HEART DISEASE PREDICTION USING MACHINE LEARNING

Competitive Programming

A PROJECT REPORT

Submitted by

M.Ubika-19BCD7176

M.Taran-19BCE7346

Under the Guidance of

Dr. D.Sumathi

Associate Professor, CSE,

VIT-AP

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1	Introduction	3
2	Background Study	3
3	Problem Definition	4
4	Objective	4
5	Methodology/Procedure	5
6	Results and Discussion	5
7	Conclusion and Future Scope	6
	References	6

Chapter -1

Introduction

“Machine Learning is a way of Manipulating and extraction of implicit, previously unknown/known and potential useful information about data”. Machine Learning is a very vast and diverse field and its scope and implementation is increasing day by day. Machine learning Incorporates various classifiers of Supervised, Unsupervised and Ensemble Learning which are used to predict and Find the Accuracy of the given dataset. We can use that knowledge in our project of Heart disease prediction as it will help a lot of people.

Chapter -2

Background Study

This work has been prompted by a sizable quantity of research into the diagnosis of cardiovascular heart disease utilising machine learning algorithms. A brief literature review is included in this essay. Several algorithms, including Logistic Regression, KNN, Random Forest Classifier, and others, have been used to effectively predict cardiovascular disease. Results show that each algorithm is capable of registering the specified objectives to a different degree.

The risk factors of coronary Heart disease or atherosclerosis is identified by McPherson using the inbuilt implementation algorithm using uses some techniques of Neural Network and were just accurately able to predict whether the test patient is suffering from the given disease or not. Diagnosis and prediction of Heart Disease and Blood Pressure along with other attributes using the aid of neural networks was introduced by R. Subramanian.

Using both the old and new machine learning and deep learning models, the prior model was able to determine the decision boundary. It facilitated the most fundamental and crucial factors/knowledge, like family history related to any cardiac condition. However, the accuracy of this model was much lower than that of newer models that are in the works, such as those that use artificial neural networks and other machine and deep learning algorithms to detect coronary heart disease. When a doctor tested the model using unfamiliar data, the model used the previously learned data to train on and predicted the outcome, allowing the accuracy of the model to be determined.

Chapter -3

Problem Definition

Cardiovascular diseases are very common these days, they describe a range of conditions that could affect human heart. World health organization estimates that 17.9 million global deaths from (Cardio vascular diseases) CVDs.

It is the primary reason of deaths in adults. Our project can help predict the people who are likely to diagnose with a heart disease by help of their medical history. It recognizes who all are having any symptoms of heart disease such as chest pain or high blood pressure and can help in diagnosing disease with less medical tests and effective treatments, so that they can be cured accordingly.

This project Consists of the comparison models Logistic regression, KNN, Random Forest Classifier, SVM, Naives Bayes, XGBoost, Decision Tree and Neural Network.

Chapter -4

Objective

The objective of this project is to check whether the patient is likely to be diagnosed with any cardiovascular heart diseases based on their medical attributes such as gender, age, chest pain, fasting sugar level, etc. A dataset is selected from the UCI repository with patient's medical history and attributes. By using this dataset, we predict whether the patient can have a heart disease or not. To predict this, we use 14 medical attributes of a patient and classify him if the patient is likely to have a heart disease. These medical attributes are trained under these algorithms: Logistic regression, Naive Bayes, SVM, KNN, Decision tree, Random Forest Classifier, XGBoost and neural network. Most efficient of these algorithms is Random Forest which gives us the accuracy of 86.89%. And, finally we classify patients that are at risk of getting a heart disease or not and also this method is totally cost efficient.

Chapter -5

Methodology/Procedure

This project shows the analysis of various machine learning algorithms, the algorithms that are used in this paper are Naive bayes, SVM, K nearest neighbors (KNN), Logistic Regression, Decision Tree Random Forest Classifiers, XGBoost and Neural network which can be helpful for practitioners or medical analysts for accurately diagnose Heart Disease. This document includes examining the journals, published paper and the data of cardiovascular disease of the recent times.

The methodology includes steps that transform given data into recognized data patterns for the knowledge of the users. The proposed methodology includes steps, where first step is referred as the collection of the data than in second stage it extracts significant values than the 3rd is the preprocessing stage where we explore the data. Data preprocessing deals with the missing values, cleaning of data and normalization depending on algorithms used. After pre-processing of data, classifier is used to classify the pre-processed data the classifier used in the proposed model are Naive bayes, SVM, K nearest neighbors (KNN), Logistic Regression, Decision Tree Random Forest Classifiers, XGBoost and Neural network. Finally, the proposed model is undertaken, where we evaluated our model on the basis of accuracy and performance using various performance metrics. Here in this model, an effective Heart Disease Prediction is done using different classifiers. This model uses 13 medical parameters such as chest pain, fasting sugar, blood pressure, cholesterol, age, sex etc. for prediction.

Chapter -6

Results and Discussion

From these results we can see that after using different algorithms, Decision tree, Random Forest Classifier and Neural Network yield a better results out of them. The algorithms that we used are more accurate, saves a lot of money i.e. it is efficient and faster than the other algorithms used. Moreover, the maximum accuracy obtained by Random forest is equal to 86.89%.

Chapter -7

Conclusion and Future Scope

A cardiovascular disease detection model has been developed using different Machine learning modelling techniques. This project predicts people with cardiovascular disease by extracting the patient medical history that leads to a fatal heart disease from a dataset that includes patients' medical history such as chest pain, sugar level, blood pressure, etc. This Heart Disease detection system assists a patient based on his/her clinical information of them been diagnosed with a previous heart disease. Use of more training data ensures the higher chances of the model to accurately predict whether the given person has a heart disease or not. By using these, computer aided techniques we can predict the patient fast and better and the cost can be reduced very much. There are a number of medical databases that we can work on as these Machine learning techniques are better and they can predict better than a human being which helps the patient as well as the doctors. Therefore, in conclusion this project helps us predict the patients who are diagnosed with heart diseases by applying different Machine Learning Models.

References (Sample)

1. Soni J, Ansari U, Sharma D & Soni S (2011). Predictive data mining for medical diagnosis: an overview of heart disease prediction. International Journal of Computer Applications, 17(8), 43-8

2. Dangare C S & Apte S S (2012). Improved study of heart disease prediction system using data mining classification techniques. International Journal of Computer Applications, 47(10), 44-8.

3. Ordonez C (2006). Association rule discovery with the train and test approach for heart disease prediction. IEEE Transactions on Information Technology in Biomedicine, 10(2), 334-43.

