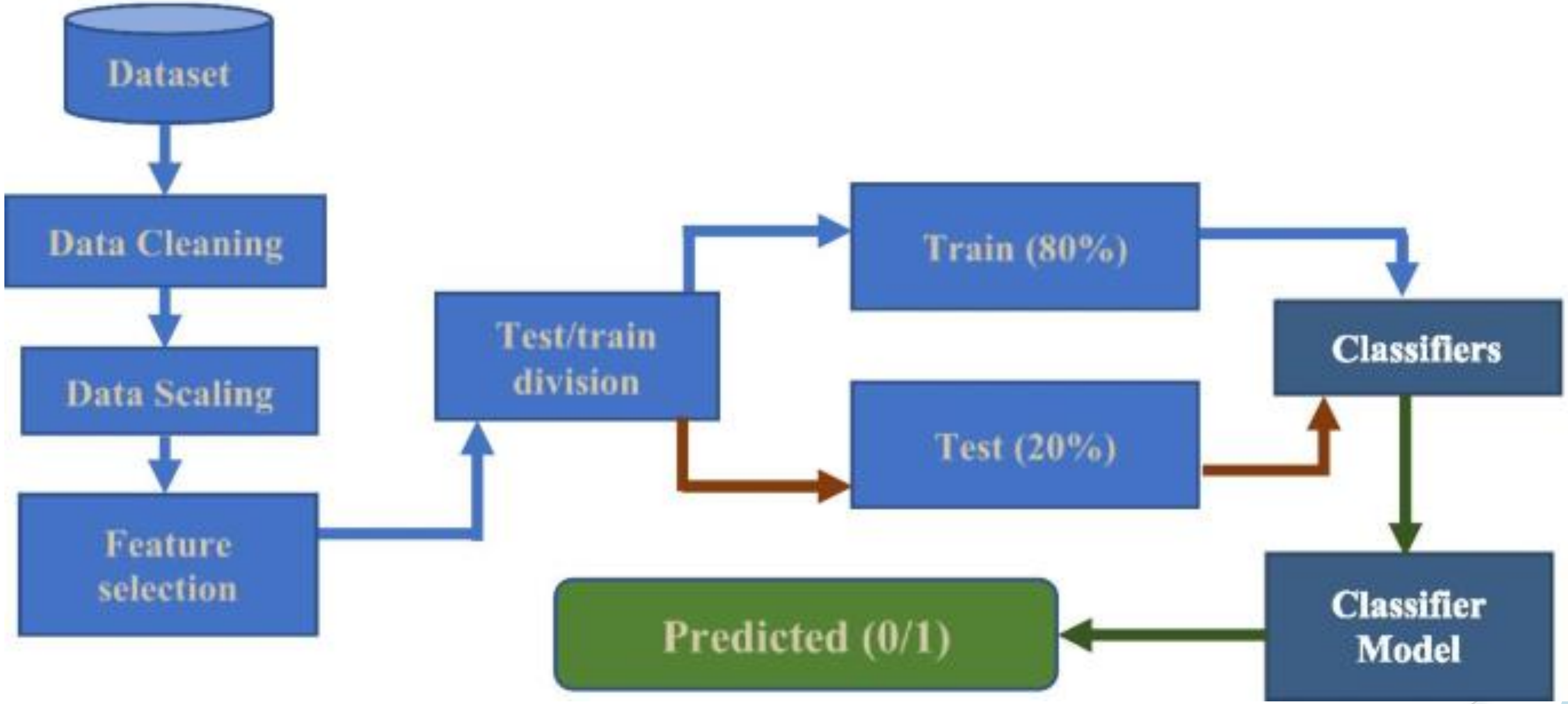
- Diabetes, also known as diabetes mellitus, is a group of endocrine diseases that cause high blood sugar levels.
- It occurs when the pancreas doesn't produce enough insulin or the body's cells don't respond properly to insulin.
- Diabetes is a chronic disease that affects millions of people worldwide. Early detection and accurate diagnosis are crucial for effective treatment and management.
- The most common long-term diabetes-related health problems are: damage to the large blood vessels of the heart, brain and legs (macrovascular complications) damage to the small blood vessels, causing problems in the eyes, kidneys, feet and nerves (microvascular complications).

