Title:

PIMA-INDIANS DIABETES PREDICTION MODEL

Name-Ganesh N Anil
Email-ganesh.n.anil007@gmail.com
Ph-7034007173

Predicting the onset of diabetes.

Abstract-The diabetes dataset is a binary classification problem where it needs to be analysed whether a patient is suffering from the disease or not on the basis of many available features in the dataset. Different methods and procedures of cleaning the data, feature extraction, feature engineering and algorithms to predict the onset of diabetes are used based for diagnostic measure on Pima Indians Diabetes Dataset.

Dataset details:

The data sets comprise several variables of the medical predictor, and one objective variable. The forecasting variables include the patient's number of pregnancies, BMI levels, insulin levels, age, etc.

1: Pregnancies: Number of times pregnant

2: Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test.

3: Blood Pressure: Diastolic blood pressure (mm Hg)

4: Skin Thickness: Triceps skinfold thickness (mm)

5: Insulin: 2-Hour serum insulin (mu U/ml)

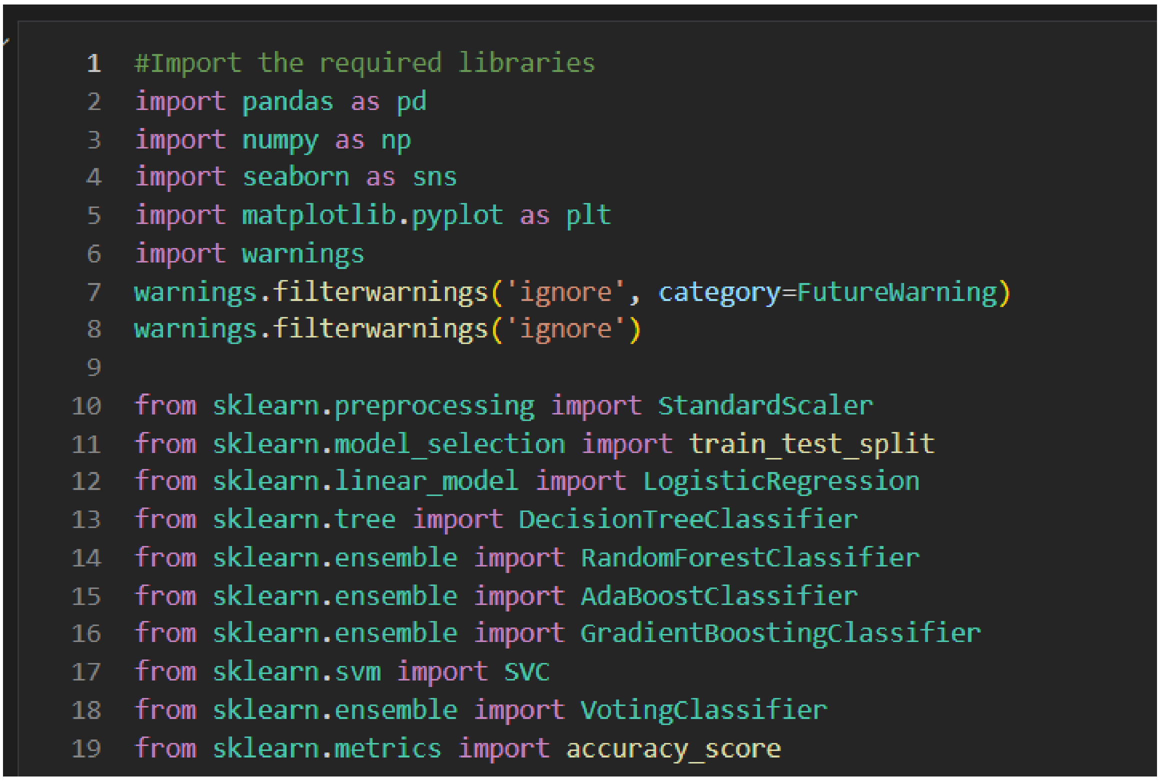
6: BMI: Body mass index (weight in kg/ (height in m^2))

7: Diabetes Pedigree Function: Diabetes pedigree function

8: Age: Age (years)