4. Shinde R, Arjun S, Patil P & Waghmare J (2015). An intelligent heart disease prediction system using k-means clustering and Naïve Bayes algorithm. International Journal of Computer Science and Information Technologies, 6(1), 637-9.

5. Bashir S, Qamar U & Javed M Y (2014, November). An ensemble-based decision support framework for intelligent heart disease diagnosis. In International Conference on Information Society (i-Society 2014) (pp. 259-64). IEEE.

6. Jee S H, Jang Y, Oh D J, Oh B H, Lee S H, Park S W & Yun Y D (2014). A coronary heart disease prediction model: the Korean Heart Study. BMJ open, 4(5), e005025.

7. Ganna A, Magnusson P K, Pedersen N L, de Faire U, Reilly M, Ärnlöv J & Ingelsson E (2013). Multilocus genetic risk scores for coronary heart disease prediction.

8. www.ieeeexplore.com

Appendix – A

Team Work and Work Management

Taran-Coding

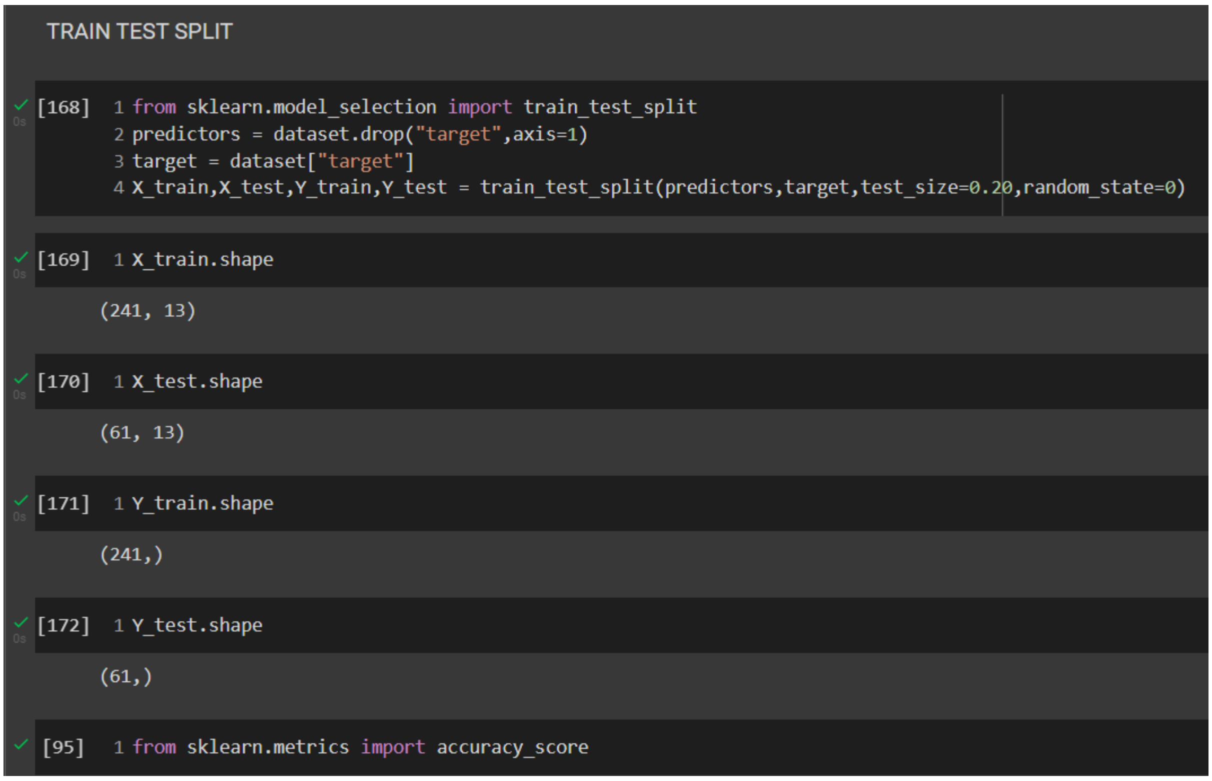
Ubika-Documentation

Appendix – B

Coding and Snap Shot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

dataset = pd.read_csv("sample_data/heart.csv")
type(dataset)
dataset.shape
dataset.head(5)
dataset.sample(5)
dataset.describe()
dataset.info()

info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina,3:
non-anginal pain, 4: asymptomatic",
        "resting blood pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120
mg/dl",
        "resting electrocardiographic results (values 0,1,2)"," maximum heart rate
achieved","exercise induced angina",
        "oldpeak = ST depression induced by exercise relative to rest","the slope of the peak
exercise ST segment",
    "number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed
defect; 7 = reversable defect"]

for i in range(len(info)):
   print(dataset.columns[i]+":\t\t\t"+info[i])

dataset["target"].describe()
dataset["target"].unique()
print(dataset.corr()["target"].abs().sort_values(ascending=False))
y = dataset["target"]
sns.countplot(y)
target_temp = dataset.target.value_counts()
print(target_temp)
print("Percentage of patience without heart problems:
"+str(round(target_temp[0]*100/302,2)))
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/302,2)))
dataset["sex"].unique()
sns.barplot(dataset["sex"],y)
dataset["cp"].unique()
sns.barplot(dataset["cp"],y)
dataset["fbs"].describe()
dataset["fbs"].unique()
sns.barplot(dataset["fbs"],y)
dataset["restecg"].unique()
sns.barplot(dataset["restecg"],y)
dataset["exang"].unique()
sns.barplot(dataset["exang"],y)
dataset["slope"].unique()
sns.barplot(dataset["slope"],y)
dataset["ca"].unique()
sns.countplot(dataset["ca"])
sns.barplot(dataset["ca"],y)
dataset["thal"].unique()
sns.barplot(dataset["thal"],y)
sns.distplot(dataset["thal"])

from sklearn.model_selection import train_test_split
predictors = dataset.drop("target",axis=1)
target = dataset["target"]
X_train,X_test,Y_train,Y_test =
train_test_split(predictors,target,test_size=0.20,random_state=0)
X_train.shape
X_test.shape
Y_train.shape
```

```
Y_test.shape

from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
pipe = make_pipeline(StandardScaler(), LogisticRegression())
pipe.fit(X_train,Y_train)
Y_pred_lr = lr.predict(X_test)
Y_pred_lr.shape
score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)
print("The accuracy score achieved using Logistic Regression is: "+str(score_lr)+" %")

