DIABETES PREDICTION SYSTEM

ARTIFICAL INTELLIGENCE

ABSTRACT:

- Globally, diabetes affects 537 million people, making it the deadliest and the most common non-communicable disease. Many factors can cause a person to get affected by diabetes, like excessive body weight, abnormal cholesterol level, family history, physical inactivity, bad food habit etc. Increased urination is one of the most common symptoms of this disease. People with diabetes for a long time can get several complications like heart disorder, kidney disease, nerve damage, diabetic retinopathy etc. But its risk can be reduced if it is predicted early.
- In this project, an automatic diabetes prediction system has been developed using a dataset and various machine learning techniques. A semi-supervised model with extreme gradient boosting has been utilized to predict the insulin features of the dataset. SMOTE and ADASYN approaches have been employed to manage the class imbalance problem. We use machine learning classification methods, i.e., decision tree, SVM, Random Forest, Logistic Regression, KNN, and various ensemble techniques, to determine which algorithm produces the best prediction results. Furthermore, the domain adaptation method has been implemented to demonstrate the versatility of the proposed system. The explainable AI approach with LIME and SHAP frameworks is implemented to understand how the model predicts the final results.

INTRODUCTION:

- Diabetes is a chronic disease that directly affects the pancreas, and the body is incapable of producing insulin . Insulin is mainly responsible for maintaining the blood glucose level. Many factors, such as excessive body weight, physical inactivity, high blood pressure, and abnormal cholesterol level, can cause a person get affected by diabetes . It can cause many complications, but an increase in urination is one of the most common ones . It can damage the skin, nerves, and eyes, and if not treated early, diabetes can cause kidney failure and diabetic retinopathy ocular disease. According to IDF (International Diabetes Federation) statistics, 537 million people had diabetes around the world in 2021.
- Early and accurate diagnosis of diabetes mellitus, especially during its initial development, is challenging for medical professionals. Artificial intelligence and machine learning techniques, providing a reference, can help them gain preliminary knowledge about this disease and reduce their workload accordingly.
- For predicting blood pressure status, they used conditional decision making and for predicting diabetes, they used SVM, KNN, and decision tree. Among these models, SVM worked better as they got 75% accuracy which is better than other classifier algorithms.

PROBLEM STATEMENT:

- Develop an AI-powered diabetes prediction system that leverages machine learning algorithms to analyze medical data and predict the likelihood of an individual developing diabetes, providing early risk assessment and personalized preventive measures.

PROBLEM DEFINITION:

- The problem is to build an AI-powered diabetes prediction system that uses machine learning algorithms to analyze medical data and predict the likelihood of an individual developing diabetes. The system aims to provide early risk assessment and personalized preventive measures, allowing individuals to take proactive actions to manage their health.

DESING THINKING:

To address this problem, we will follow a systematic approach involving the following key steps:

1. Data Collection
2. Data Preprocessing
3. Feature Selection
4. Model Selection
5. Evaluation
6. Iterative Improvement

DATA COLLECTION:

- In this stage, we use the data from the given data set provided through the Kaggle software.
- Ensure the dataset is representative of the real-world scenarios of diabetes patients.
- The data set provided makes it easier for data preprocessing.

DATA PREPROCESSING:

Clean the data to prepare it for predicting. This involves several steps:

- **Handling Missing Data**: Deal with missing values, if any, through imputation or removal.
- **Removing Duplicates:** Eliminate the duplicate data to prevent error in the prediction.

FEATURE SELECTION:

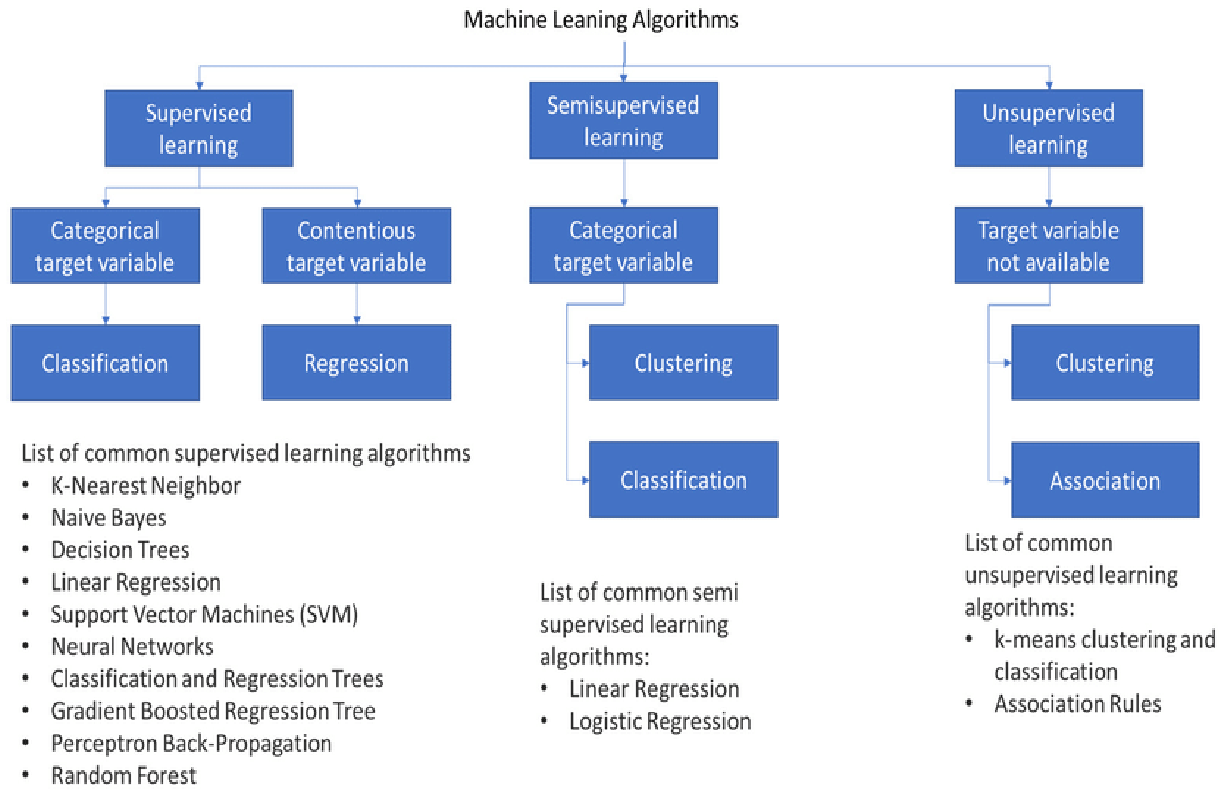
Based on the following features from the given data set, we predict the diabetes positive patients:

1. Glucose level
2. Blood pressure
3. Insulin
4. Age
5. Skin Thickness
6. BMI

7. Diabetes Pedigree Function

8. Pregnancies

MODEL SELECTION:


Machine Leaning Algorithms
Supervised learning
Semisupervised learning
Unsupervised learning
Categorical target variable
Contentious target variable
Categorical target variable
Target variable not available
Classification
Regression
Clustering
Classification
Clustering
Association
List of common supervised learning algorithms
• K-Nearest Neighbor
• Naive Bayes
• Decision Trees
• Linear Regression
• Support Vector Machines (SVM)
• Neural Networks
• Classification and Regression Trees
• Gradient Boosted Regression Tree
• Perceptron Back-Propagation
• Random Forest
List of common semi supervised learning algorithms:
• Linear Regression
• Logistic Regression
List of common unsupervised learning algorithms:
• k-means clustering and classification
• Association Rules

EVALUTION:

Performance Metrics: We will evaluate the model's performance using various metrics, including:

1. Accuracy: To measure the overall correctness of the classification.

2. Precision: To assess the model's ability to correctly classify the given data set.

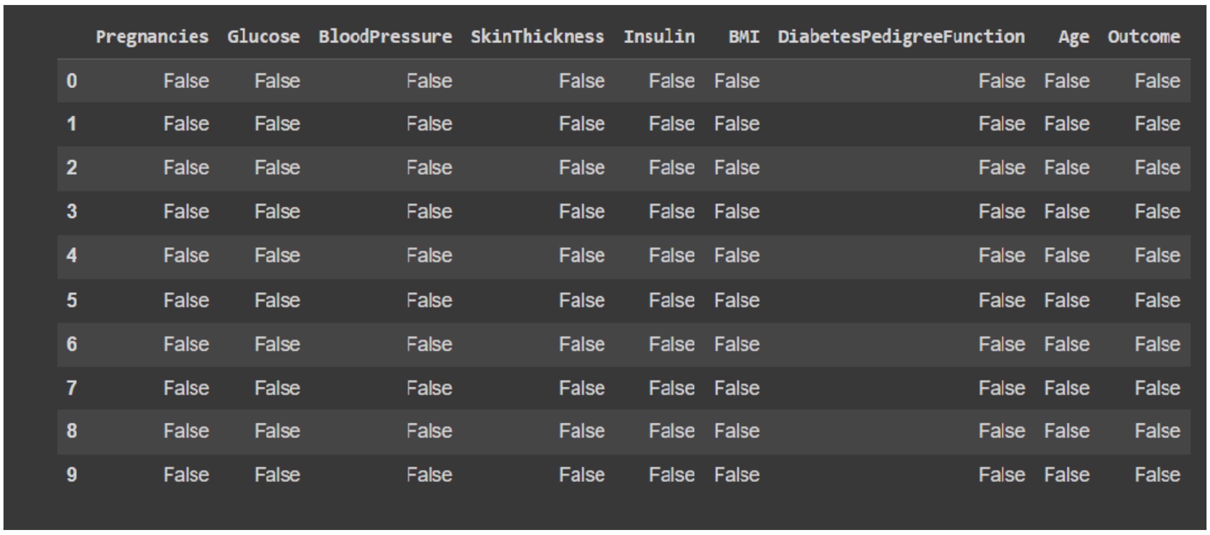
3. Recall: To measure the model's capacity to correctly identify the given data set.

ITERATIVE IMPROVEMENT:

The improve the accuracy of the diabetes prediction using feedback loops, Ethical Considerations, Deployment and Monitoring, Interpretability and Explainability, Regularization, Ensemble Methods, Iterate and Refine, Performance Metrics, Validation and Cross-Validation, Hyperparameter Tuning, Model Selection, Feature Engineering, Data Collection and Preprocessing.

Dataset: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

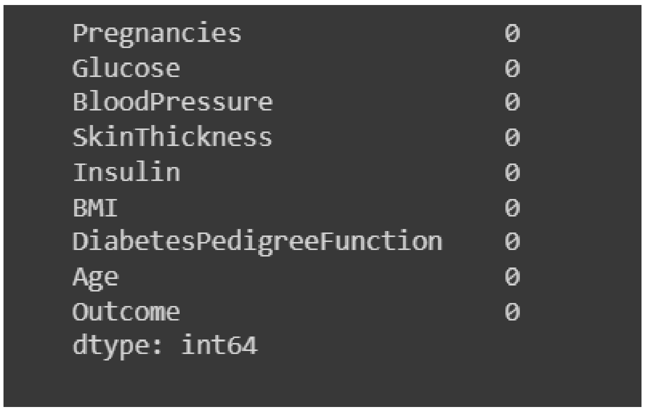
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

from mlxtend.plotting import plot_decision_regions

import missingno as msno

from pandas.plotting import scatter_matrix

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix

from sklearn import metrics

from sklearn.metrics import classification_report

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

diabetes_df = pd.read_csv('./sample_data/diabetes.csv')


# Now let's check that if our dataset have null values or not

 diabetes_df.isnull().head(10)
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False

Now let's check that if our dataset have null values or not

diabetes_df.isnull().sum()



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

Here from above code we first checked that is there any null values from isnull() function then we are going to take the sum of all those missing values from sum() function and the inference we now get is that there are no missing values but that is actually not a true story as in this particular dataset all the missing values were given the 0 as value which is not good for the authenticity of the dataset. Hence we will first replace the 0 value to NAN value then start the imputation process.

3 Machine learning algorithm to proceed the prediction analysis :

- Random Forest
- Decision Tree
- Support Vector Machine (SVM)

ALGORITHM SELECTION :

The algorithm selected are :

1.Random Forest

2. Decision Tree

3.Support Vector Machine (SVM)

RANDOM FOREST:

Random Forest is a machine learning ensemble algorithm that combines multiple decision trees to make more accurate predictions. It works by constructing a forest of decision trees during training and averaging their predictions for better overall results. It's known for its robustness, versatility, and ability to handle both classification and regression tasks.