9: Outcome: Class variable (0 or 1) 268 of 768 are 1, the others are 0

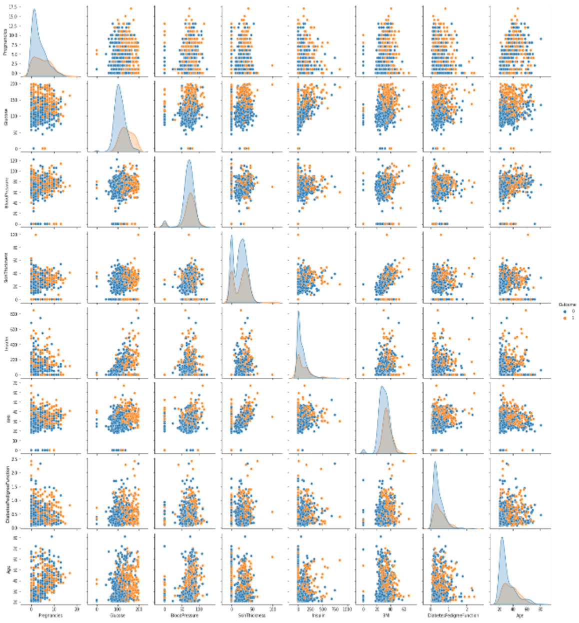
Importing the required libraries:

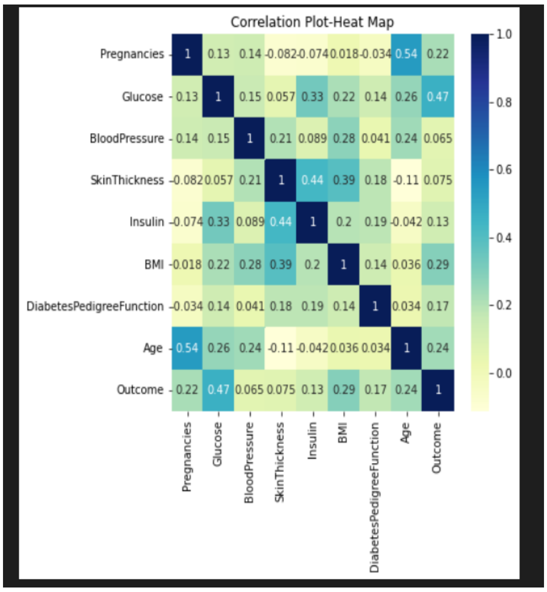


```python
#Import the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score
```

EDA (Exploratory Data Analysis):

- Checking the unique values in target variable and plotting the target variable.
- Value count of output variable was found out and outcome is dataset is imbalance.
- Checking the basic info of the dataset such as shape, size, info etc.
- Finding out the missing values in each of the features.
- Plotting the heatmap and pairplot.