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,Y_train)
Y_pred_nb = nb.predict(X_test)
Y_pred_nb.shape
score_nb = round(accuracy_score(Y_pred_nb,Y_test)*100,2)
print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")

from sklearn import svm
sv = svm.SVC(kernel='linear')
sv.fit(X_train, Y_train)
Y_pred_svm = sv.predict(X_test)
Y_pred_svm.shape
score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)
print("The accuracy score achieved using Linear SVM is: "+str(score_svm)+" %")

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
Y_pred_knn=knn.predict(X_test)
Y_pred_knn.shape
score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)
print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")

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)
print(Y_pred_dt.shape)
score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
print("The accuracy score achieved using Decision Tree is: "+str(score_dt)+" %")

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)+" %")

import xgboost as xgb
xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
xgb_model.fit(X_train, Y_train)
Y_pred_xgb = xgb_model.predict(X_test)
Y_pred_xgb.shape
score_xgb = round(accuracy_score(Y_pred_xgb,Y_test)*100,2)
print("The accuracy score achieved using XGBoost is: "+str(score_xgb)+" %")


from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(11,activation='relu',input_dim=13))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(X_train,Y_train,epochs=300)
Y_pred_nn = model.predict(X_test)
Y_pred_nn.shape
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)+" %")


scores = [score_lr,score_nb,score_svm,score_knn,score_dt,score_rf,score_xgb,score_nn]
algorithms = ["Logistic Regression","Naive Bayes","Support Vector Machine","K-Nearest 
Neighbors","Decision Tree","Random Forest","XGBoost","Neural Network"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")

sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")
sns.barplot(algorithms,scores)
```



TRAIN TEST SPLIT

```
[168]
from sklearn.model_selection import train_test_split
predictors = dataset.drop("target",axis=1)
target = dataset["target"]
X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_size=0.20,random_state=0)
```

```
[169]
X_train.shape
```

```
(241, 13)
```

```
[170]
X_test.shape
```

```
(61, 13)
```

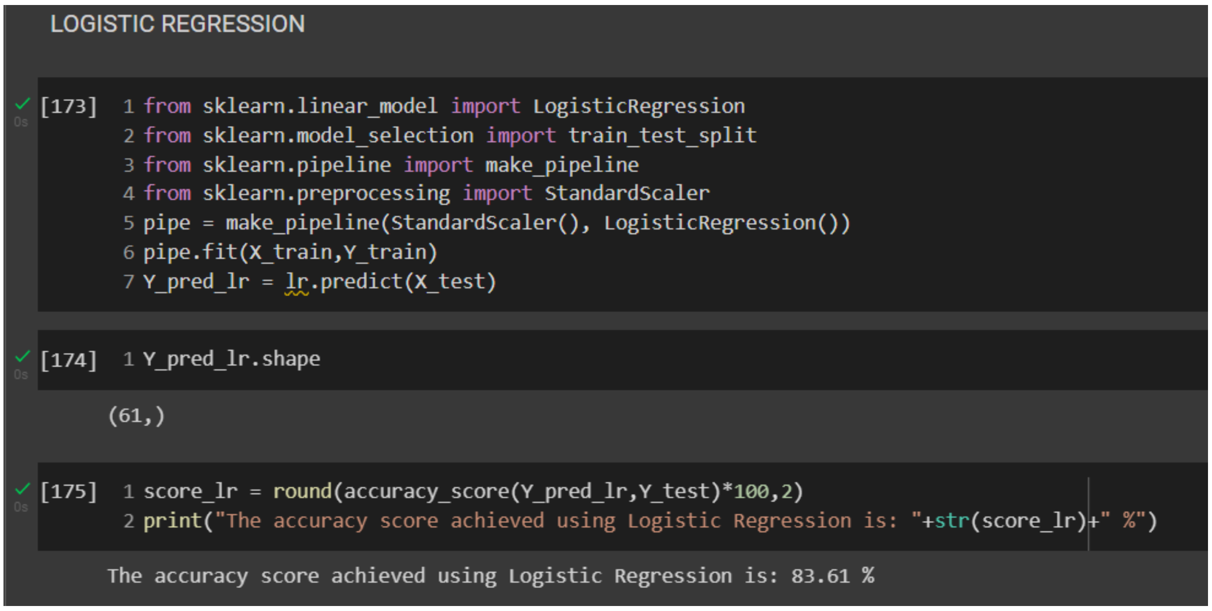
```
[171]
Y_train.shape
```

```
(241,)
```

```
[172]
Y_test.shape
```

```
(61,)
```

```
[95]
from sklearn.metrics import accuracy_score
```

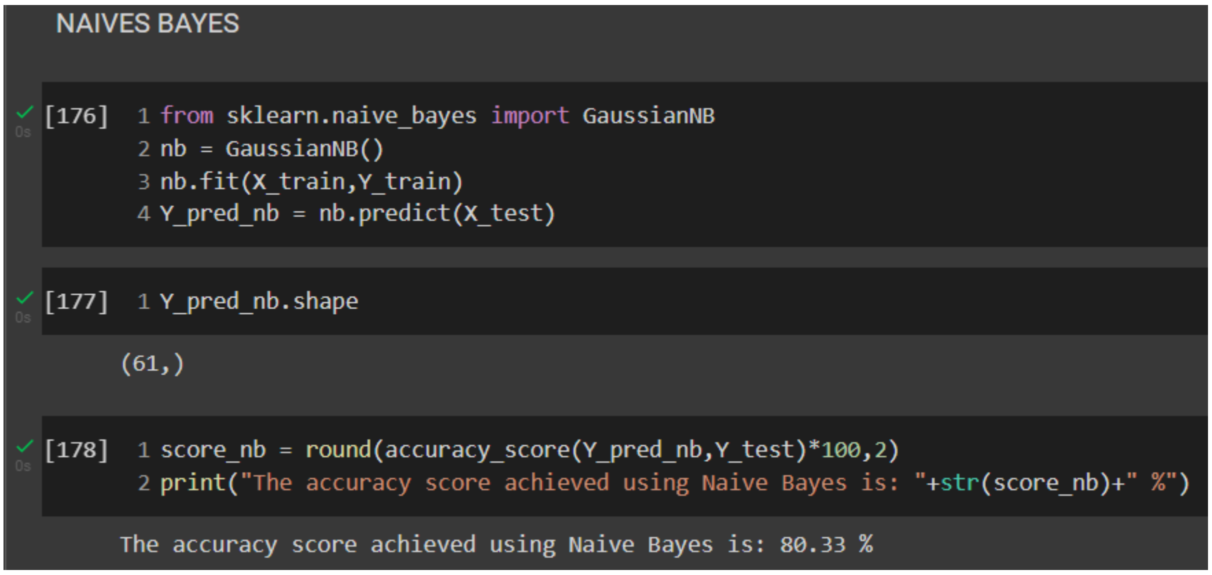


LOGISTIC REGRESSION