Objective

To develop a robust machine learning model capable of accurately predicting the likelihood of diabetes based on various features.

This project aims to leverage data science techniques to enhance early detection of diabetes, ultimately contributing to improved healthcare outcomes and patient well-being.

Process Flow



Tools and Platforms used for Model Building

- Tools : Python, Tableau
- Platform : Jupyter Notebook
- Library Used : Scikit-learn, Matplotlib

Dataset

Source of data :- **kaggle** (https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset)

Data description :-

- Diabetes – This is a target variable containing two classes 0 and 1 . 1 for prediabetes or diabetes and 0 for no diabetes.
- HighBP – This variable shows if a person has high blood pressure or not.
- HighChol – This variable shows if a person has high cholesterol present or not.
- CholCheck – Cholesterol check in 5 years.
- BMI – Body Mass Index of a person.

- Smoker – Tells if person is a smoker or a non-smoker.
- Stroke - Had a stroke or not.
- HeartDisease_or_Attack – Tells if person has any heart disease or had any attacks in past.
- PhysActivity - Person's physical activity status in past 30 days.
- Fruits_consumption – Consumption of fruits by patient 1 or more times per day.
- Veggies_consumption - Consumption of veggies.
- HvyAlcoholConsump - Is there heavy alcohol consumption.
- AnyHealthcare – Any healthcare taken or any insurance taken.
- GenHlth – How good is the general health on the scale of 1 to 5. 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor
- MentHlth – Status of mental health.
- DiffWalk - Is there any difficulty in walking.
- Sex – Female or male.
- Age – Age of the person.

Exploratory Data Analysis(EDA)

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense(analyze) of the data you have.

- Data Importing

In [2]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
df = pd.read_csv(r"C:\Users\tanus\Downloads\diabetes_data.csv")
df.head(5)
```

Out[3]:

	Diabetes	HighBP	High_Cholesterol	CholCheck	BMI	Smoker	Stroke	HeartDisease_or_Attack	PhysActivity	Fruits_consumption	Veggies_consumption
0	0	1	0	1	26	0	0	0	1	0	1
1	0	1	1	1	26	1	1	0	0	1	0
2	0	0	0	1	26	0	0	0	1	1	1
3	0	1	1	1	28	1	0	0	1	1	1
4	0	0	0	1	29	1	0	0	1	1	1

Number of rows and columns :

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Diabetes              70692 non-null  int64
 1   HighBP                70692 non-null  int64
 2   High_Cholesterol      70692 non-null  int64
 3   CholCheck             70692 non-null  int64
 4   BMI                   70692 non-null  int64
 5   Smoker                70692 non-null  int64
 6   Stroke                70692 non-null  int64
 7   HeartDisease_or_Attack  70692 non-null  int64
 8   PhysActivity          70692 non-null  int64
 9   Fruits_consumption    70692 non-null  int64
 10  Veggies_consumption   70692 non-null  int64
 11  HvyAlcoholConsump     70692 non-null  int64
 12  AnyHealthcare         70692 non-null  int64
 13  GenHlth               70692 non-null  int64
```

- There are 70,692 rows and 19 columns in dataset.

