Heart Disease Prediction

Submitted by-

NAME- SHAWGAT ASHRAFI

ASTU ROLL NUMBER- 170610107063

COLLEGE- ASSAM ENGINEERING COLLEGE

CLASS- 6TH SEMESTER

Duration: 2 Weeks

Problem Statement: Build a Machine Learning model for Heart Disease Prediction.

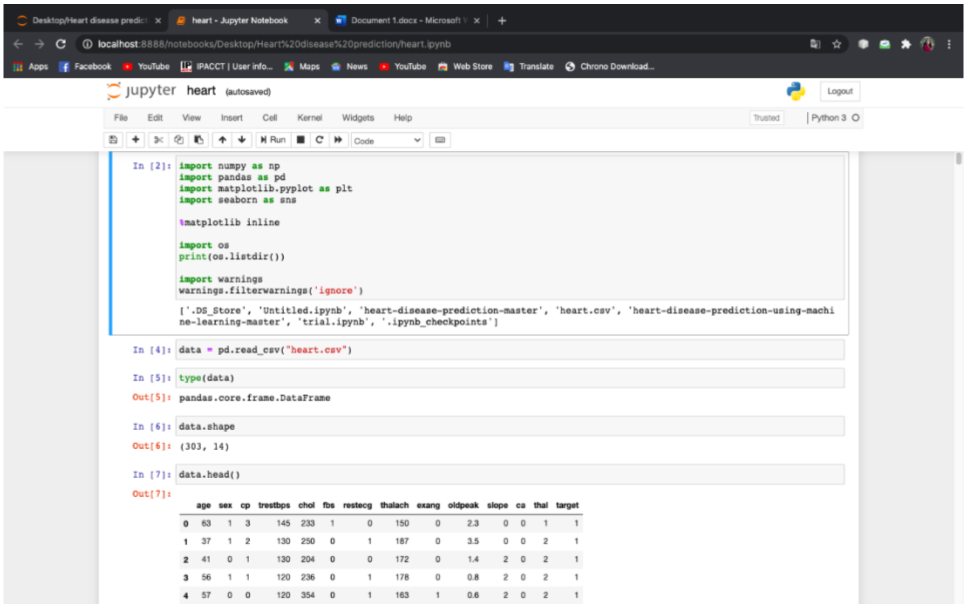
- **Data set:** heart.csv

- The heart.csv Data Set contains attributes like: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target.
- Apply 3-4 algorithms or classifiers like K Nearest Neighbors

Classifier, Random Forest Classifier, Decision Tree Classifier and Support Vector Classifier. Compare all the applied algorithms and try to find out the algorithm or classifier which is best fit to this data for prediction of disease (based on the evaluation of each model).

