****

**Project Report**

**On**

**Diabetes-Prediction**

**Using**

**Machine-Learning**

|  |  |
| --- | --- |
| **Roll No** | **Name** |
| 17csu074 | Harsh Yadav |
| 17csu063 | Dhruv Saini |

**INTRODUCTION :**

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a [hormone](https://www.niddk.nih.gov/Dictionary/H/hormone) made by the [pancreas](https://www.niddk.nih.gov/Dictionary/P/pancreas), helps glucose from food get into your cells to be used for energy. Sometimes your body doesn’t make enough—or any—insulin or doesn’t use insulin well. Glucose then stays in your blood and doesn’t reach your cells.

Preventing Diabetes diseases has become more than necessary. Good data-driven systems for predicting diabetes diseases can improve the entire research and prevention process, making sure that more people can live healthy lives. This is where Machine Learning comes into play. Machine Learning helps in predicting the Diabetes diseases, and the predictions made are quite accurate.

The project involved analysis of the diabetes disease patient dataset with proper data processing. Then, different models were trained and and predictions are made with different algorithms KNN, Decision Tree, Random Forest , SVM, Logistic Regression etc. This is the SPYDER code and dataset I've used for my project is taken from Kaggle . We have used python inbuilt libraries i.e “Sklearn and Keras'”.

I've used a variety of Machine Learning algorithms, implemented in Python, to predict the presence of diabetes disease in a patient. This is a classification problem, with input features as a variety of parameters, and the target variable as a binary variable, predicting whether diabetes disease is present or not.

Machine Learning algorithms used:

1. Logistic Regression (Scikit-learn)
2. Support Vector Machine (Kernel) (Scikit-learn)
3. K-Nearest Neighbours (Scikit-learn)
4. Decision Tree (Scikit-learn)
5. Random Forest (Scikit-learn)
6. Artificial Neural Network with 1 Hidden layer (Keras)

Accuracy achieved: 100% (Random Forest)

**LITERATURE REVIEW :**

Diabetes Mellitus (DM) is an increasingly prevalent chronic disease characterized by the body’s inability to metabolize glucose. Finding the disease at the early stage helps reduce medical costs and the risk of patients having more complicated health problems. Wilson et al. developed the Framingham Diabetes Risk Scoring Model (FDRSM) to predict the risk for developing DM in middle-aged American adults (45 to 64 years of age) using Logistic Regression. The risk factors considered in this simple clinical model are parental history of DM, obesity, high blood pressure, low levels of high-density lipoprotein cholesterol, elevated triglyceride levels, and impaired fasting glucose. The number of subjects in the sample was 3140 and the area under the receiver operating characteristic curve (AROC) was reported to be 85.0%. The performance of this algorithm was evaluated in a Canadian population by Mashayekhi et al. using the same predictors as Wilson et al. with the exception of parental history of DM. The number of subjects in the sample was 4403 and the reported AROC was 78.6%.

Data mining techniques have been widely used in DM studies to explore the risk factors for DM. Machine learning methods, such as logistic regression, artificial neural network, and decision tree were used by Meng et al.to predict DM and pre-diabetes. The data included 735 patients who had DM or pre-diabetes and 752 who are healthy from Guangzhou, China. The accuracy was reported to be 77.87% using a decision tree model; 76.13% using a logistic regression model; and 73.23% using the Artificial Neural Network (ANN) procedure.

Diabetes is considered as one of the deadliest and chronic diseases which causes an increase in blood sugar. Many complications occur if diabetes remains untreated and unidentified. The tedious identifying process results in visiting of a patient to a diagnostic center and consulting doctor. But the rise in machine learning approaches solves this critical problem. The motive of those study is to design a model which can prognosticate the likelihood of diabetes in patients with maximum accuracy. Therefore three machine learning classification algorithms namely Decision Tree, SVM and Naive Bayes are used in existing experiment to detect diabetes at an early stage. Experiments are performed on Pima Indians Diabetes Database (PIDD) which is sourced from UCI machine learning repository. The performances of all the three algorithms are evaluated on various measures like Precision, Accuracy, F-Measure, and Recall. Accuracy is measured over correctly and incorrectly classified instances. Results obtained show Naive Bayes outperforms with the highest accuracy of 76.30% comparatively other algorithms. These results are verified using Receiver Operating Characteristic (ROC) curves in a proper and systematic manner.