- Data Cleaning

Missing values in data

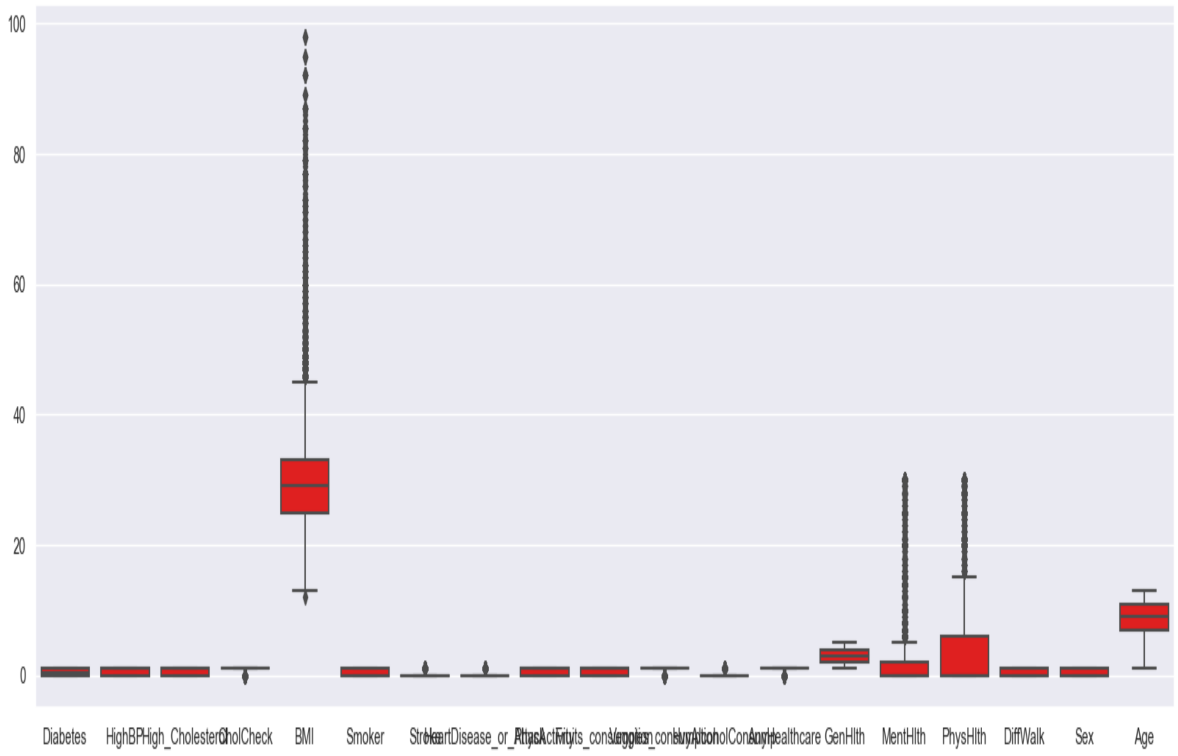
```
df.isnull().sum()
```

```
Diabetes                    0
HighBP                      0
High_Cholesterol            0
CholCheck                   0
BMI                         0
Smoker                      0
Stroke                      0
HeartDisease_or_Attack      0
PhysActivity                0
Fruits_consumption          0
Veggies_consumption         0
HvyAlcoholConsump           0
AnyHealthcare               0
GenHlth                     0
MentHlth                    0
PhysHlth                    0
DiffWalk                    0
Sex                         0
Age                         0
dtype: int64
```