```
[173]  from sklearn.linear_model import LogisticRegression
       from sklearn.model_selection import train_test_split
       from sklearn.pipeline import make_pipeline
       from sklearn.preprocessing import StandardScaler
       pipe = make_pipeline(StandardScaler(), LogisticRegression())
       pipe.fit(X_train,Y_train)
       Y_pred_lr = lr.predict(X_test)
```

```
[174]  Y_pred_lr.shape

(61,)
```

```
[175]  score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)
       print("The accuracy score achieved using Logistic Regression is: "+str(score_lr)+" %")

The accuracy score achieved using Logistic Regression is: 83.61 %
```

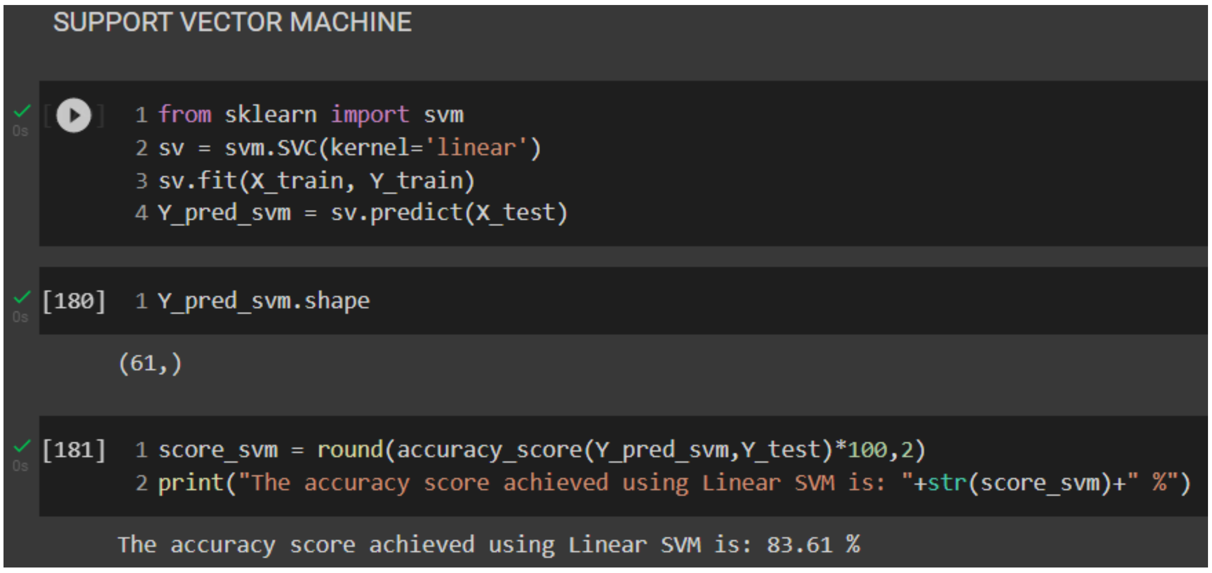


NAIVES BAYES

```
[176]  from sklearn.naive_bayes import GaussianNB
       nb = GaussianNB()
       nb.fit(X_train,Y_train)
       Y_pred_nb = nb.predict(X_test)
```

```
[177]  Y_pred_nb.shape

(61,)
```

```
[178]  score_nb = round(accuracy_score(Y_pred_nb,Y_test)*100,2)
       print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")

The accuracy score achieved using Naive Bayes is: 80.33 %
```

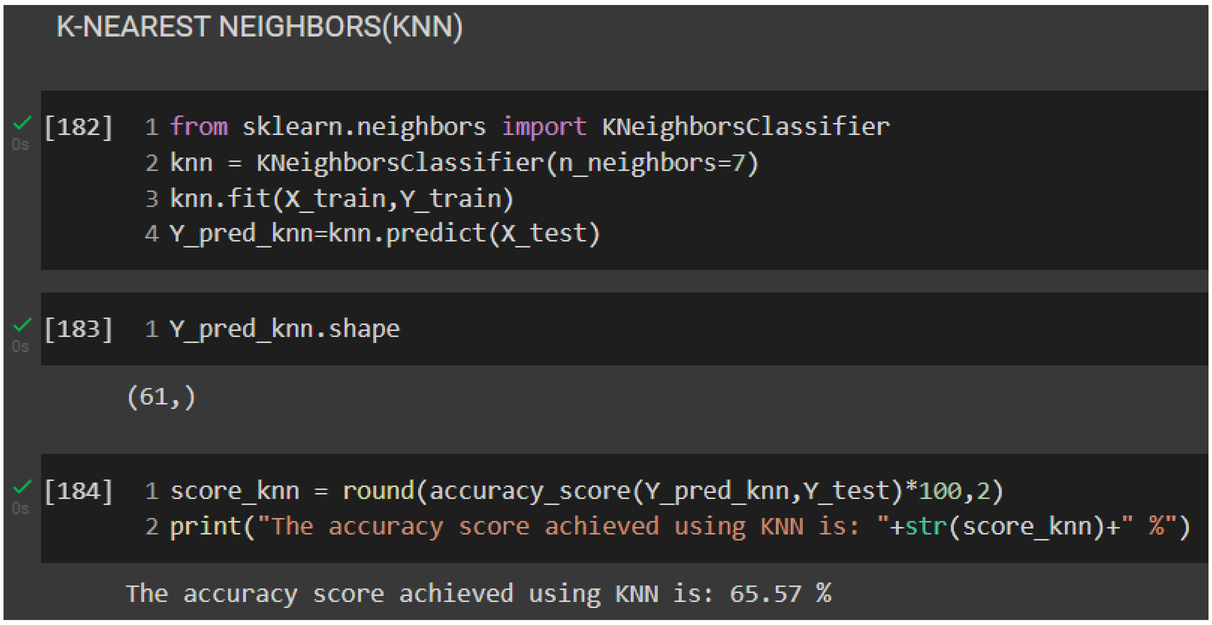


SUPPORT VECTOR MACHINE

```
       from sklearn import svm
       sv = svm.SVC(kernel='linear')
       sv.fit(X_train, Y_train)
       Y_pred_svm = sv.predict(X_test)
```

```
[180]  Y_pred_svm.shape

(61,)
```

```
[181]  score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)
       print("The accuracy score achieved using Linear SVM is: "+str(score_svm)+" %")