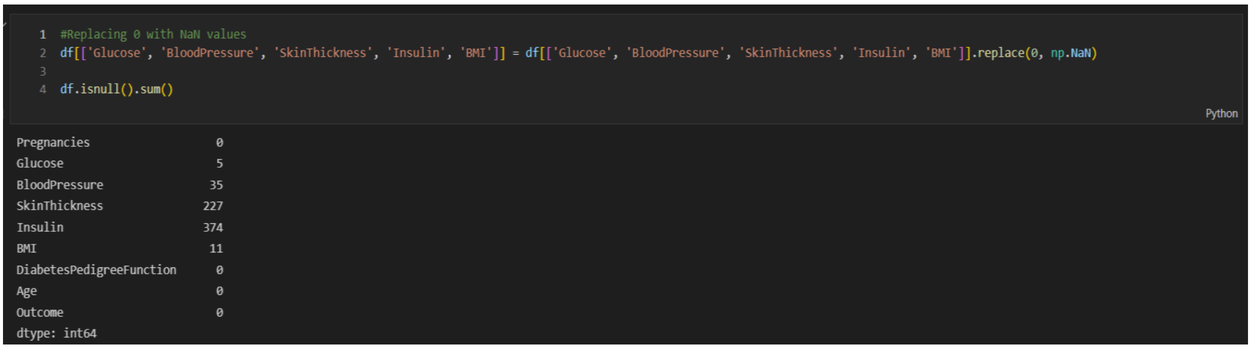




	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.14	-0.082	-0.074	0.018	-0.034	0.54	0.22
Glucose	0.13	1	0.15	0.057	0.33	0.22	0.14	0.26	0.47
BloodPressure	0.14	0.15	1	0.21	0.089	0.28	0.041	0.24	0.065
SkinThickness	-0.082	0.057	0.21	1	0.44	0.39	0.18	-0.11	0.075
Insulin	-0.074	0.33	0.089	0.44	1	0.2	0.19	-0.042	0.13
BMI	-0.018	0.22	0.28	0.39	0.2	1	0.14	0.036	0.29
DiabetesPedigreeFunction	-0.034	0.14	0.041	0.18	0.19	0.14	1	0.034	0.17
Age	0.54	0.26	0.24	-0.11	-0.042	0.036	0.034	1	0.24
Outcome	0.22	0.47	0.065	0.075	0.13	0.29	0.17	0.24	1

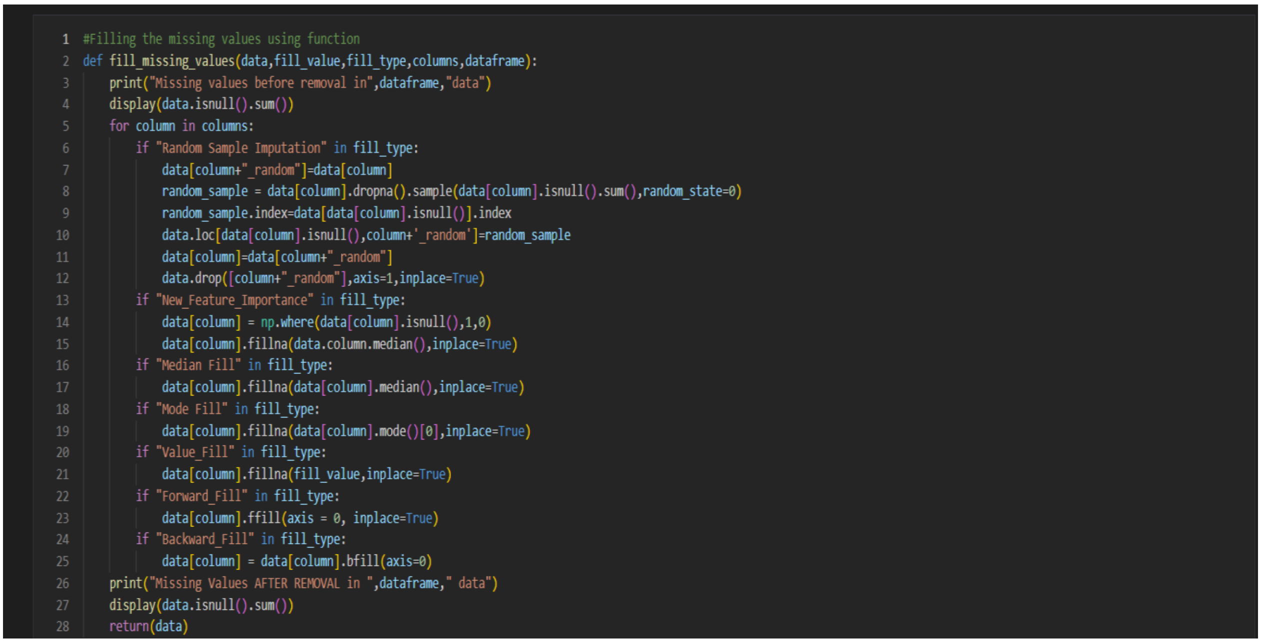
Feature Engineering:

- Handling the missing values using the function.
- Handling the outlier treatment using standard deviation method.