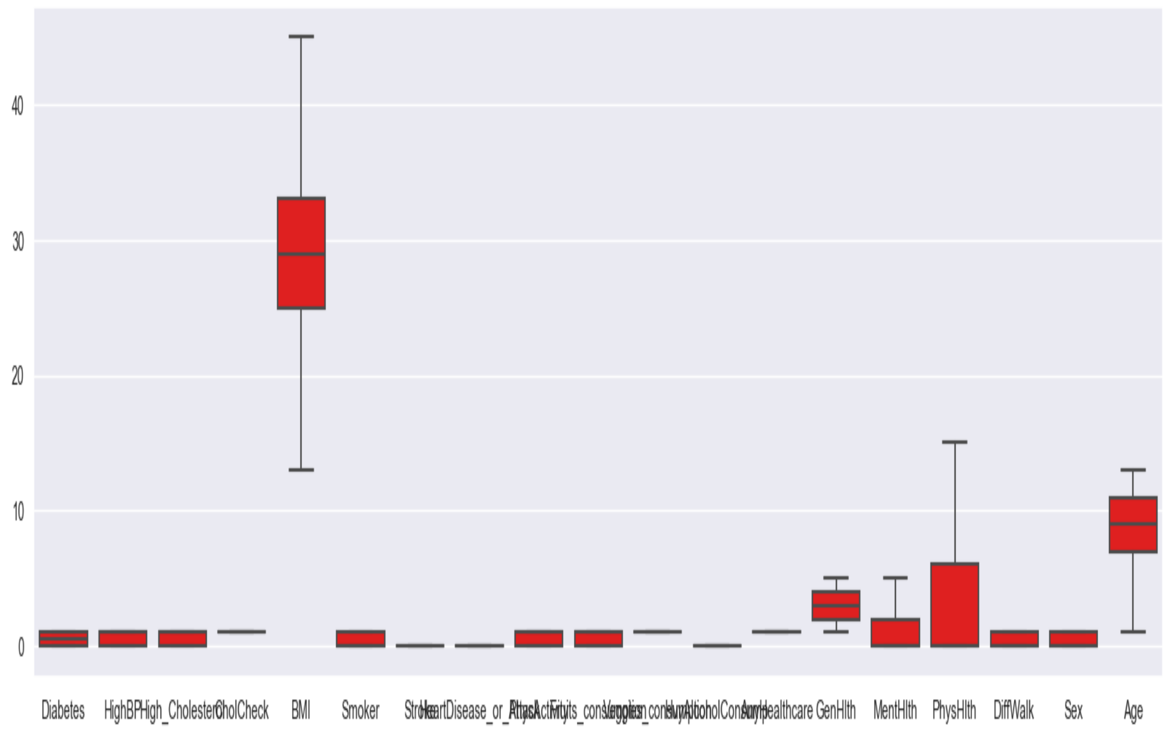
- There are no missing values in data.

Treating Outliers

Before removing outlier



After removing outlier



Data Partition for building models

- Data splitting is a machine learning technique that involves dividing data into subsets for training and testing.
- 80% of data is taken for training and remaining 20% is taken for testing.
- Subset of data is further divided into X_train, Y_train, X_test, Y_test.

```
from sklearn.model_selection import train_test_split

X = df.drop('Diabetes', axis = 1)
Y = df['Diabetes']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state=56)
```

Models Used for Prediction

- **Logistics Regression**

  - *Logistic regression is a data analysis technique that estimates the probability of an event occurring. It makes predictions based on probability.*

- **Decision Tree**

  - *Decision trees are hierarchical, tree-like structures made up of a root node, branches, internal nodes, and leaf nodes. It makes predictions and categorize based on how a previous set of questions were answered.*

- **Random Forest**

  - *A random forest (RF) is a machine learning algorithm that combines the output of multiple decision trees to produce a single result.*

Logistic Regression

- Model

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(multi_class='multinomial')
```

```
LogisticRegression
LogisticRegression(multi_class='multinomial')
```

The 5 features selected are :-

1. HighBP
2. BMI
3. GenHlth
4. PhysHlth
5. Age

Classification report

- **Training data**

Accuracy of Bad Customer Capture by Model is 76% (Sensitivity)

Accuracy of Good Customer Capture by Model is 71% (Specificity)

Accuracy = 74%

Hence model is a good fit on training data.

Classification Report for Training Data:

	precision	recall	f1-score	support
0	0.75	0.71	0.73	28205
1	0.73	0.76	0.74	28348
accuracy			0.74	56553
macro avg	0.74	0.74	0.74	56553
weighted avg	0.74	0.74	0.74	56553

- **Testing data**

Accuracy of Bad Customer Capture by Model is 77% (Sensitivity)

Accuracy of Good Customer Capture by Model is 72% (Specificity)

Accuracy = 75%

Hence model is a good fit on testing data as well.

Classification Report for Test Data:

	precision	recall	f1-score	support
0	0.76	0.72	0.74	7141
1	0.73	0.77	0.75	6998
accuracy			0.75	14139
macro avg	0.75	0.75	0.75	14139
weighted avg	0.75	0.75	0.75	14139

Decision Tree

- Model