EXECUTION OF THE MODEL-

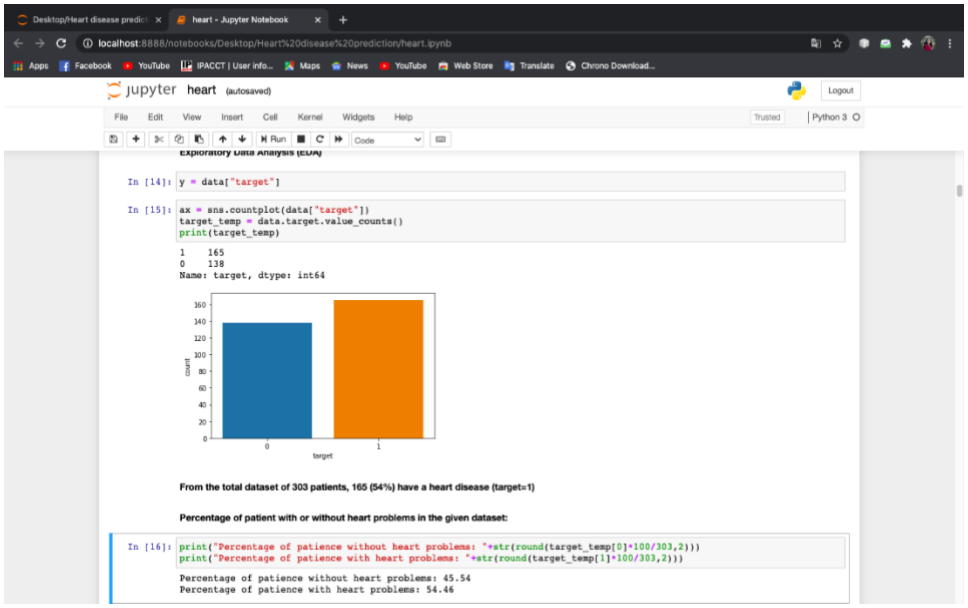
1. Importing the libraries:



```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline

        import os
        print(os.listdir())

        import warnings
        warnings.filterwarnings('ignore')

        ['.DS_Store', 'Untitled.ipynb', 'heart-disease-prediction-master', 'heart.csv', 'heart-disease-prediction-using-machi
        ne-learning-master', 'trial.ipynb', '.ipynb_checkpoints']

In [4]: data = pd.read_csv("heart.csv")

In [5]: type(data)

Out[5]: pandas.core.frame.DataFrame

In [6]: data.shape

Out[6]: (303, 14)

In [7]: data.head()

Out[7]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

2. Exploratory Data Analysis (EDA):



Exploratory Data Analysis (EDA)

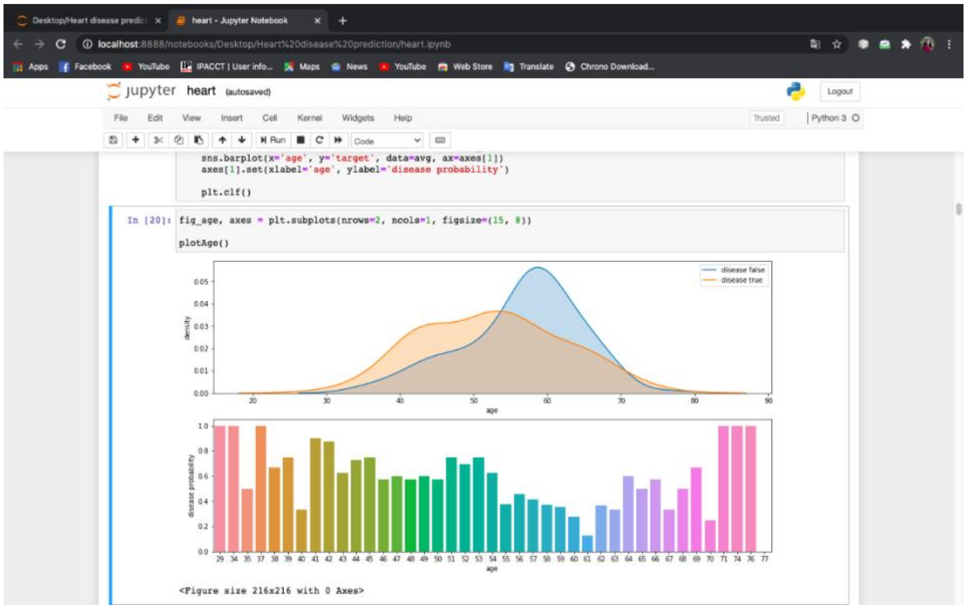
```
In [14]: y = data["target"]

In [15]: ax = sns.countplot(data["target"])
         target_temp = data.target.value_counts()
         print(target_temp)

         1    165
         0    138
         Name: target, dtype: int64
```

From the total dataset of 303 patients, 165 (54%) have a heart disease (target=1)

Percentage of patient with or without heart problems in the given dataset:

```
In [16]: print("Percentage of patience without heart problems: "+str(round(target_temp[0]*100/303,2)))
         print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/303,2)))