DECISION TREE:

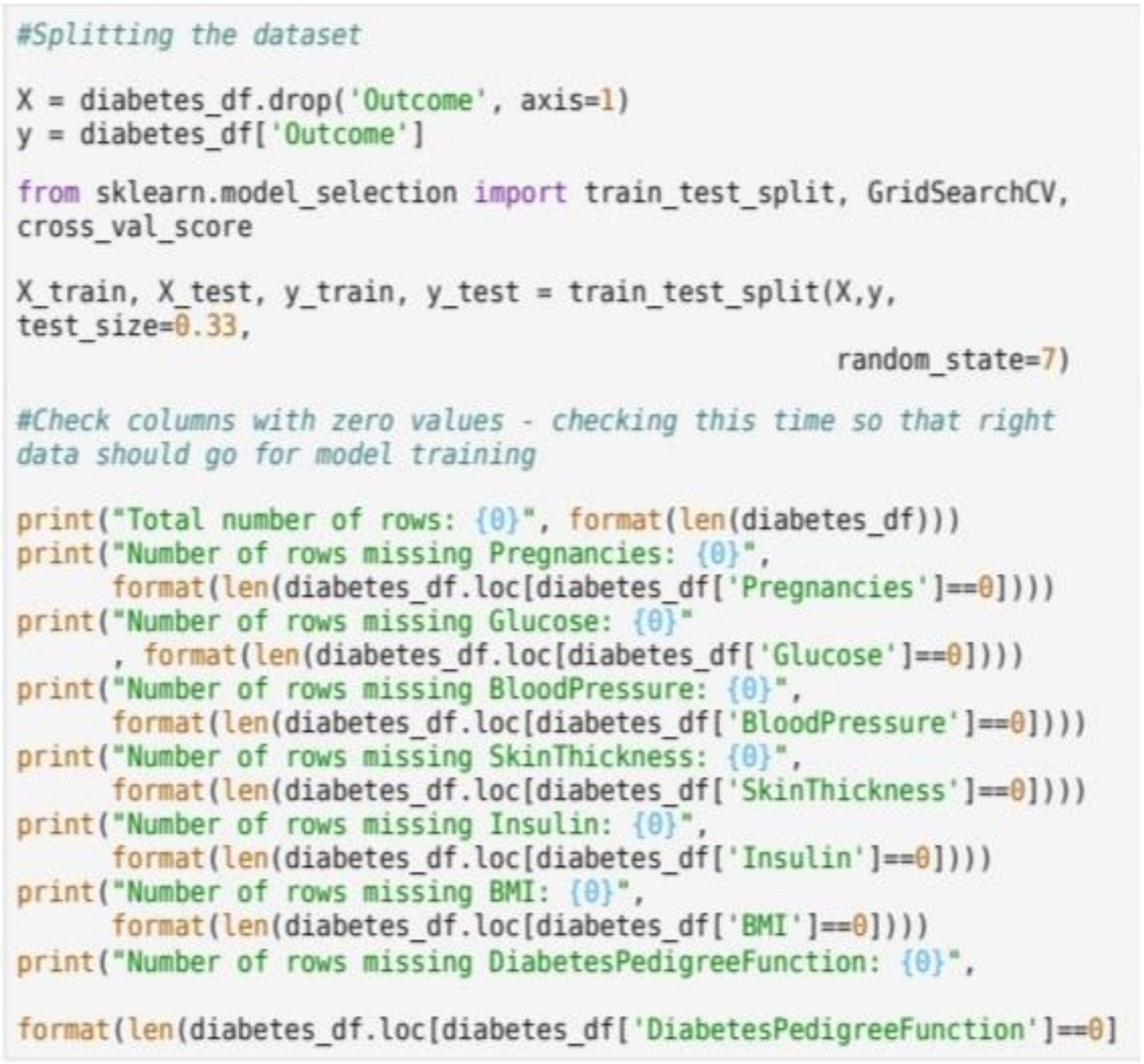
Decision tree is a popular machine learning algorithm used for classification and regression tasks. It models decisions and their possible consequences in a tree-like structure. At each node of the tree, a decision is made based on a feature, leading to one of several possible outcomes. This process continues until a final decision or prediction is reached. Decision trees are interpretable, easy to understand, and can handle both categorical and numerical data. Popular variations include Rand

SUPPORT VECTOR MACHINE (SVM):

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression. It works by finding a hyperplane that best separates different classes in the input data while maximizing the margin between them. The data points closest to this hyperplane are called support vectors. SVMs can handle high-dimensional data and are effective for both linear and non-linear problems through the use of kernel functions. They are particularly useful when dealing with binary classification tasks.