**IMPLEMENTATION :**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

import os

print(os.listdir())

import warnings

warnings.filterwarnings('ignore')

# Importing the dataset

dataset = pd.read\_csv('diabetes.csv')

#ABOUT DATABASE

print("")

print("Shape of DATASET : ")

dataset.shape

dataset.info()

X = dataset.iloc[:, 0:8].values

#### Analysing the 'target' variable

dataset["Outcome"].describe()

Y = dataset.iloc[:, 8].values

#Checking Correlation:

print(dataset.corr()["Outcome"].abs().sort\_values(ascending=False))

#Graphs of columns :

dataset.hist(bins=50, figsize=(20,15))

print(" ")

# there is no categorical value here so we don't need labelEncoder and onehotencoder.

#checking TARGET column

target\_temp = dataset.Outcome.value\_counts()

print("Percentage of patience without problems: "+str(round(target\_temp[0]\*100/2000,2)))

print("Percentage of patience with problems: "+str(round(target\_temp[1]\*100/2000,2)))

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 0)

# CHECKING ALL CLASSIFICATION MODELS :

#LOGISTIC REGRESSION :

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X\_train,Y\_train)

Y\_pred\_lr = lr.predict(X\_test)

score\_lr = round(accuracy\_score(Y\_pred\_lr,Y\_test)\*100,2)

print("The accuracy score achieved using Logistic Regression is: "+str(score\_lr)+" %")

print(classification\_report(Y\_test,Y\_pred\_lr))

fpr, tpr, thresholds = roc\_curve(Y\_test, Y\_pred\_lr)

plt.plot(fpr,tpr)

# Area Under The Curve (AUC)

print(roc\_auc\_score(Y\_test,Y\_pred\_lr))

# KERNEL Support Vector Regression:

from sklearn import svm

sv = svm.SVC(kernel='linear',random\_state=0)

sv.fit(X\_train, Y\_train)

Y\_pred\_svm = sv.predict(X\_test)

score\_svm = round(accuracy\_score(Y\_pred\_svm,Y\_test)\*100,2)

print("The accuracy score achieved using Linear SVM is: "+str(score\_svm)+" %")

print(classification\_report(Y\_test,Y\_pred\_svm))

fpr, tpr, thresholds = roc\_curve(Y\_test, Y\_pred\_svm)

plt.plot(fpr,tpr)

# Area Under The Curve (AUC)

print(roc\_auc\_score(Y\_test,Y\_pred\_svm))

#K-NN :

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=9 ,metric='minkowski', p=2)

# minkowski is for Euclidean distance.

knn.fit(X\_train,Y\_train)

Y\_pred\_knn=knn.predict(X\_test)

score\_knn = round(accuracy\_score(Y\_pred\_knn,Y\_test)\*100,2)

print("The accuracy score achieved using KNN is: "+str(score\_knn)+" %")

print(classification\_report(Y\_test,Y\_pred\_knn))

fpr, tpr, thresholds = roc\_curve(Y\_test, Y\_pred\_knn)

plt.plot(fpr,tpr)

# Area Under The Curve (AUC)

print(roc\_auc\_score(Y\_test,Y\_pred\_knn))

#DESICION TREE :

from sklearn.tree import DecisionTreeClassifier

max\_accuracy = 0

for x in range(200):

dt = DecisionTreeClassifier(random\_state=x)

dt.fit(X\_train,Y\_train)

Y\_pred\_dt = dt.predict(X\_test)

current\_accuracy = round(accuracy\_score(Y\_pred\_dt,Y\_test)\*100,2)

if(current\_accuracy>max\_accuracy):

max\_accuracy = current\_accuracy

best\_x = x

dt = DecisionTreeClassifier(random\_state=best\_x)

dt.fit(X\_train,Y\_train)

Y\_pred\_dt = dt.predict(X\_test)

score\_dt = round(accuracy\_score(Y\_pred\_dt,Y\_test)\*100,2)

print("The accuracy score achieved using Decision Tree is: "+str(score\_dt)+" %")

print(classification\_report(Y\_test,Y\_pred\_dt))

fpr, tpr, thresholds = roc\_curve(Y\_test, Y\_pred\_dt)

plt.plot(fpr,tpr)

# Area Under The Curve (AUC)

print(roc\_auc\_score(Y\_test,Y\_pred\_dt))

#RANDOM FOREST :

from sklearn.ensemble import RandomForestClassifier

max\_accuracy = 0

for x in range(2000):

rf = RandomForestClassifier(random\_state=x)

rf.fit(X\_train,Y\_train)

Y\_pred\_rf = rf.predict(X\_test)

current\_accuracy = round(accuracy\_score(Y\_pred\_rf,Y\_test)\*100,2)

if(current\_accuracy>max\_accuracy):

max\_accuracy = current\_accuracy

best\_x = x

rf = RandomForestClassifier(random\_state=best\_x)

rf.fit(X\_train,Y\_train)

Y\_pred\_rf = rf.predict(X\_test)

score\_rf = round(accuracy\_score(Y\_pred\_rf,Y\_test)\*100,2)

print("The accuracy score achieved using Random Forest is: "+str(score\_rf)+" %")

print(classification\_report(Y\_test,Y\_pred\_rf))

fpr, tpr, thresholds = roc\_curve(Y\_test, Y\_pred\_rf)

plt.plot(fpr,tpr)

# Area Under The Curve (AUC)

print(roc\_auc\_score(Y\_test,Y\_pred\_rf))

#NEURAL NETWORKS :

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

# Adding the input layer and the first hidden layer

model.add(Dense(11,activation='relu',input\_dim=8))

# Adding the second hidden layer

model.add(Dense(output\_dim = 11, init = 'uniform', activation = 'relu'))

#output Layer

model.add(Dense(1,activation='sigmoid'))

# Compiling the ANN

model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(X\_train,Y\_train,batch\_size=10 , nb\_epoch=50)

#nb\_epochs is tne no. of time the model will cycle through the data.