```
from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor

dt = DecisionTreeClassifier()  # by default it use Gini index for split
dt.fit(X_train,y_train)  # Model = dt
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

- Model improvement by Pruning

```
from sklearn.tree import DecisionTreeClassifier

dt1 = DecisionTreeClassifier(criterion='gini',  #splitter
                             min_samples_leaf=100, ## child
                             min_samples_split=150, #parent
                             max_depth=4)  #branches
#Train the model using the training sets
dt1.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, min_samples_leaf=100, min_samples_split=150)
```

Classification report Before Pruning

- **Training data**

Accuracy of Bad Customer Capture by Model is 92% (Sensitivity)

Accuracy of Good Customer Capture by Model is 98% (Specificity)

Accuracy = 95%

Hence model is overfitting on training data.

	precision	recall	f1-score	support
0	0.93	0.98	0.95	28205
1	0.98	0.92	0.95	28348
accuracy			0.95	56553
macro avg	0.95	0.95	0.95	56553
weighted avg	0.95	0.95	0.95	56553

- **Testing data**

Accuracy of Bad Customer Capture by Model is 64% (Sensitivity)

Accuracy of Good Customer Capture by Model is 69% (Specificity)

Accuracy = 66%

Hence model is not a good fit on testing data.

	precision	recall	f1-score	support
0	0.66	0.69	0.67	7141
1	0.67	0.64	0.65	6998
accuracy			0.66	14139
macro avg	0.66	0.66	0.66	14139
weighted avg	0.66	0.66	0.66	14139

Classification report after Pruning

- **Training data**

Accuracy of Bad Customer Capture by Model is 77% (Sensitivity)

Accuracy of Good Customer Capture by Model is 69% (Specificity)

Accuracy = 73%

Hence model is a good fit on training data.

	precision	recall	f1-score	support
0	0.75	0.69	0.72	28205
1	0.71	0.77	0.74	28348
accuracy			0.73	56553
macro avg	0.73	0.73	0.73	56553
weighted avg	0.73	0.73	0.73	56553

- **Testing data**

Accuracy of Bad Customer Capture by Model is 76% (Sensitivity)