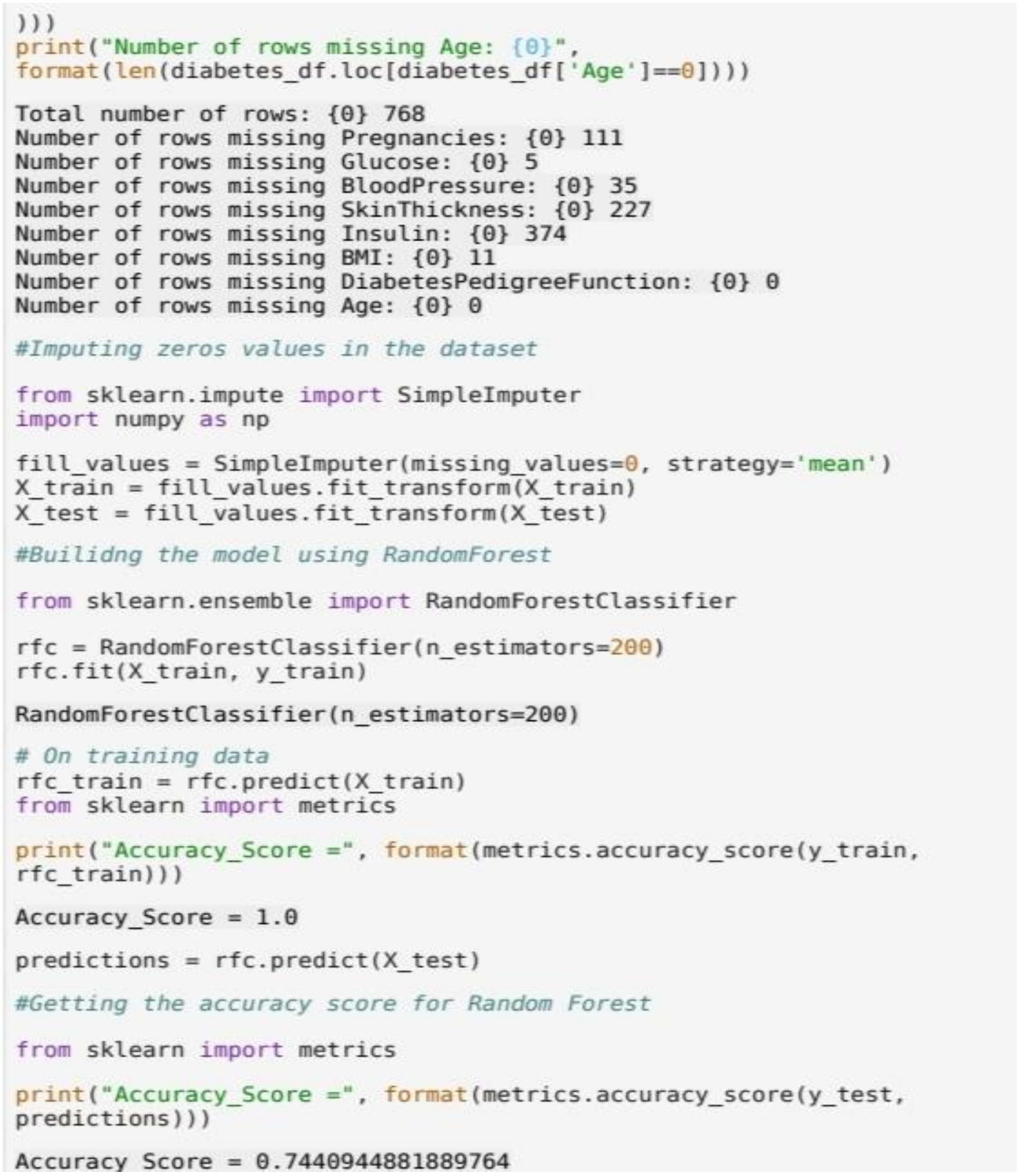
MODEL BUILDING :