Y\_pred\_nn = model.predict(X\_test)

rounded = [round(x[0]) for x in Y\_pred\_nn]

Y\_pred\_nn = rounded

score\_nn = round(accuracy\_score(Y\_pred\_nn,Y\_test)\*100,2)

print("The accuracy score achieved using Neural Network is: "+str(score\_nn)+" %")

print(classification\_report(Y\_test,Y\_pred\_nn))

fpr, tpr, thresholds = roc\_curve(Y\_test, Y\_pred\_nn)

plt.plot(fpr,tpr)

# Area Under The Curve (AUC)

print(roc\_auc\_score(Y\_test,Y\_pred\_nn))

#COMAPRING ALL MODELS :

scores = [score\_lr,score\_svm,score\_knn,score\_dt,score\_rf,score\_nn]

algorithms = ["Logistic Regression","Support Vector Machine","K-Nearest Neighbors","Decision Tree","Random Forest","Neural Network"]

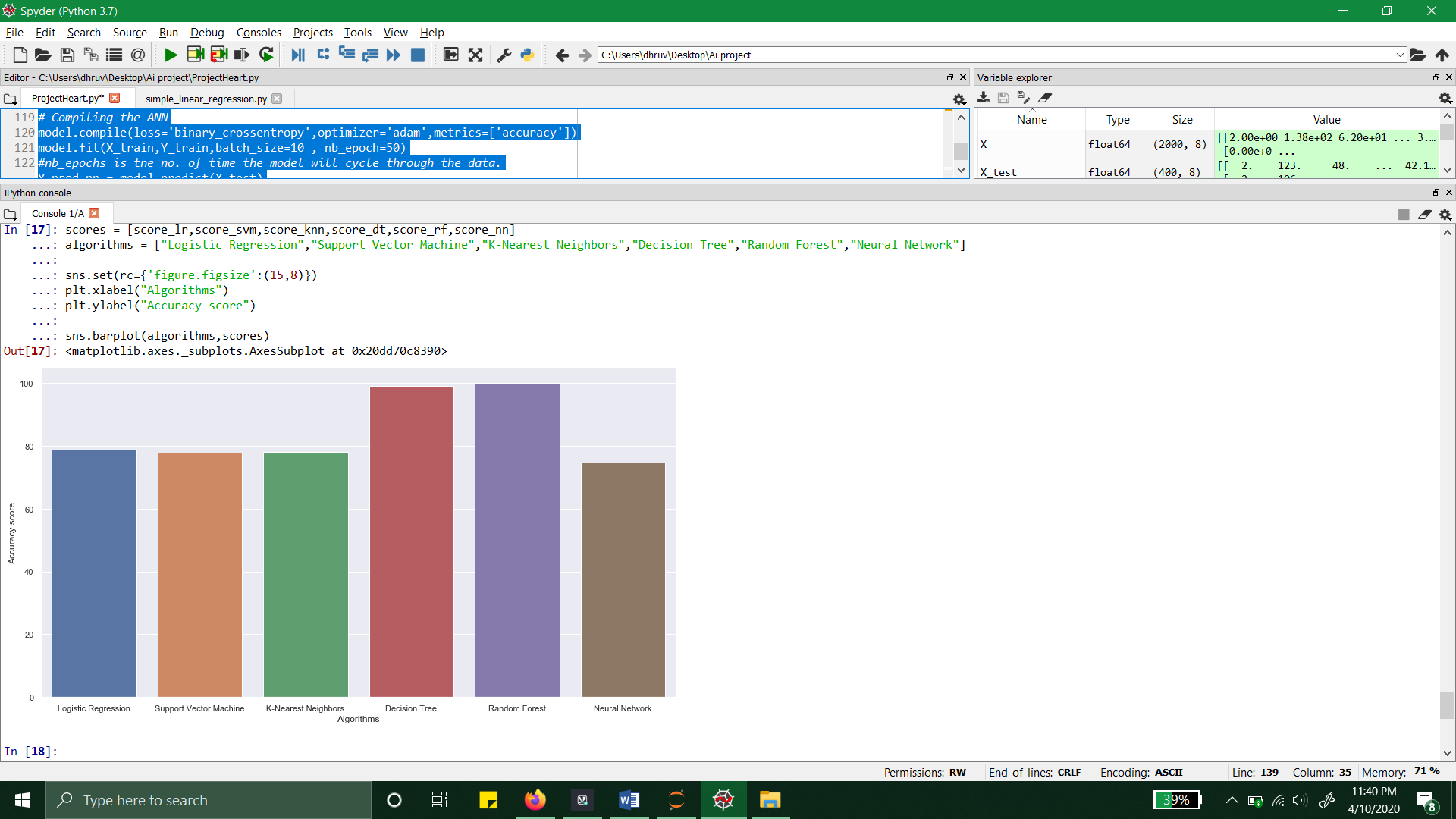
sns.set(rc={'figure.figsize':(15,8)})

plt.xlabel("Algorithms")

plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)