The accuracy score achieved using Linear SVM is: 83.61 %
```



K-NEAREST NEIGHBORS(KNN)

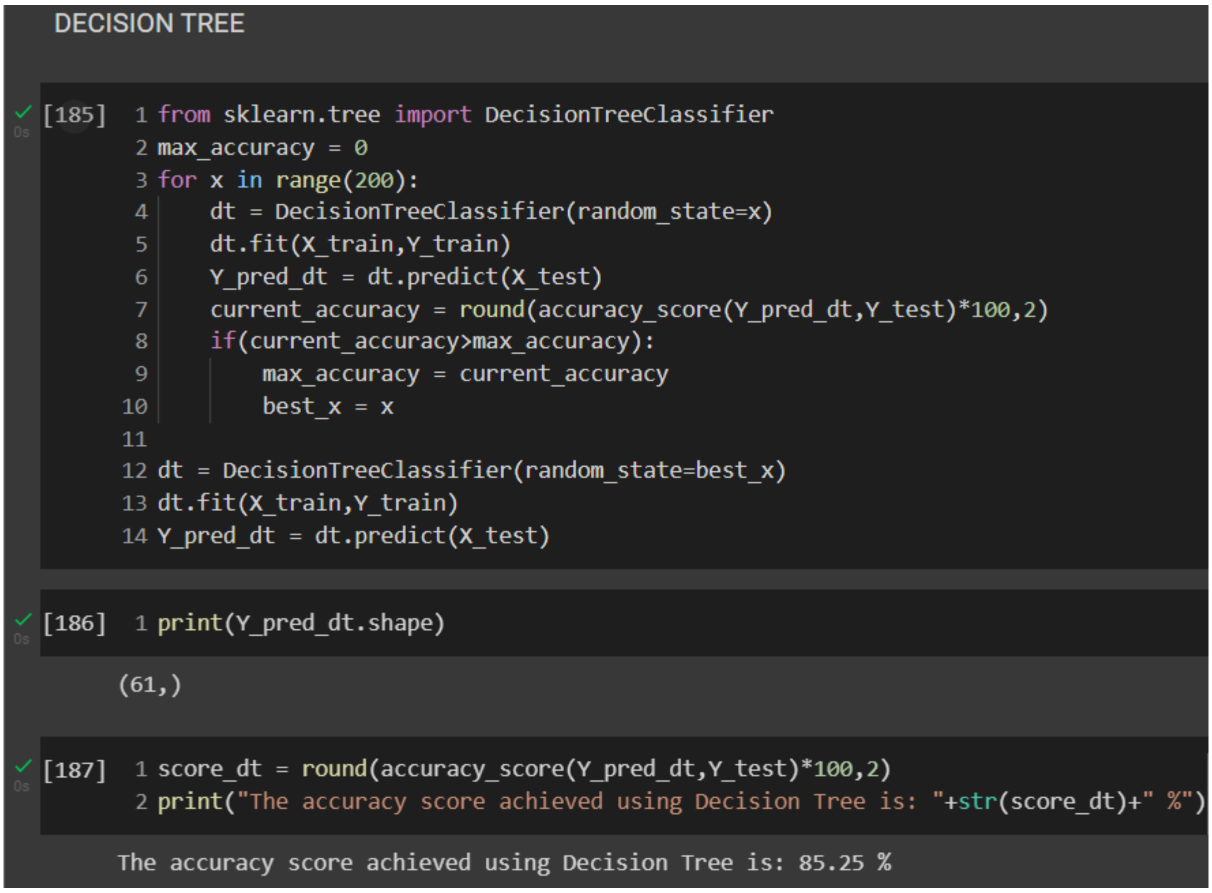
```
[182]
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
Y_pred_knn=knn.predict(X_test)
```

```
[183]
Y_pred_knn.shape
```

```
(61,)
```

```
[184]
score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)
print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")
```

```
The accuracy score achieved using KNN is: 65.57 %
```



DECISION TREE

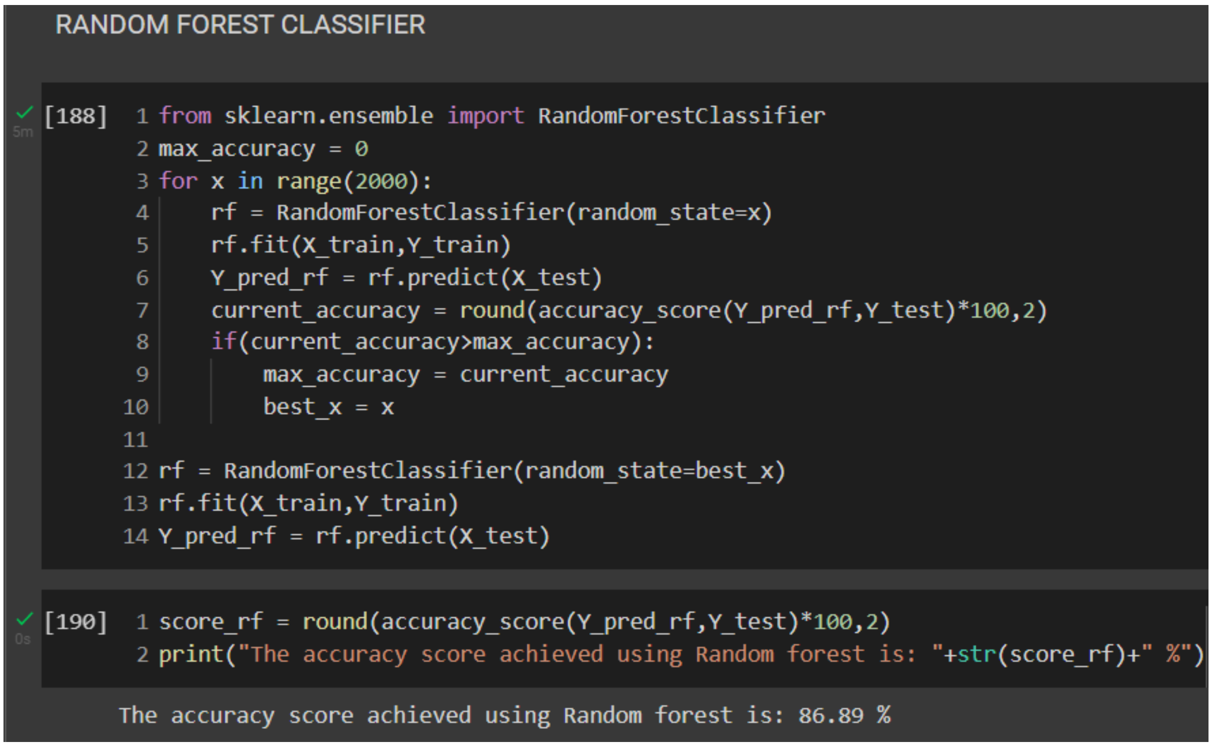
```
[185]
from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0
for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train,Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        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)
```