Spitting the Dataset:



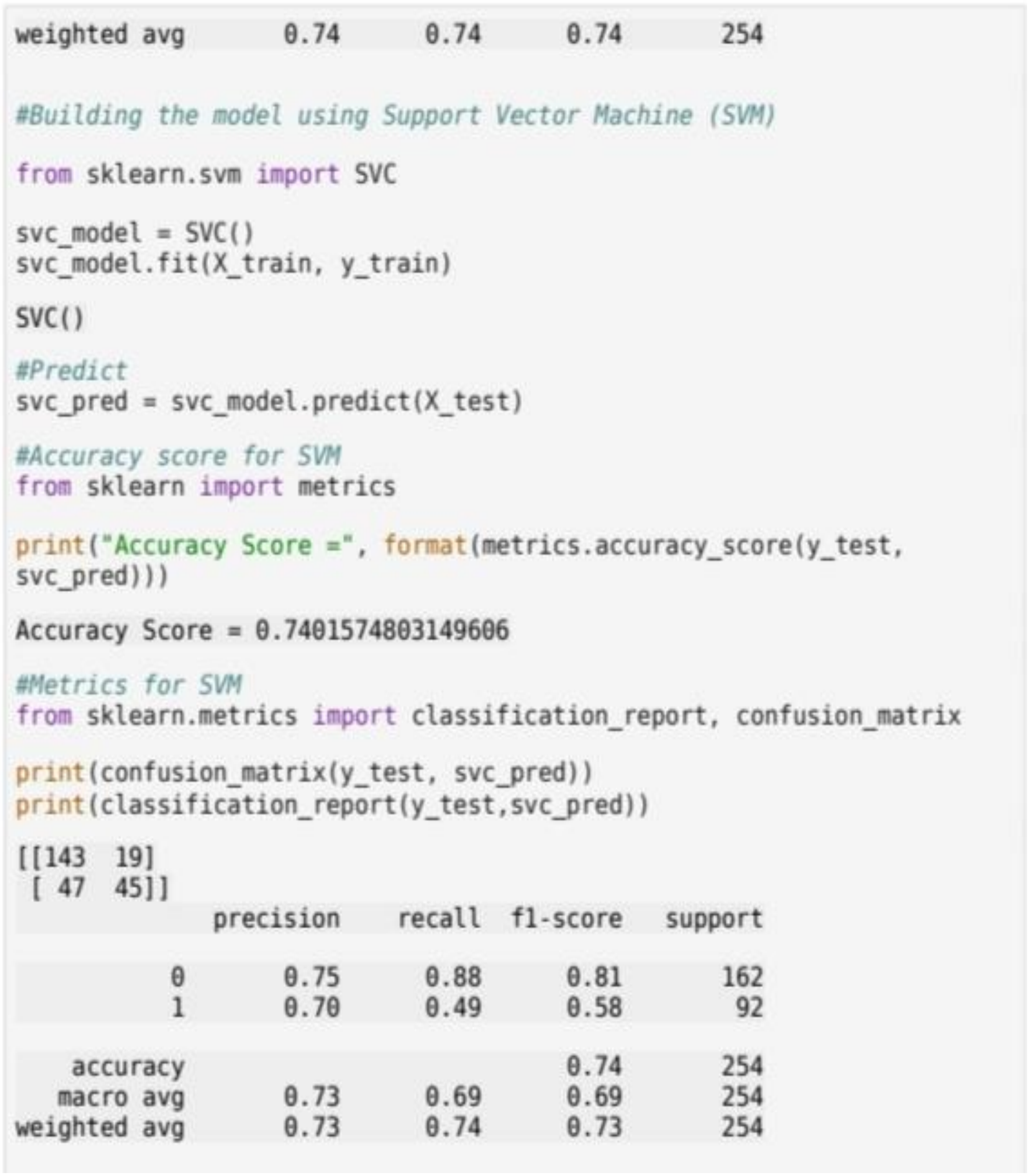
```
#Splitting the dataset

X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']

from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.33,
                                                    random_state=7)

#Check columns with zero values - checking this time so that right
data should go for model training

print("Total number of rows: {0}", format(len(diabetes_df)))
print("Number of rows missing Pregnancies: {0}",
      format(len(diabetes_df.loc[diabetes_df['Pregnancies']==0])))
print("Number of rows missing Glucose: {0}"
      , format(len(diabetes_df.loc[diabetes_df['Glucose']==0])))
print("Number of rows missing BloodPressure: {0}",
      format(len(diabetes_df.loc[diabetes_df['BloodPressure']==0])))
print("Number of rows missing SkinThickness: {0}",
      format(len(diabetes_df.loc[diabetes_df['SkinThickness']==0])))
print("Number of rows missing Insulin: {0}",
      format(len(diabetes_df.loc[diabetes_df['Insulin']==0])))
print("Number of rows missing BMI: {0}",
      format(len(diabetes_df.loc[diabetes_df['BMI']==0])))
print("Number of rows missing DiabetesPedigreeFunction: {0}",

format(len(diabetes_df.loc[diabetes_df['DiabetesPedigreeFunction']==0]
```