**EXPERIMENTAL RESULTS :**



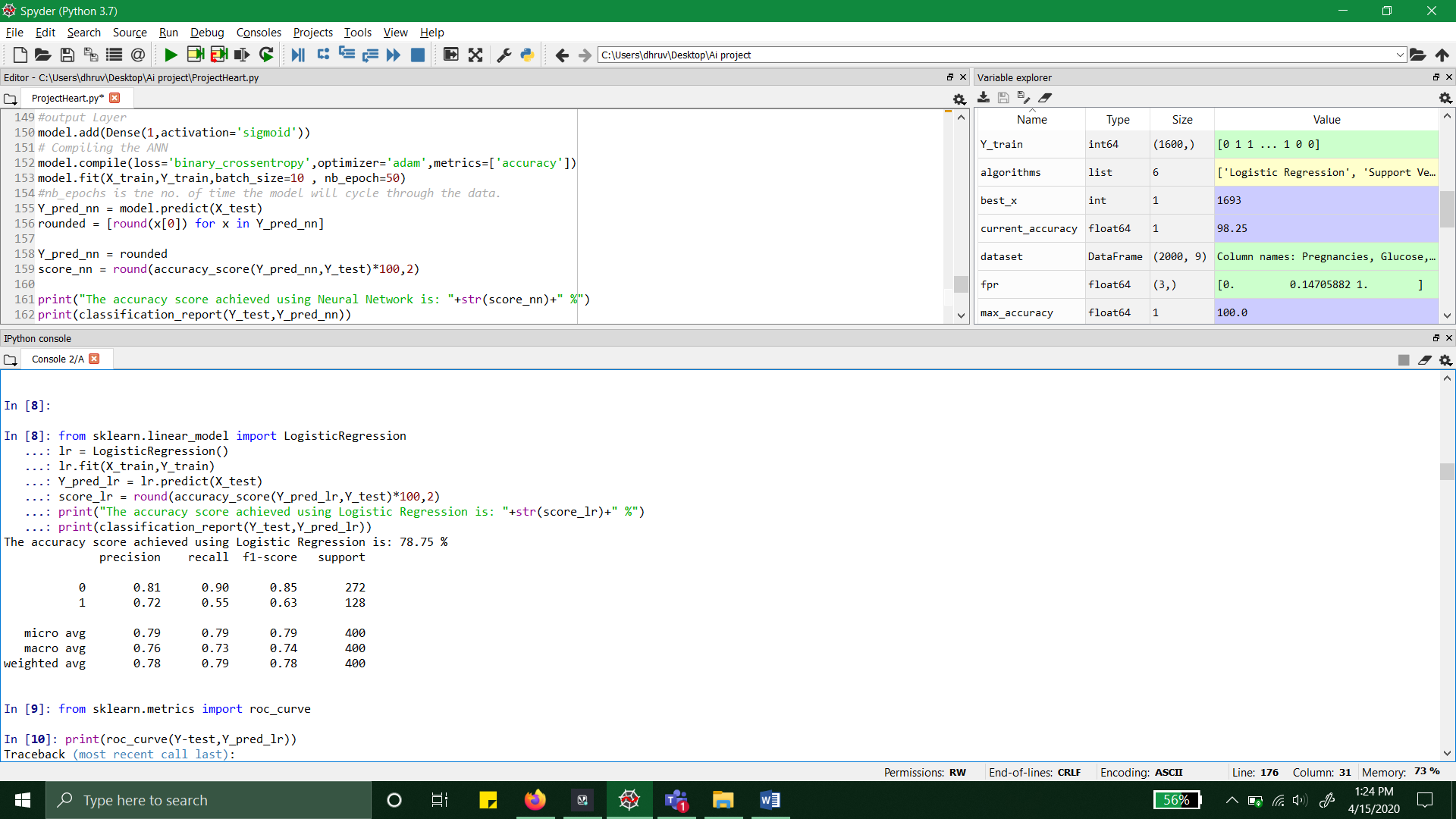
This Graph shows the ACCURACY SCORE of all the models applied to DATASET (diabetes.csv).