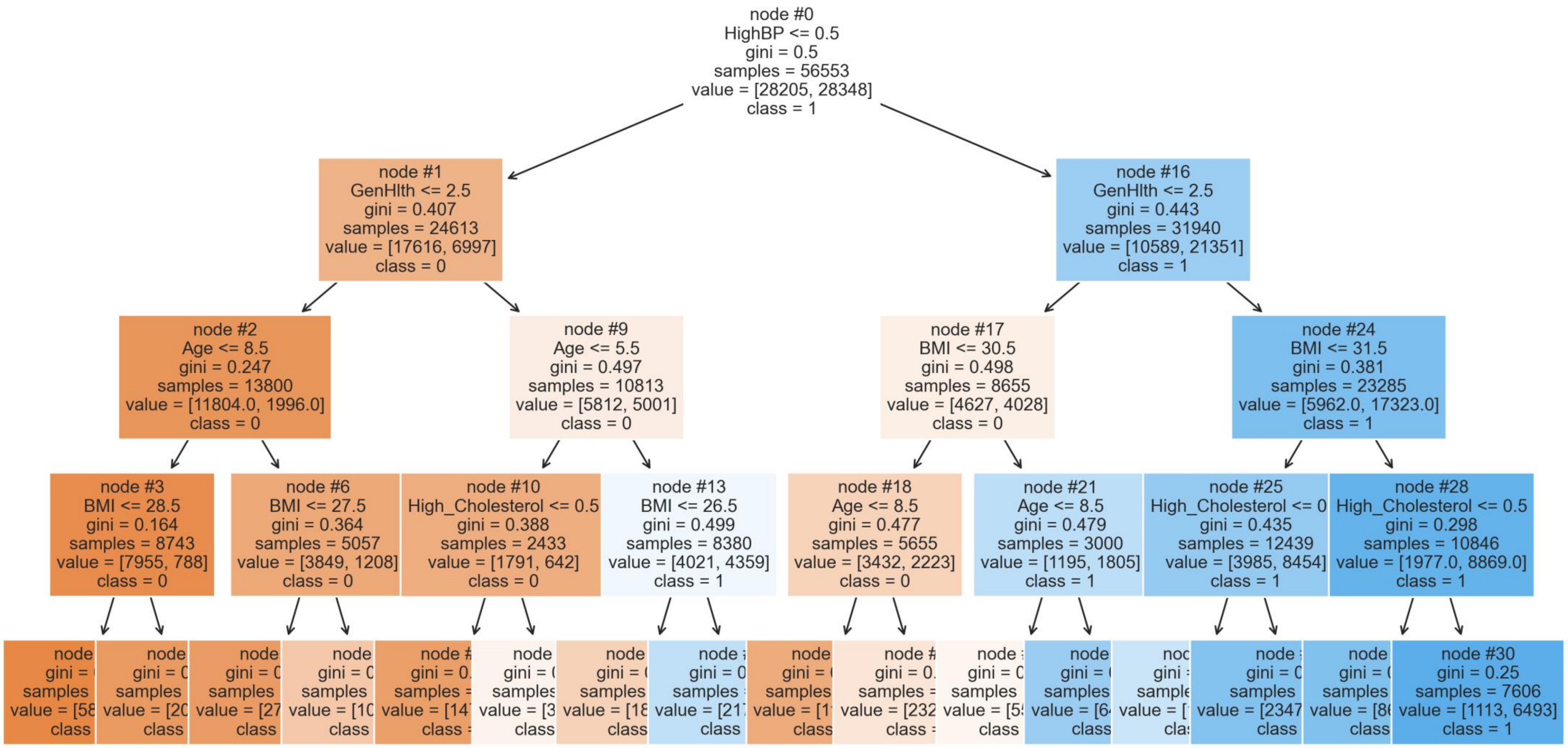
Accuracy of Good Customer Capture by Model is 70% (Specificity)

Accuracy = 73%

Hence model is a good fit on testing data.

	precision	recall	f1-score	support
0	0.75	0.70	0.72	7141
1	0.71	0.76	0.74	6998
accuracy			0.73	14139
macro avg	0.73	0.73	0.73	14139
weighted avg	0.73	0.73	0.73	14139