RANDOM FOREST:



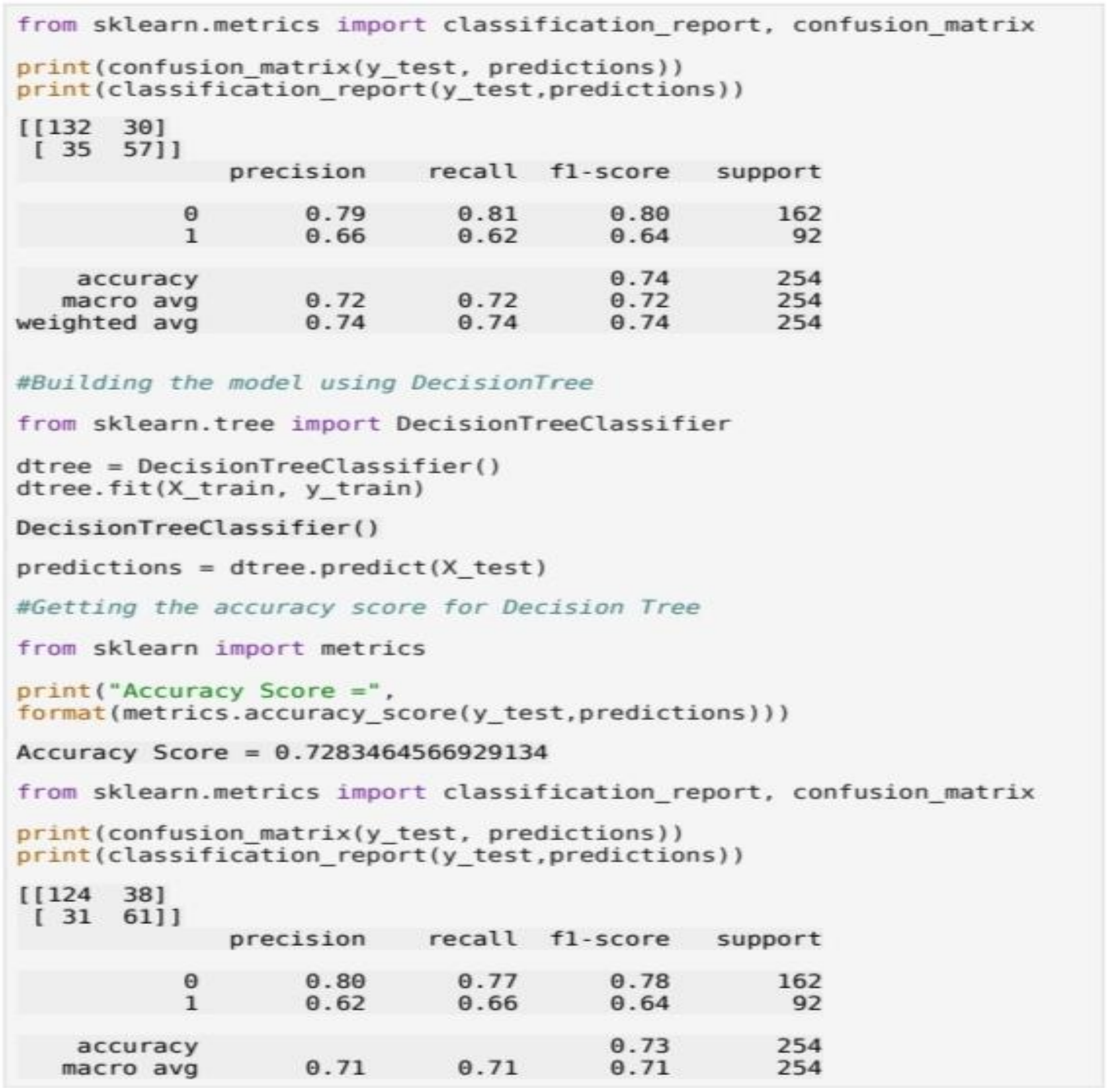
```
)))
print("Number of rows missing Age: {0}",
format(len(diabetes_df.loc[diabetes_df['Age']==0])))

Total number of rows: {0} 768
Number of rows missing Pregnancies: {0} 111
Number of rows missing Glucose: {0} 5
Number of rows missing BloodPressure: {0} 35
Number of rows missing SkinThickness: {0} 227
Number of rows missing Insulin: {0} 374
Number of rows missing BMI: {0} 11
Number of rows missing DiabetesPedigreeFunction: {0} 0
Number of rows missing Age: {0} 0

#Imputing zeros values in the dataset

from sklearn.impute import SimpleImputer
import numpy as np

fill_values = SimpleImputer(missing_values=0, strategy='mean')
X_train = fill_values.fit_transform(X_train)
X_test = fill_values.fit_transform(X_test)

#Builidng the model using RandomForest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)

RandomForestClassifier(n_estimators=200)

# On training data
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train,
rfc_train)))

Accuracy_Score = 1.0

predictions = rfc.predict(X_test)

#Getting the accuracy score for Random Forest

from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_test,
predictions)))

Accuracy Score = 0.7440944881889764
```