         Percentage of patience without heart problems: 45.54
         Percentage of patience with heart problems: 54.46
```

3. Density v/s age and disease probability v/s age:



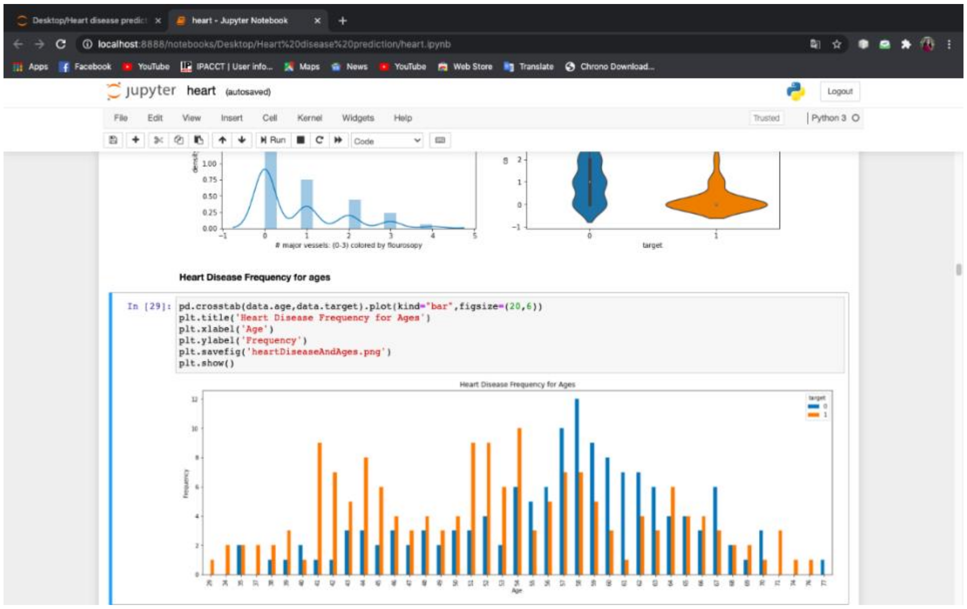
Desktop/Heart disease predict
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
jupyter heart (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
disease false
disease true
density
age
disease probability

```
sns.barplot(x='age', y='target', data=avg, ax=axes[1])
axes[1].set(xlabel='age', ylabel='disease probability')

plt.clf()
```

```
In [20]: fig_age, axes = plt.subplots(nrows=2, ncols=1, figsize=(15, 8))

         plotAge()
```

```
<Figure size 216x216 with 0 Axes>
```

4. Heart Disease Frequency for ages:

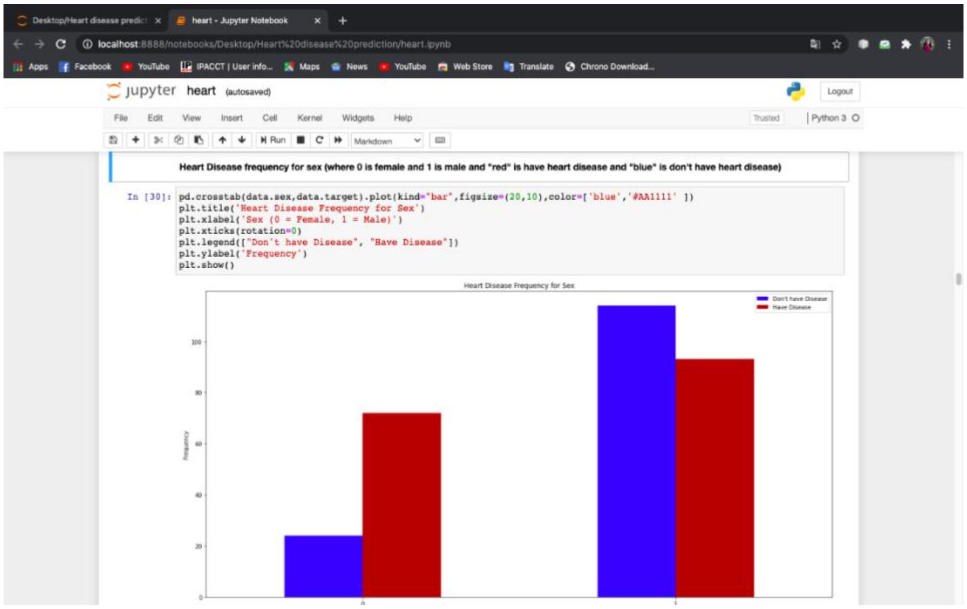


Desktop/Heart disease predict
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
jupyter heart (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
density
# major vessels: (0-3) colored by flourosopy
target
Heart Disease Frequency for Ages
Frequency
Age
target
0
1

Heart Disease Frequency for ages