Plotting of tree



Random Forest

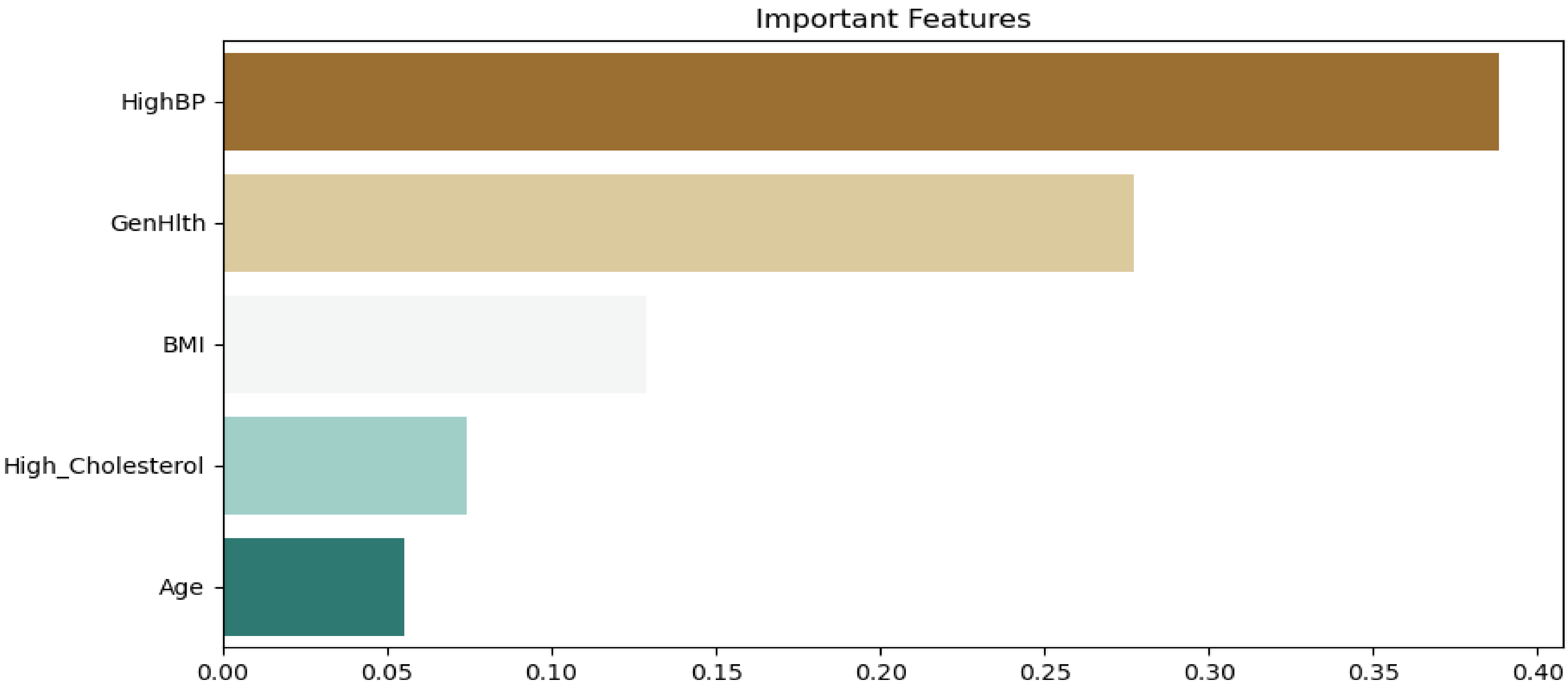
- Model

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier

Model_rf = RandomForestClassifier(random_state=20,
                                  n_estimators=25, # make 25 tress
                                  criterion="gini",
                                  max_depth=4,  # each tree will have 4 branches
                                  min_samples_split=100,# each tree will have parent node
                                  min_samples_leaf=50,# each tree will have Child node
                                  max_features="sqrt")# n_estimators means number tree we want