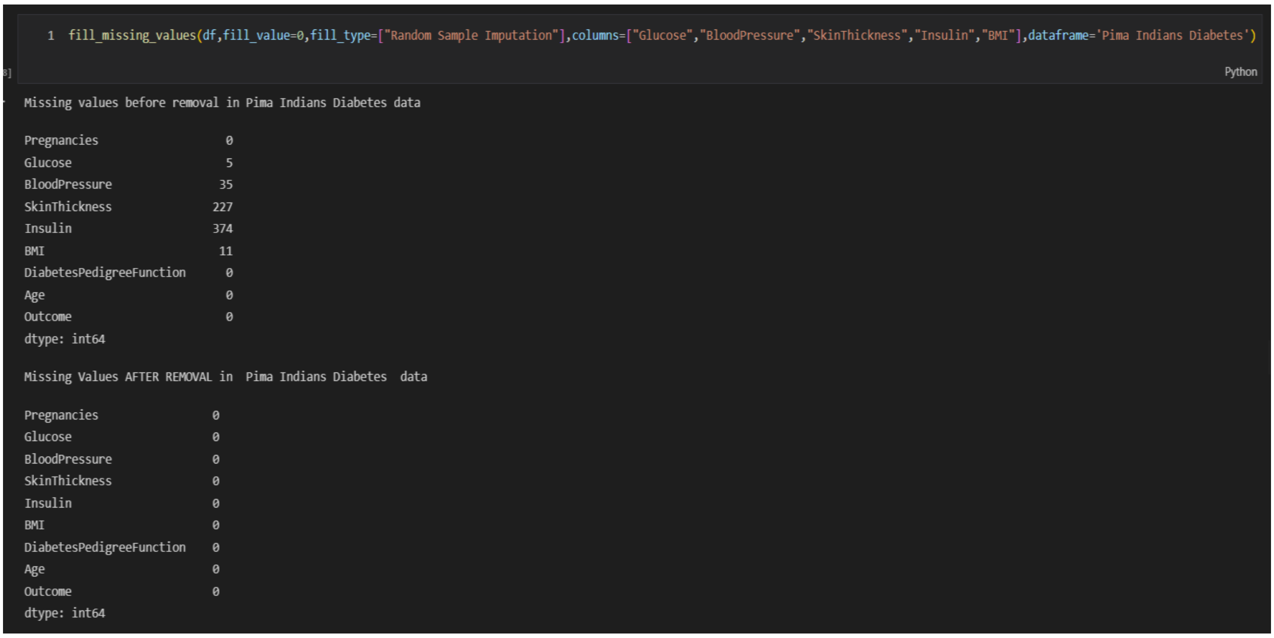


```python
#Replacing 0 with NaN values
df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] = df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']].replace(0, np.NaN)

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

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```



```python
#Filling the missing values using function
def fill_missing_values(data,fill_value,fill_type,columns,dataframe):
    print("Missing values before removal in",dataframe,"data")
    display(data.isnull().sum())
    for column in columns:
        if "Random Sample Imputation" in fill_type:
            data[column+"_random"]=data[column]
            random_sample = data[column].dropna().sample(data[column].isnull().sum(),random_state=0)
            random_sample.index=data[data[column].isnull()].index
            data.loc[data[column].isnull(),column+'_random']=random_sample
            data[column]=data[column+"_random"]
            data.drop([column+"_random"],axis=1,inplace=True)
        if "New_Feature_Importance" in fill_type:
            data[column] = np.where(data[column].isnull(),1,0)
            data[column].fillna(data.column.median(),inplace=True)
        if "Median Fill" in fill_type:
            data[column].fillna(data[column].median(),inplace=True)
        if "Mode Fill" in fill_type:
            data[column].fillna(data[column].mode()[0],inplace=True)
        if "Value_Fill" in fill_type:
            data[column].fillna(fill_value,inplace=True)
        if "Forward_Fill" in fill_type:
            data[column].ffill(axis = 0, inplace=True)
        if "Backward_Fill" in fill_type:
            data[column] = data[column].bfill(axis=0)
    print("Missing Values AFTER REMOVAL in ",dataframe," data")
    display(data.isnull().sum())
    return(data)
```

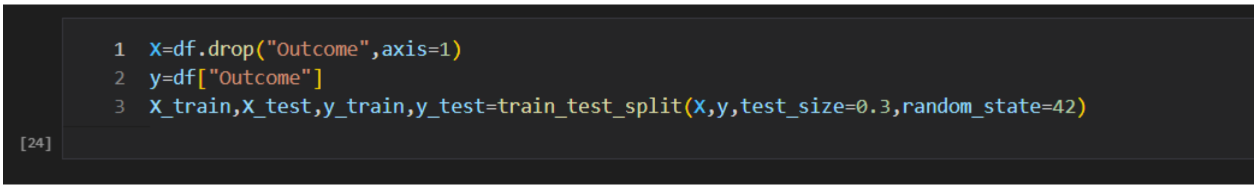


```python
fill_missing_values(df,fill_value=0,fill_type=["Random Sample Imputation"],columns=["Glucose","BloodPressure","SkinThickness","Insulin","BMI"],dataframe='Pima Indians Diabetes')
```

```
Missing values before removal in Pima Indians Diabetes data

Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64

Missing Values AFTER REMOVAL in  Pima Indians Diabetes  data

Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Train Test Split:

- Splitting into X_train, X_test, y_train, y_test.



```python
X=df.drop("Outcome",axis=1)
y=df["Outcome"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
```

Scaling:

- Scale down using Standardscaler function.
- Train using fit_transform and test using transform.
- Created pickle file using joblib.