```
In [29]: pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6))
         plt.title('Heart Disease Frequency for Ages')
         plt.xlabel('Age')
         plt.ylabel('Frequency')
         plt.savefig('heartDiseaseAndAges.png')
         plt.show()
```

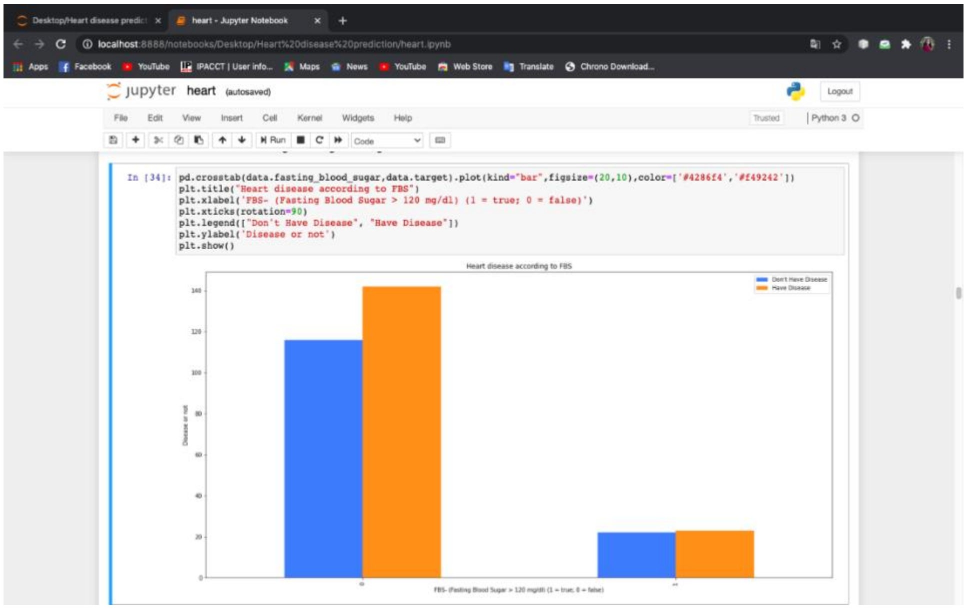
5. **Heart Disease frequency for sex (where 0 is female and 1 is male and "red" is have heart disease and "blue" is don't have heart disease):**



Heart Disease frequency for sex (where 0 is female and 1 is male and "red" is have heart disease and "blue" is don't have heart disease)

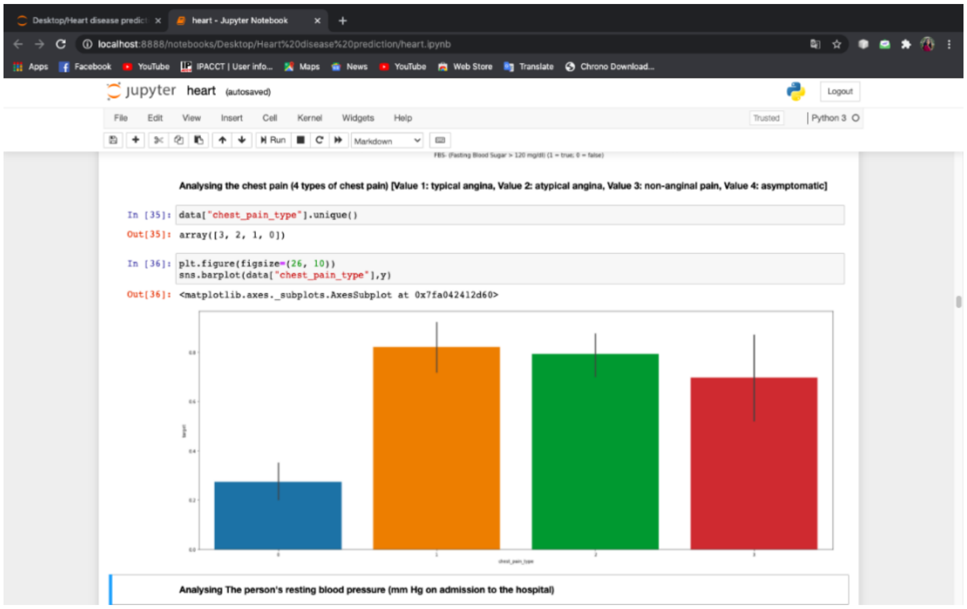
```
In [30]: pd.crosstab(data.sex,data.target).plot(kind="bar",figsize=(20,10),color=['blue','#AA1111' ])
         plt.title('Heart Disease Frequency for Sex')
         plt.xlabel('Sex (0 = Female, 1 = Male)')
         plt.xticks(rotation=0)
         plt.legend(["Don't have Disease", "Have Disease"])
         plt.ylabel('Frequency')
         plt.show()
```

6. **Heart disease according to Fasting Blood sugar:**