Model_rf.fit(X_train, y_train)
```

Feature Importance



- **Top 5 features are**

1. HighBP
2. GenHlth
3. BMI
4. High_cholesterol
5. Age

Classification report

- **Training data**

Accuracy of Bad Customer Capture by Model is 79% (Sensitivity)

Accuracy of Good Customer Capture by Model is 68% (Specificity)

Accuracy = 74%

Hence model is a good fit on training data.

	precision	recall	f1-score	support
0	0.76	0.68	0.72	28205
1	0.71	0.79	0.75	28348
accuracy			0.74	56553
macro avg	0.74	0.74	0.74	56553
weighted avg	0.74	0.74	0.74	56553

- **Testing data**

Accuracy of Bad Customer Capture by Model is 79% (Sensitivity)

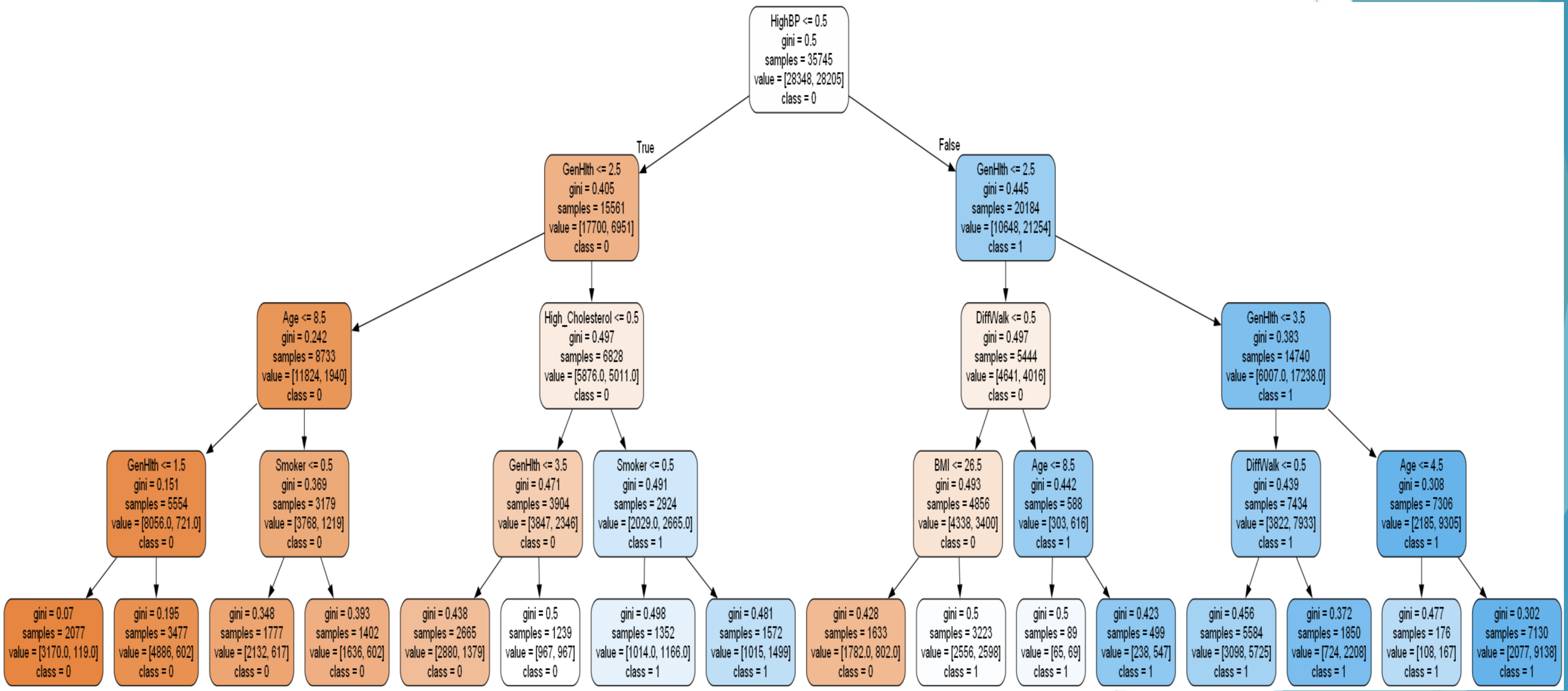
Accuracy of Good Customer Capture by Model is 69% (Specificity)

Accuracy = 74%

Hence model is not a good fit on testing data.

	precision	recall	f1-score	support
0	0.77	0.69	0.73	7141
1	0.72	0.79	0.75	6998
accuracy			0.74	14139
macro avg	0.74	0.74	0.74	14139
weighted avg	0.74	0.74	0.74	14139

Plotting Random Forest



Comparative Analysis

Logistic Regression

Accuracy = 75%

(74.5314786689299)

Sensitivity = 77%

Specificity = 72%

Decision Tree

Accuracy = 73%

(73.01082113303629)

Sensitivity = 76%

Specificity = 70%

Random Forest

Accuracy = 74%

(74.000990169036)

Sensitivity = 79%

Specificity = 69%

Conclusion

Based on the provided accuracy metrics, the Logistic Regression model achieves the highest accuracy among the three models, with an accuracy of 75%. Therefore, the conclusion drawn from these results is that the Logistic Regression model performs the best in terms of overall accuracy compared to the Decision Tree and Random Forest models.Hence Logistic Regression model best fits the data.

THANK YOU !!