Data Modelling.

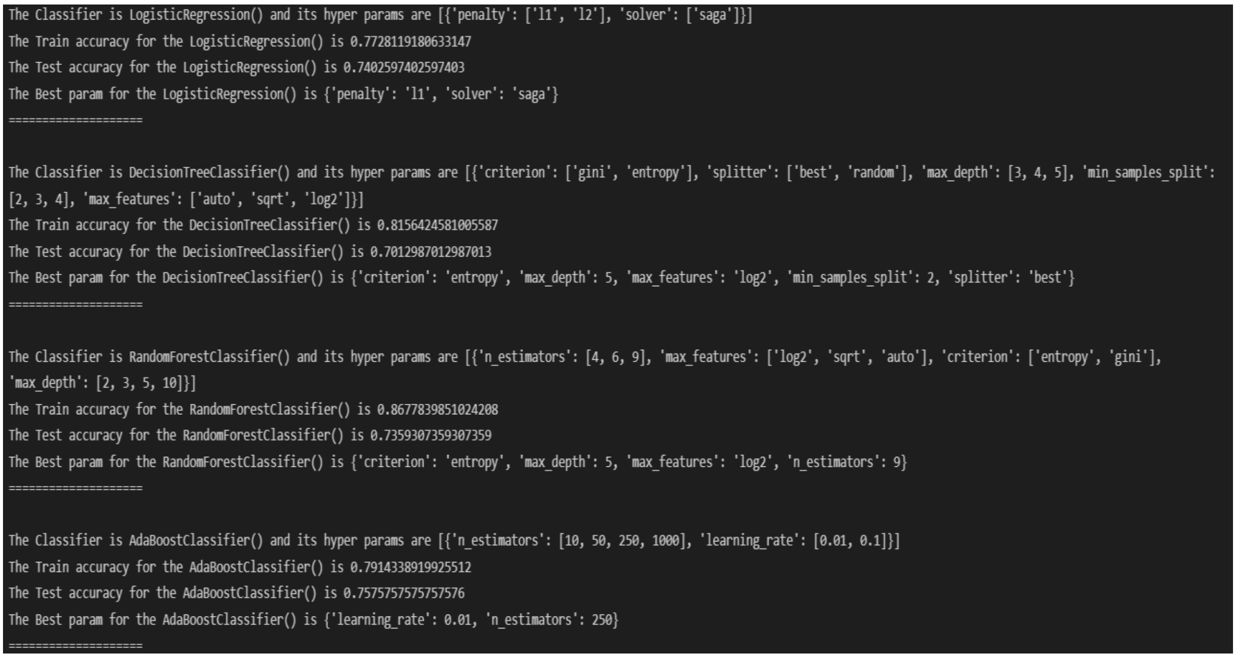
Applying All Algorithms:

- Applying all classification algorithms such as logistic regression, decision tree classifier, random forest classifier, adaboost classifier, gradient boosting classifier and svc classifier.