```
[186]
print(Y_pred_dt.shape)
```

```
(61,)
```

```
[187]
score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
print("The accuracy score achieved using Decision Tree is: "+str(score_dt)+" %")
```

```
The accuracy score achieved using Decision Tree is: 85.25 %
```

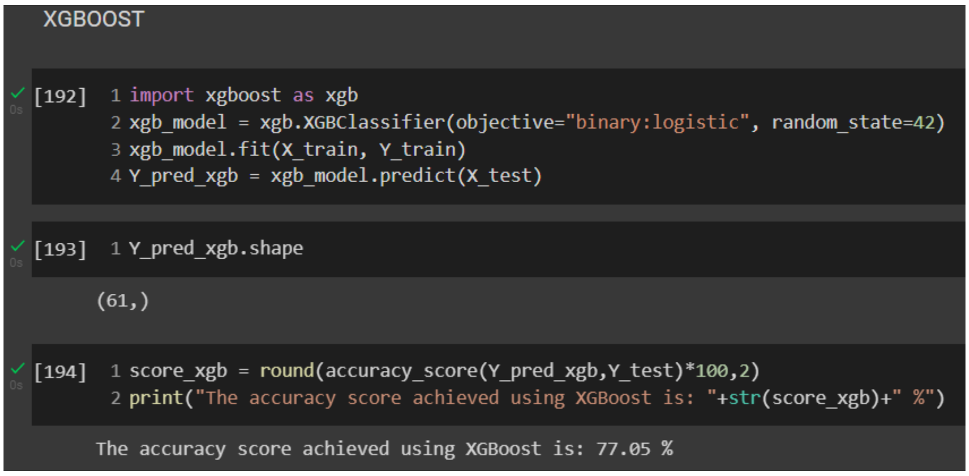


RANDOM FOREST CLASSIFIER

[188]
```python
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)
```

[190]
```python
score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
print("The accuracy score achieved using Random forest is: "+str(score_rf)+" %")
```

```
The accuracy score achieved using Random forest is: 86.89 %
```



XGBOOST

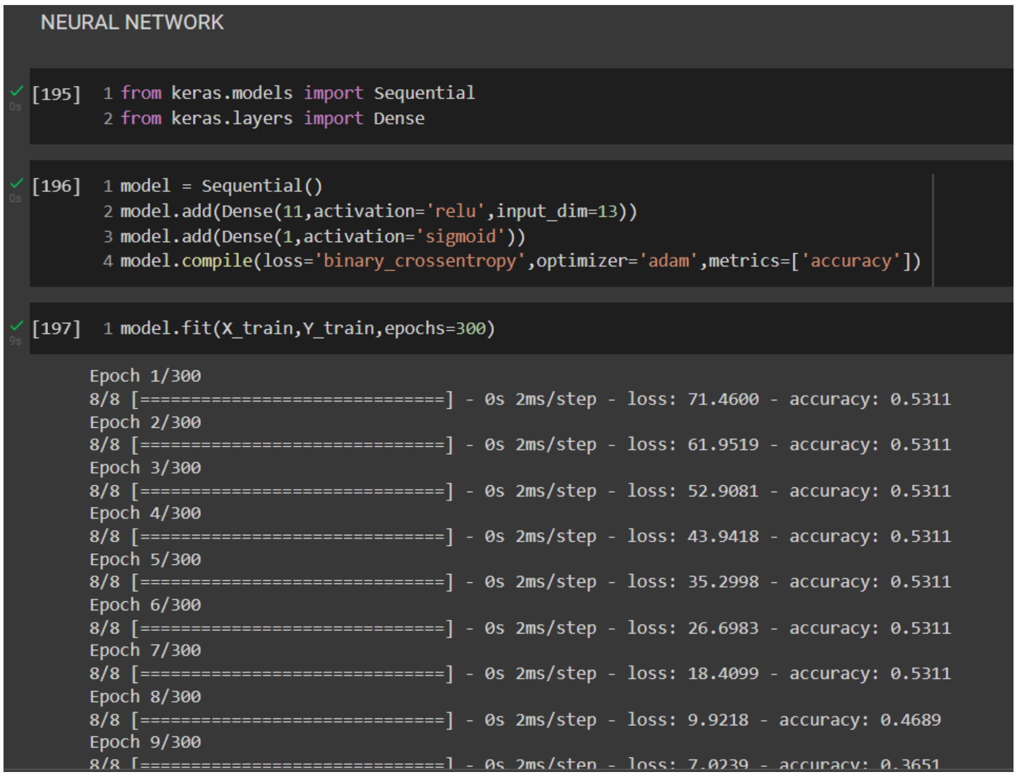
[192]
```python
import xgboost as xgb
xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
xgb_model.fit(X_train, Y_train)
Y_pred_xgb = xgb_model.predict(X_test)
```

[193]
```python
Y_pred_xgb.shape
```

```
(61,)
```

[194]
```python
score_xgb = round(accuracy_score(Y_pred_xgb,Y_test)*100,2)
print("The accuracy score achieved using XGBoost is: "+str(score_xgb)+" %")
```

```
The accuracy score achieved using XGBoost is: 77.05 %
```



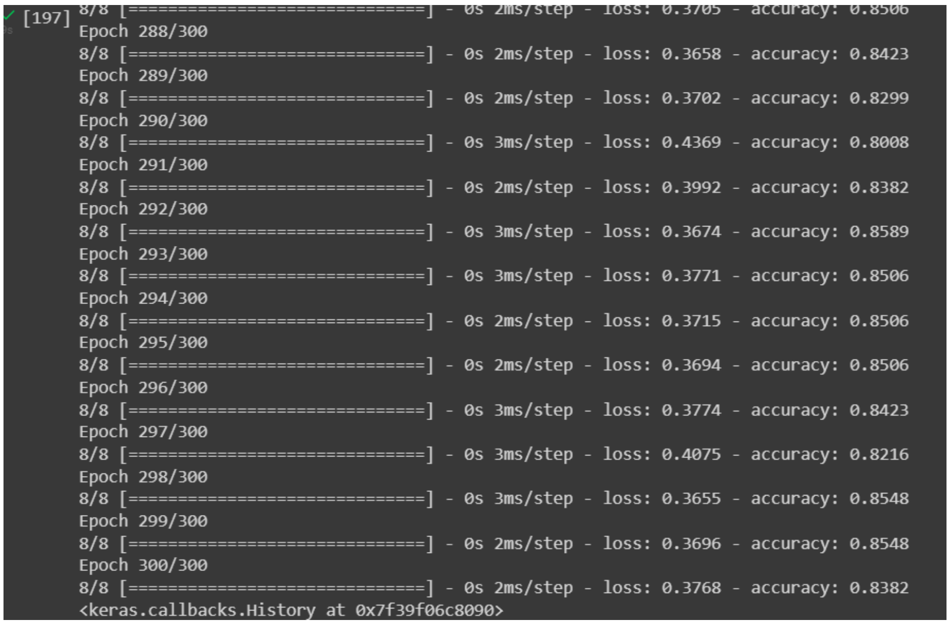
NEURAL NETWORK