It is visible from this graph that RANDOM FOREST is best of all of Models

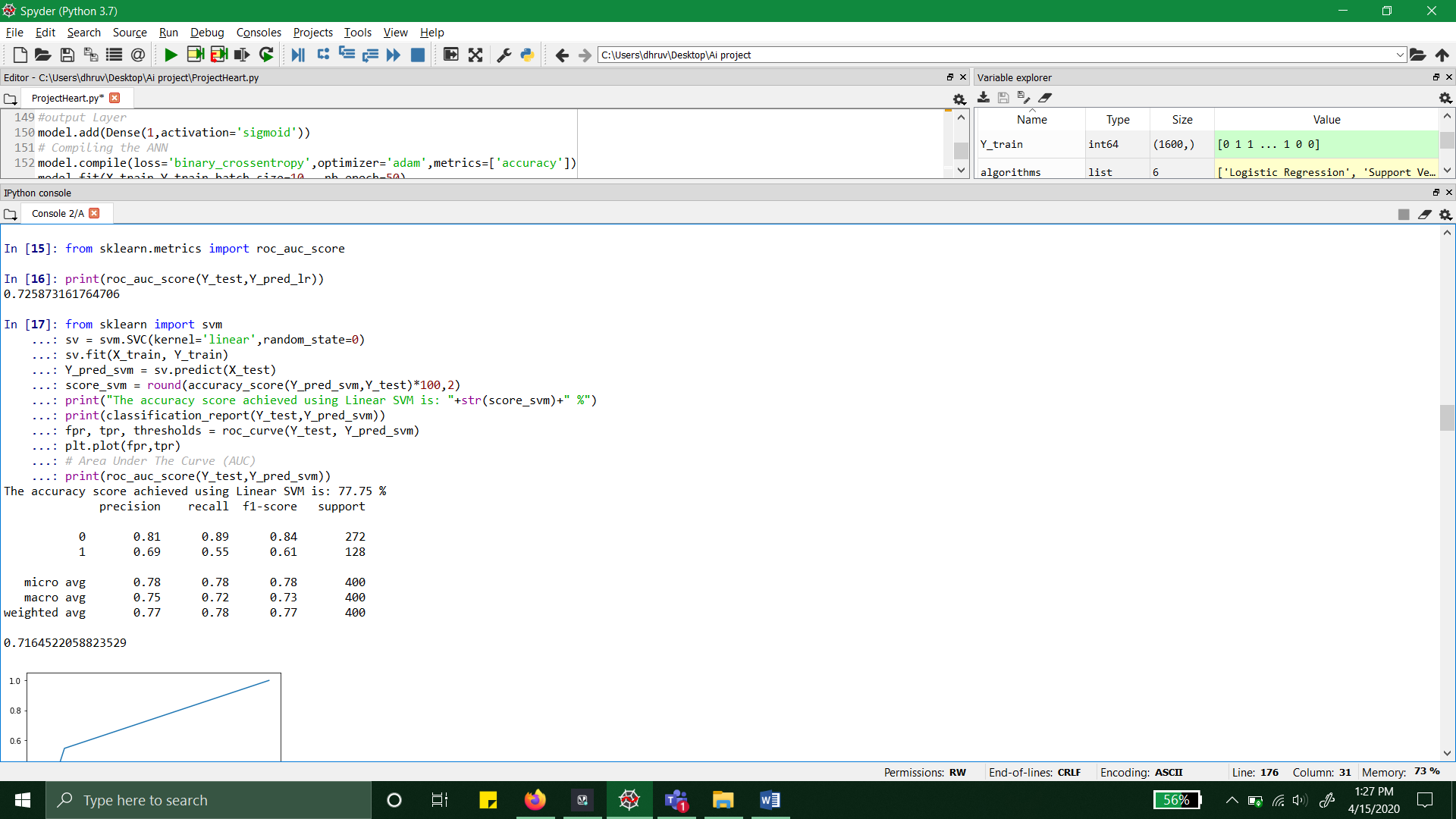
**RESULT**

**Classification Report :**

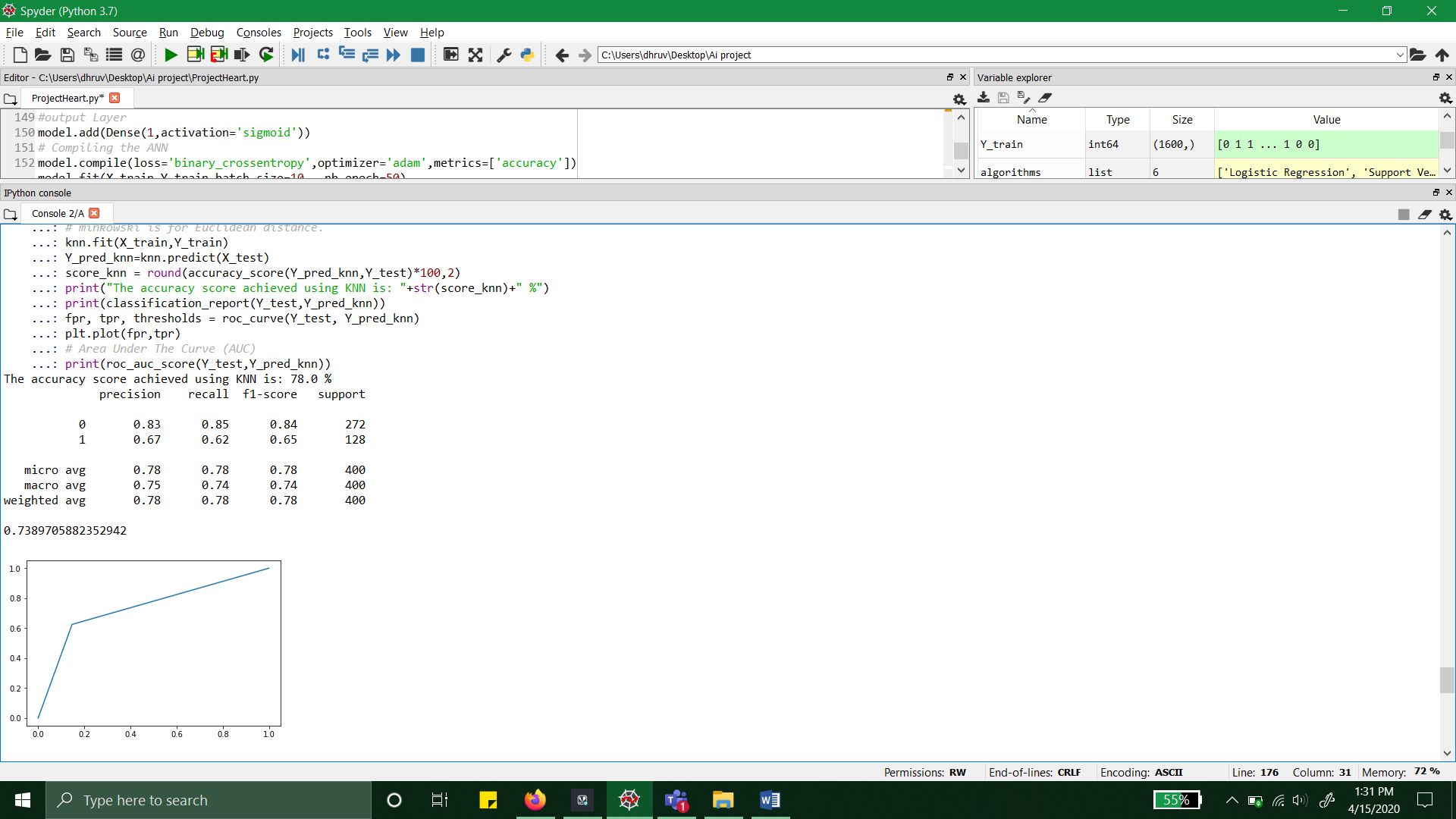