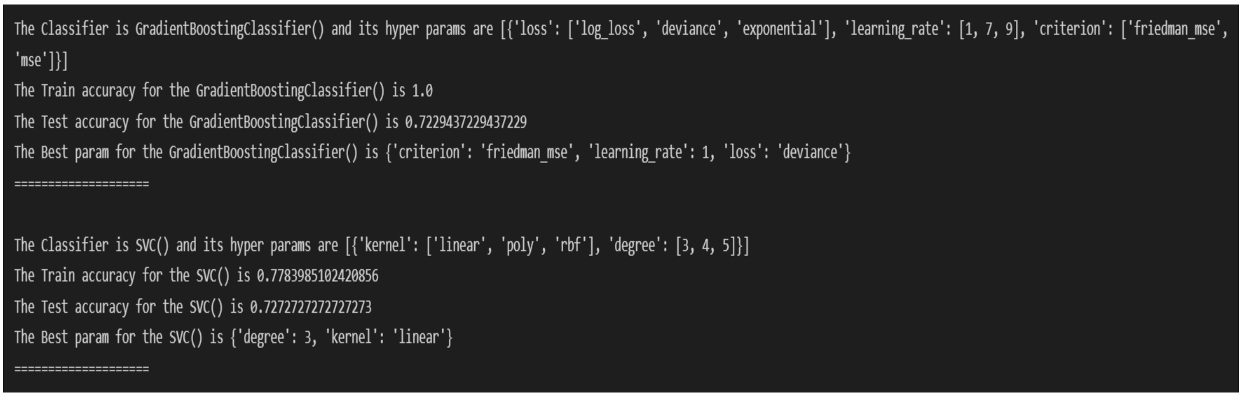
Hyperparameter Tuning:

- Train accuracy and test accuracy for each classifier are found out using hyperparameter tuning.
- Best parameter for each classifier is found out.
- The best model was selected by comparing the train and test accuracy.

- In my case, I selected Random Forest Classifier as it is the best generalised model.



```
The Classifier is LogisticRegression() and its hyper params are [{'penalty': ['l1', 'l2'], 'solver': ['saga']}]
The Train accuracy for the LogisticRegression() is 0.7728119180633147
The Test accuracy for the LogisticRegression() is 0.7402597402597403
The Best param for the LogisticRegression() is {'penalty': 'l1', 'solver': 'saga'}
===================

The Classifier is DecisionTreeClassifier() and its hyper params are [{'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': [3, 4, 5], 'min_samples_split':
[2, 3, 4], 'max_features': ['auto', 'sqrt', 'log2']}]
The Train accuracy for the DecisionTreeClassifier() is 0.8156424581005587
The Test accuracy for the DecisionTreeClassifier() is 0.7012987012987013
The Best param for the DecisionTreeClassifier() is {'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_split': 2, 'splitter': 'best'}
===================

The Classifier is RandomForestClassifier() and its hyper params are [{'n_estimators': [4, 6, 9], 'max_features': ['log2', 'sqrt', 'auto'], 'criterion': ['entropy', 'gini'],
'max_depth': [2, 3, 5, 10]}]
The Train accuracy for the RandomForestClassifier() is 0.8677839851024208
The Test accuracy for the RandomForestClassifier() is 0.7359307359307359
The Best param for the RandomForestClassifier() is {'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'n_estimators': 9}
===================

The Classifier is AdaBoostClassifier() and its hyper params are [{'n_estimators': [10, 50, 250, 1000], 'learning_rate': [0.01, 0.1]}]
The Train accuracy for the AdaBoostClassifier() is 0.7914338919925512
The Test accuracy for the AdaBoostClassifier() is 0.7575757575757576
The Best param for the AdaBoostClassifier() is {'learning_rate': 0.01, 'n_estimators': 250}
===================
```