```
In [34]: pd.crosstab(data.fasting_blood_sugar,data.target).plot(kind="bar",figsize=(20,10),color=['#4286f4','#f49242'])
         plt.title("Heart disease according to FBS")
         plt.xlabel('FBS- (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)')
         plt.xticks(rotation=90)
         plt.legend(["Don't Have Disease", "Have Disease"])
         plt.ylabel('Disease or not')
         plt.show()
```

7. Analysing the chest pain (4 types of chest pain) [Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic]:



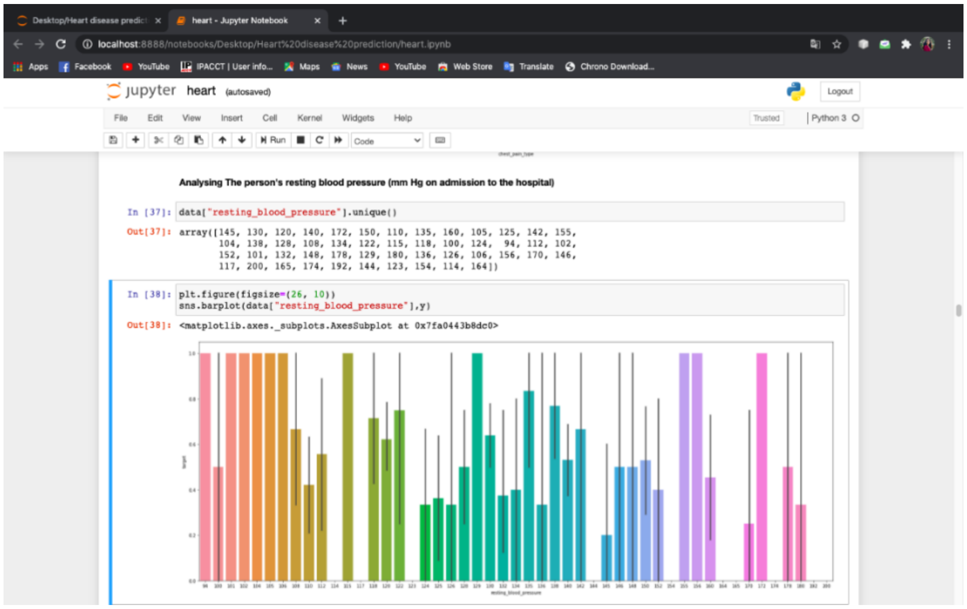
FBS- (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)

Analysing the chest pain (4 types of chest pain) [Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic]

```
In [35]: data["chest_pain_type"].unique()
Out[35]: array([3, 2, 1, 0])