**1.LOGISTIC REGRESSION :**



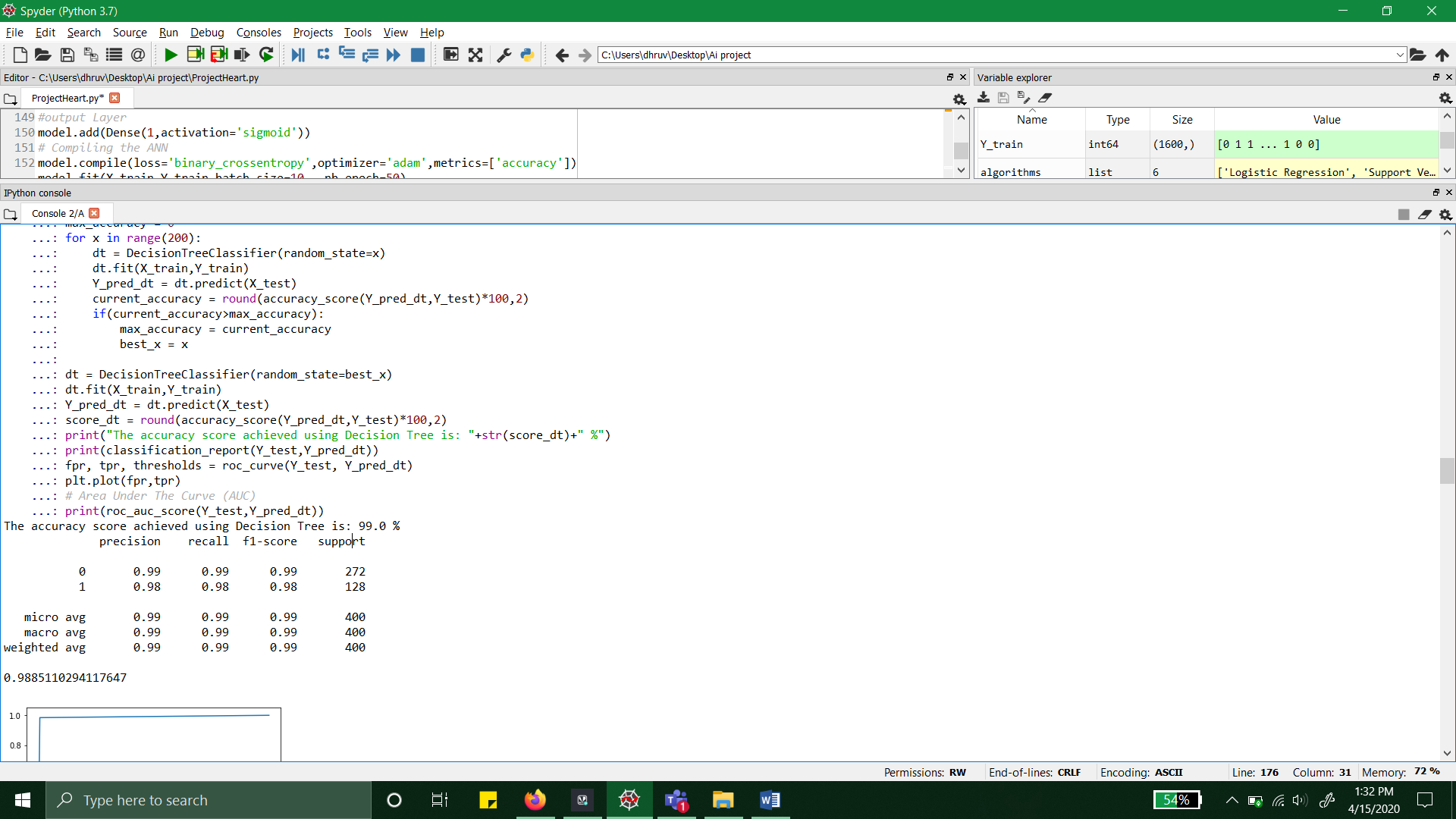
**2. SVM :**



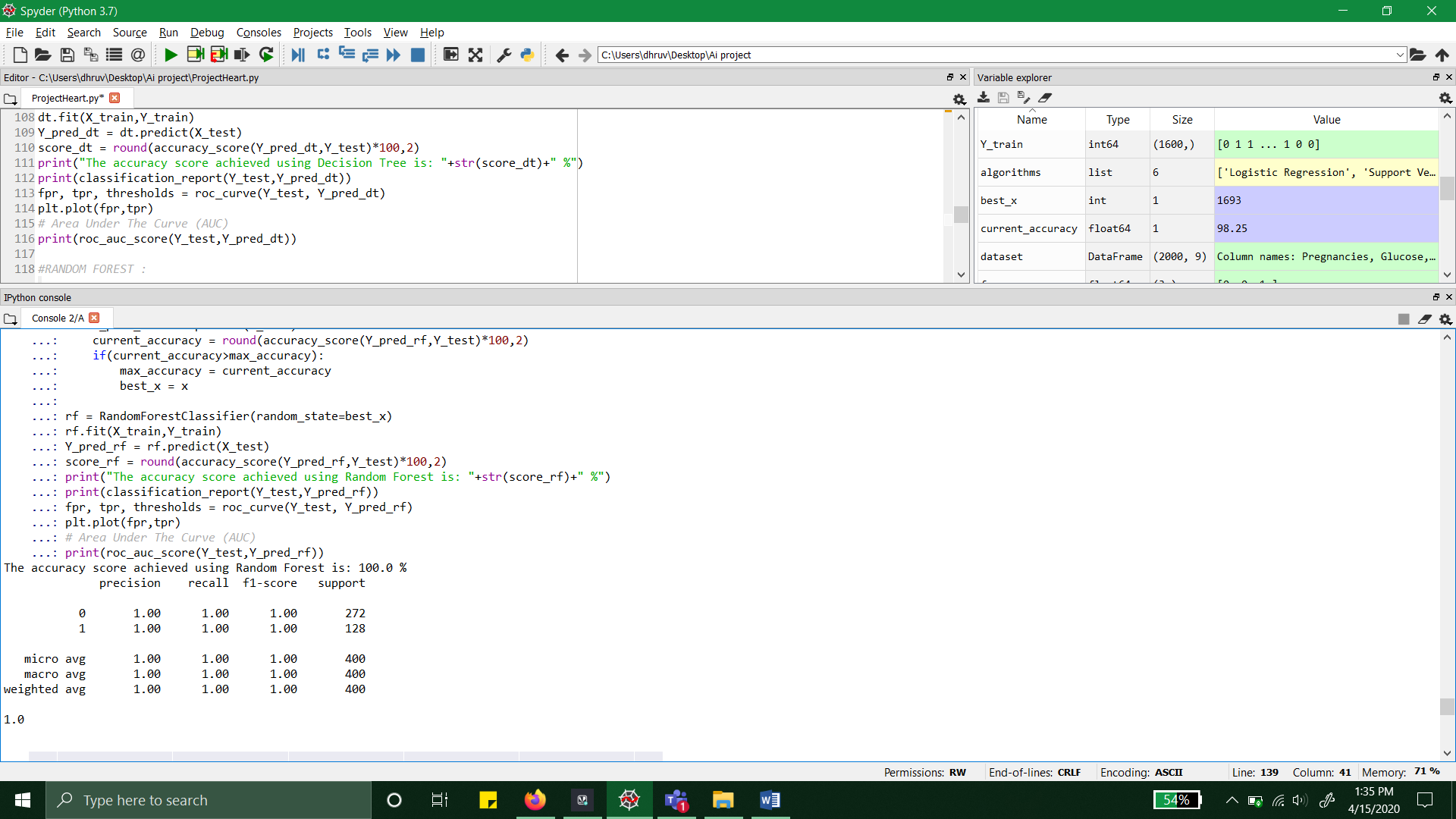
**3. KNN :**



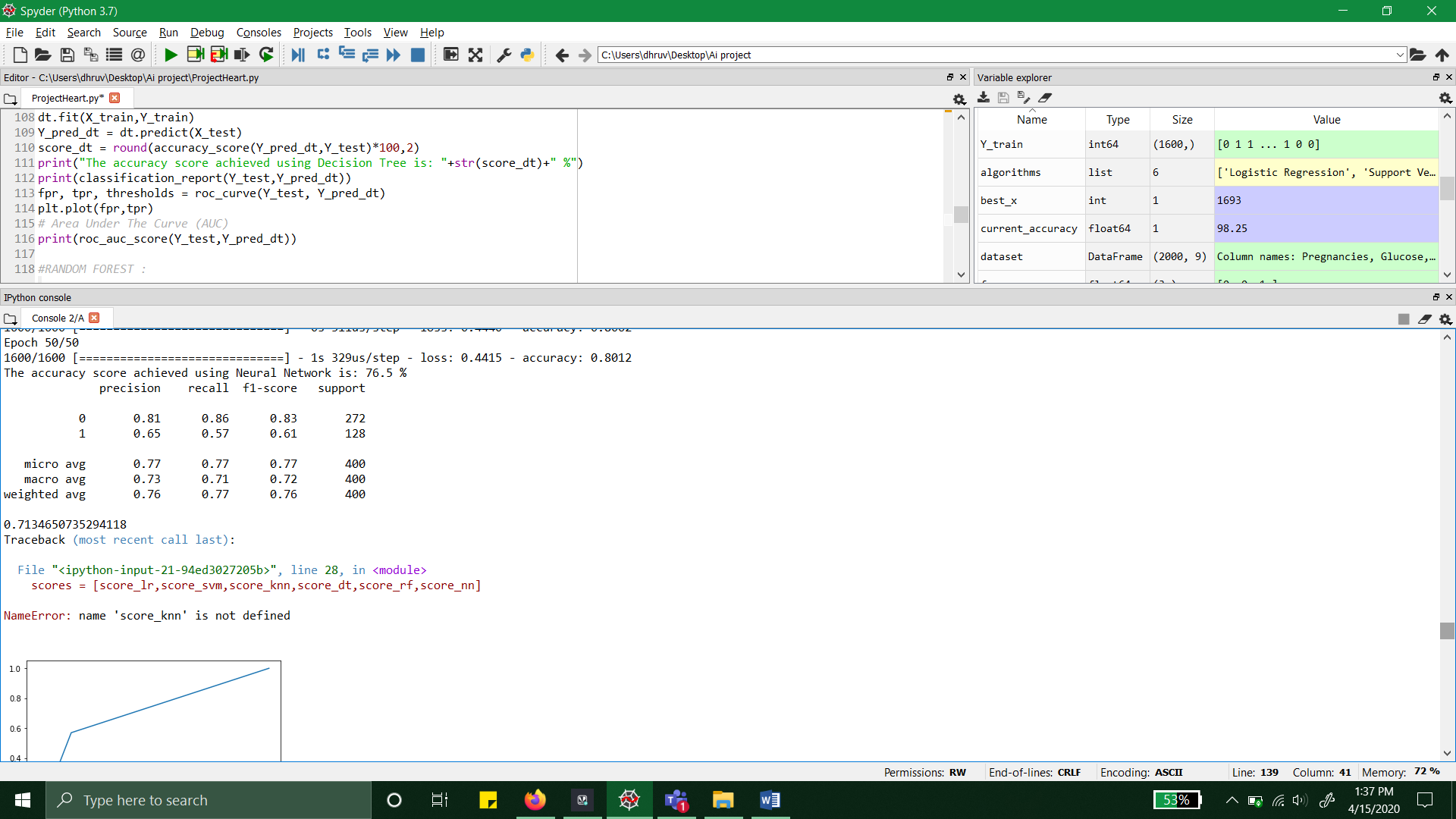
**4. DECISION TREE :**



**5. RANDOM FOREST :**



**6. NEURAL NETWORK :**



**CONCLUSION :**

**We got Highest Accuracy with RANDOM FOREST CLASSIFIER : 100%**

**REFERENCES:**

1. Aishwarya R., Gayathri P., Jaisankar N.

**A Method for Classification Using Machine Learning Technique for Diabetes**

International Journal of Engineering and Technology (IJET), 5 (2013), pp. 2903-2908

1. <https://www.kaggle.com/>
2. <https://www.youtube.com/>
3. Dilip Sisodia

**Prediction of Diabetes using Classification Algorithms**

8 June 2018.