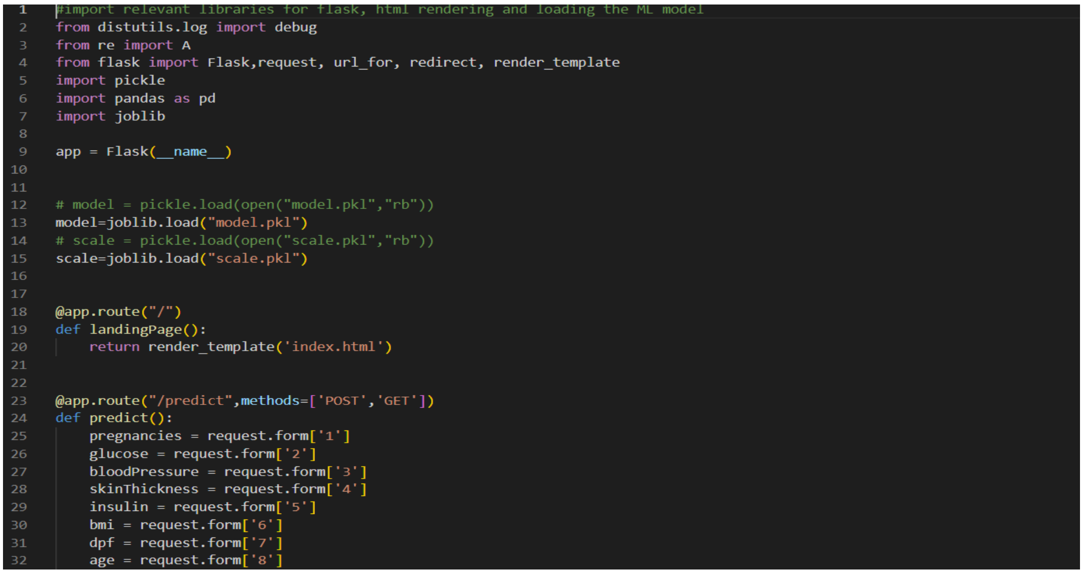


```
The Classifier is GradientBoostingClassifier() and its hyper params are [{'loss': ['log_loss', 'deviance', 'exponential'], 'learning_rate': [1, 7, 9], 'criterion': ['friedman_mse',
'mse']}]
The Train accuracy for the GradientBoostingClassifier() is 1.0
The Test accuracy for the GradientBoostingClassifier() is 0.7229437229437229
The Best param for the GradientBoostingClassifier() is {'criterion': 'friedman_mse', 'learning_rate': 1, 'loss': 'deviance'}
===================

The Classifier is SVC() and its hyper params are [{'kernel': ['linear', 'poly', 'rbf'], 'degree': [3, 4, 5]}]
The Train accuracy for the SVC() is 0.7783985102420856
The Test accuracy for the SVC() is 0.7272727272727273
The Best param for the SVC() is {'degree': 3, 'kernel': 'linear'}
===================
```

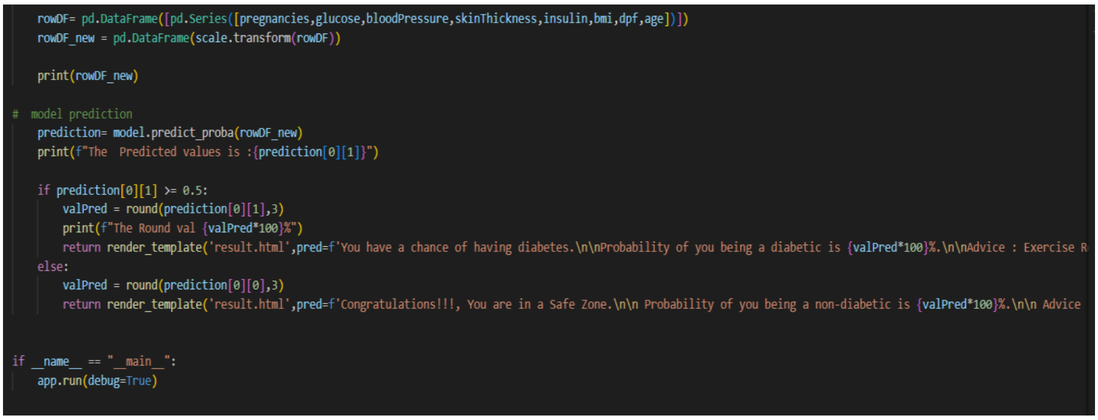
- Created pickle file using joblib.

Web Framework.

- Using the flask backend api was set
- UI (User Interface) was set using html code.