DECISION TREE:



```
weighted avg       0.74      0.74      0.74       254

#Building the model using Support Vector Machine (SVM)

from sklearn.svm import SVC

svc_model = SVC()
svc_model.fit(X_train, y_train)

SVC()

#Predict
svc_pred = svc_model.predict(X_test)

#Accuracy score for SVM
from sklearn import metrics

print("Accuracy Score =", format(metrics.accuracy_score(y_test,
svc_pred)))

Accuracy Score = 0.7401574803149606

#Metrics for SVM
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test,svc_pred))

[[143  19]
 [ 47  45]]
              precision    recall  f1-score   support

           0       0.75      0.88      0.81       162
           1       0.70      0.49      0.58        92

    accuracy                           0.74       254
   macro avg       0.73      0.69      0.69       254
weighted avg       0.73      0.74      0.73       254
```

SUPPORT VECTOR MACHINE (SVM):



```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))

[[132  30]
 [ 35  57]]
              precision    recall  f1-score   support

           0       0.79      0.81      0.80       162
           1       0.66      0.62      0.64        92

    accuracy                           0.74       254
   macro avg       0.72      0.72      0.72       254
weighted avg       0.74      0.74      0.74       254


#Building the model using DecisionTree

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

DecisionTreeClassifier()

predictions = dtree.predict(X_test)

#Getting the accuracy score for Decision Tree

from sklearn import metrics

print("Accuracy Score =",
format(metrics.accuracy_score(y_test,predictions)))

Accuracy Score = 0.7283464566929134

from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))

[[124  38]
 [ 31  61]]
              precision    recall  f1-score   support

           0       0.80      0.77      0.78       162
           1       0.62      0.66      0.64        92

    accuracy                           0.73       254
   macro avg       0.71      0.71      0.71       254
```

CONCLUSION:

- Diabetes can be a reason for reducing life expectancy and quality. Predicting this chronic disorder earlier can reduce the risk and complications of many diseases in the long run. In this paper, an automatic diabetes prediction system using various machine learning approaches has been proposed.
- There are some future scopes of this work, for example, we recommend getting additional private data with a larger cohort of patients to get better results. Another extension of this work is combining machine learning models with fuzzy logic techniques and applying optimization approaches.
- Therefore,Random forest is the best model for this prediction since it has an accuracy_score of 0.76