```
[195]  from keras.models import Sequential
       from keras.layers import Dense
```

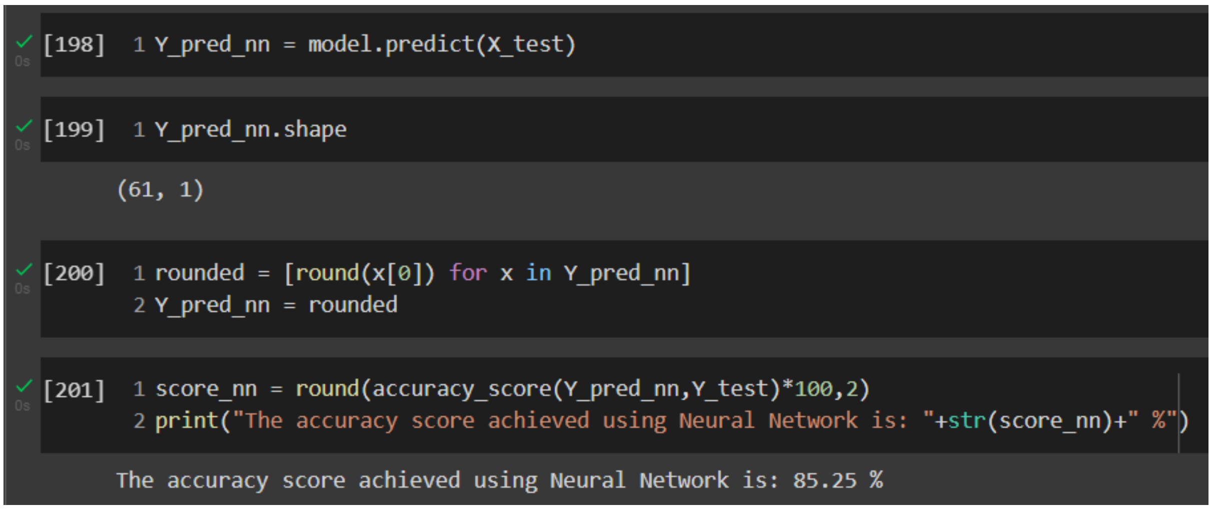
```
[196]  model = Sequential()
       model.add(Dense(11,activation='relu',input_dim=13))
       model.add(Dense(1,activation='sigmoid'))
       model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
[197]  model.fit(X_train,Y_train,epochs=300)
```

```
Epoch 1/300
8/8 [==============================] - 0s 2ms/step - loss: 71.4600 - accuracy: 0.5311
Epoch 2/300
8/8 [==============================] - 0s 2ms/step - loss: 61.9519 - accuracy: 0.5311
Epoch 3/300
8/8 [==============================] - 0s 2ms/step - loss: 52.9081 - accuracy: 0.5311
Epoch 4/300
8/8 [==============================] - 0s 2ms/step - loss: 43.9418 - accuracy: 0.5311
Epoch 5/300
8/8 [==============================] - 0s 2ms/step - loss: 35.2998 - accuracy: 0.5311
Epoch 6/300
8/8 [==============================] - 0s 2ms/step - loss: 26.6983 - accuracy: 0.5311
Epoch 7/300
8/8 [==============================] - 0s 2ms/step - loss: 18.4099 - accuracy: 0.5311
Epoch 8/300
8/8 [==============================] - 0s 2ms/step - loss: 9.9218 - accuracy: 0.4689
Epoch 9/300
8/8 [==============================] - 0s 2ms/step - loss: 7.0239 - accuracy: 0.3651
```



```
[197]  8/8 [==============================] - 0s 2ms/step - loss: 0.3705 - accuracy: 0.8506
Epoch 288/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3658 - accuracy: 0.8423
Epoch 289/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3702 - accuracy: 0.8299
Epoch 290/300
8/8 [==============================] - 0s 3ms/step - loss: 0.4369 - accuracy: 0.8008
Epoch 291/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3992 - accuracy: 0.8382
Epoch 292/300
8/8 [==============================] - 0s 3ms/step - loss: 0.3674 - accuracy: 0.8589
Epoch 293/300
8/8 [==============================] - 0s 3ms/step - loss: 0.3771 - accuracy: 0.8506
Epoch 294/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3715 - accuracy: 0.8506
Epoch 295/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3694 - accuracy: 0.8506
Epoch 296/300
8/8 [==============================] - 0s 3ms/step - loss: 0.3774 - accuracy: 0.8423
Epoch 297/300
8/8 [==============================] - 0s 3ms/step - loss: 0.4075 - accuracy: 0.8216
Epoch 298/300
8/8 [==============================] - 0s 3ms/step - loss: 0.3655 - accuracy: 0.8548
Epoch 299/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3696 - accuracy: 0.8548
Epoch 300/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3768 - accuracy: 0.8382
<keras.callbacks.History at 0x7f39f06c8090>
```