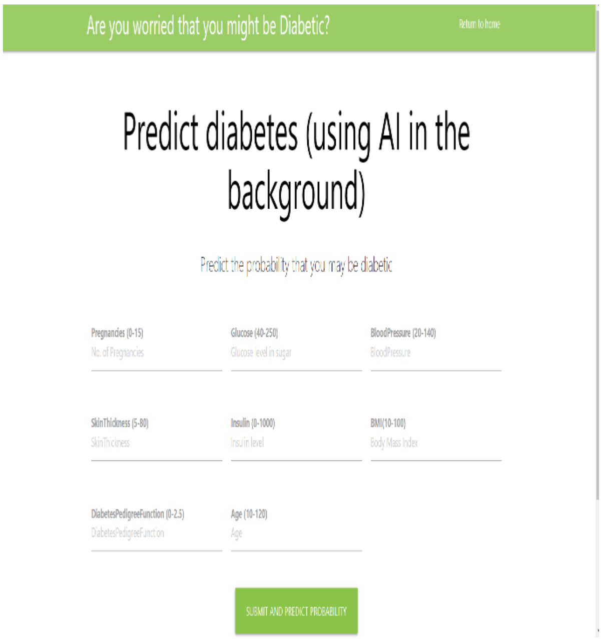


```python
#import relevant libraries for flask, html rendering and loading the ML model
from distutils.log import debug
from re import A
from flask import Flask,request, url_for, redirect, render_template
import pickle
import pandas as pd
import joblib

app = Flask(__name__)


# model = pickle.load(open("model.pkl","rb"))
model=joblib.load("model.pkl")
# scale = pickle.load(open("scale.pkl","rb"))
scale=joblib.load("scale.pkl")


@app.route("/")
def landingPage():
    return render_template('index.html')


@app.route("/predict",methods=['POST','GET'])
def predict():
    pregnancies = request.form['1']
    glucose = request.form['2']
    bloodPressure = request.form['3']
    skinThickness = request.form['4']
    insulin = request.form['5']
    bmi = request.form['6']
    dpf = request.form['7']
    age = request.form['8']
```



```python
    rowDF= pd.DataFrame([pd.Series([pregnancies,glucose,bloodPressure,skinThickness,insulin,bmi,dpf,age])])
    rowDF_new = pd.DataFrame(scale.transform(rowDF))

    print(rowDF_new)

#  model prediction
    prediction= model.predict_proba(rowDF_new)
    print(f"The  Predicted values is :{prediction[0][1]}")

    if prediction[0][1] >= 0.5:
        valPred = round(prediction[0][1],3)
        print(f"The Round val {valPred*100}%")
        return render_template('result.html',pred=f'You have a chance of having diabetes.\n\nProbability of you being a diabetic is {valPred*100}%.\n\nAdvice : Exercise R
    else:
        valPred = round(prediction[0][0],3)
        return render_template('result.html',pred=f'Congratulations!!!, You are in a Safe Zone.\n\n Probability of you being a non-diabetic is {valPred*100}%.\n\n Advice


if __name__ == "__main__":
    app.run(debug=True)
```

User Interface.



Predict diabetes (using AI in the background)

Predict the probability that you may be diabetic

Pregnancies (0-15)
No. of Pregnancies

Glucose (40-250)
Glucose level in sugar

BloodPressure (20-140)
BloodPressure

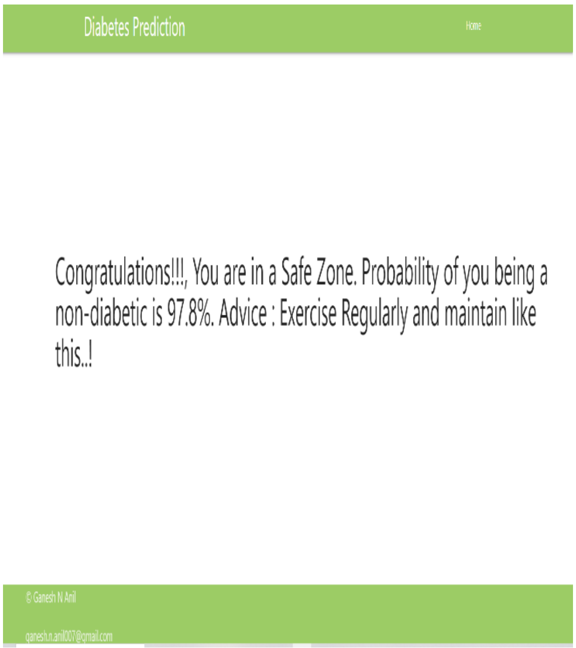
SkinThickness (5-80)
SkinThickness

Insulin (0-1000)
Insulin level

BMI(10-100)
Body Mass Index

DiabetesPedigreeFunction (0-2.5)
DiabetesPedigreeFunction

Age (10-120)
Age



Congratulations!!!, You are in a Safe Zone. Probability of you being a non-diabetic is 97.8%. Advice : Exercise Regularly and maintain like this..!

© Ganesh N Anil

ganesh.n.anil007@gmail.com

Model Deployment.

- I Have done the project in Google Collab and later pushed the file into my GitHub repository (https://github.com/GaneshNAnil/Pima_Indians_Diabetes.git).
- I Have deployed the project through Heroku (https://diabetispred.herokuapp.com/)