```
[198]  Y_pred_nn = model.predict(X_test)
```

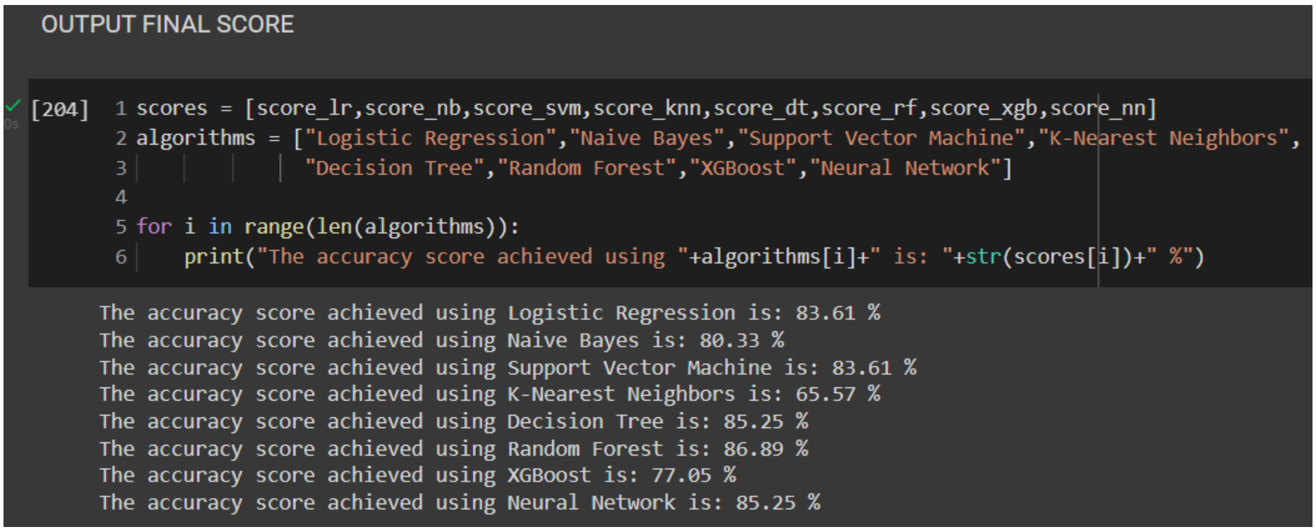
```
[199]  Y_pred_nn.shape
```

```
(61, 1)
```

```
[200]  rounded = [round(x[0]) for x in Y_pred_nn]
       Y_pred_nn = rounded
```

```
[201]  score_nn = round(accuracy_score(Y_pred_nn,Y_test)*100,2)
       print("The accuracy score achieved using Neural Network is: "+str(score_nn)+" %")
```

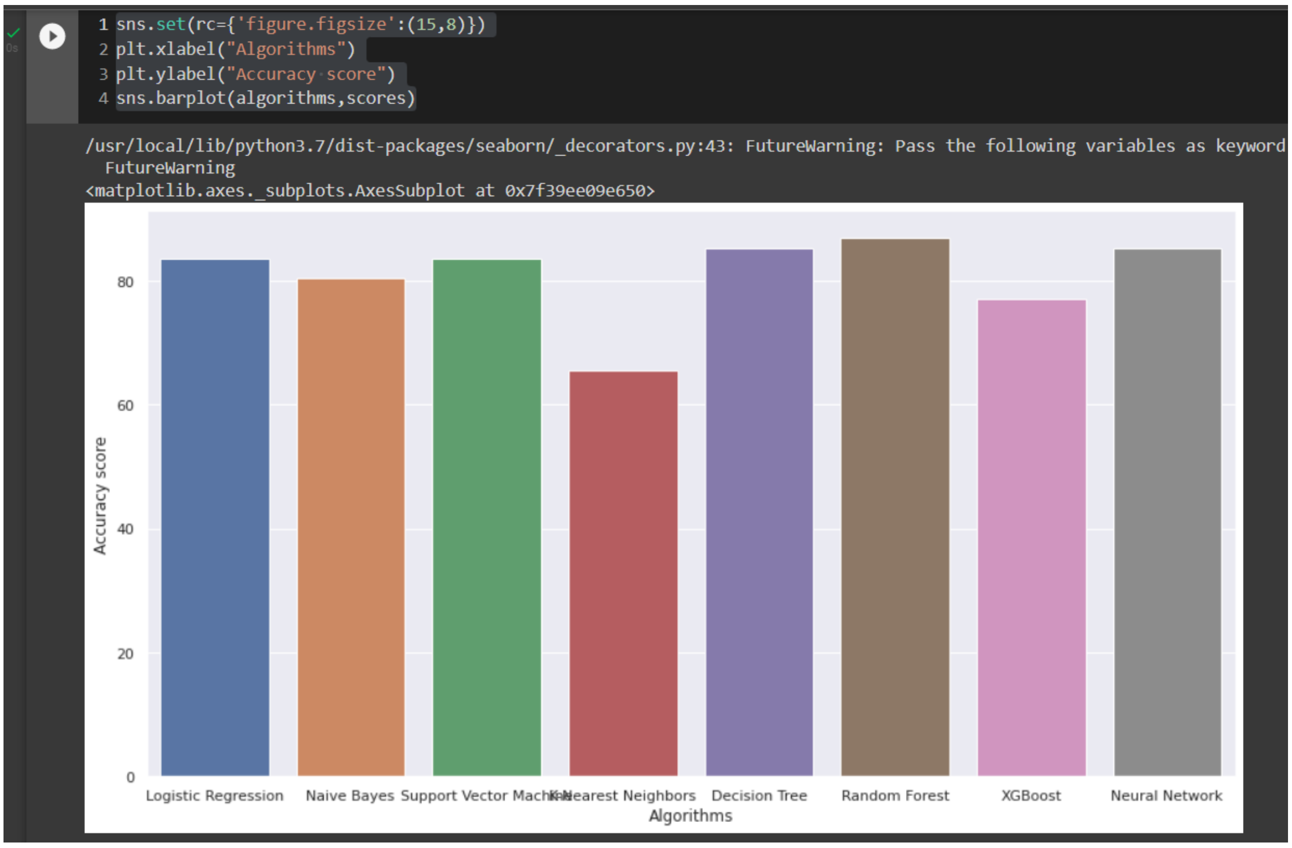
```
The accuracy score achieved using Neural Network is: 85.25 %
```



OUTPUT FINAL SCORE

```
[204]  scores = [score_lr,score_nb,score_svm,score_knn,score_dt,score_rf,score_xgb,score_nn]
       algorithms = ["Logistic Regression","Naive Bayes","Support Vector Machine","K-Nearest Neighbors",
                     "Decision Tree","Random Forest","XGBoost","Neural Network"]

       for i in range(len(algorithms)):
           print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Logistic Regression is: 83.61 %
The accuracy score achieved using Naive Bayes is: 80.33 %
The accuracy score achieved using Support Vector Machine is: 83.61 %
The accuracy score achieved using K-Nearest Neighbors is: 65.57 %
The accuracy score achieved using Decision Tree is: 85.25 %
The accuracy score achieved using Random Forest is: 86.89 %
The accuracy score achieved using XGBoost is: 77.05 %
The accuracy score achieved using Neural Network is: 85.25 %
```



```
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")
sns.barplot(algorithms,scores)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f39ee09e650>
```