In [36]: plt.figure(figsize=(26, 10))
         sns.barplot(data["chest_pain_type"],y)
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa042412d60>
```

Analysing The person's resting blood pressure (mm Hg on admission to the hospital)

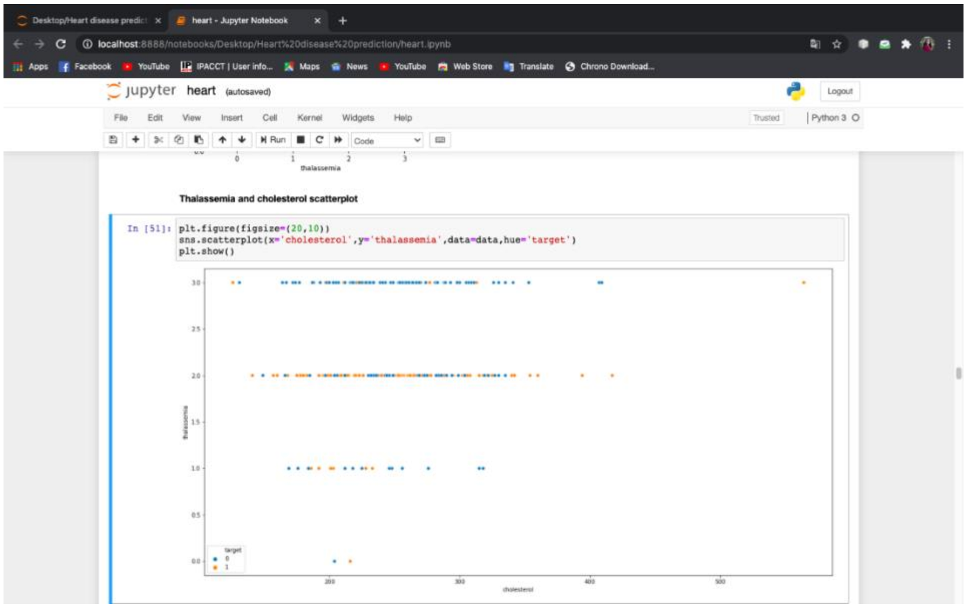
8. Analysing the person's resting blood pressure (mm Hg on admission to the hospital):



Analysing The person's resting blood pressure (mm Hg on admission to the hospital)

```
In [37]: data["resting_blood_pressure"].unique()
Out[37]: array([145, 130, 120, 140, 172, 150, 110, 135, 160, 105, 125, 142, 155,
                104, 138, 128, 108, 134, 122, 115, 118, 100, 124,  94, 112, 102,
                152, 101, 132, 148, 178, 129, 180, 136, 126, 106, 156, 170, 146,
                117, 200, 165, 174, 192, 144, 123, 154, 114, 164])

In [38]: plt.figure(figsize=(26, 10))
         sns.barplot(data["resting_blood_pressure"],y)
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa0443b8dc0>
```

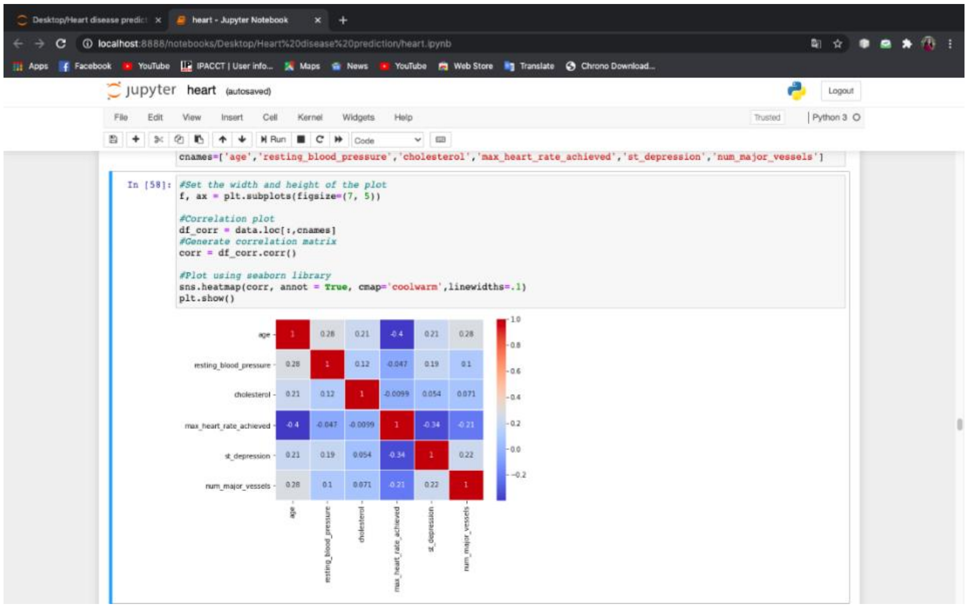
9. Thalassemia and cholesterol scatterplot:



Thalassemia and cholesterol scatterplot

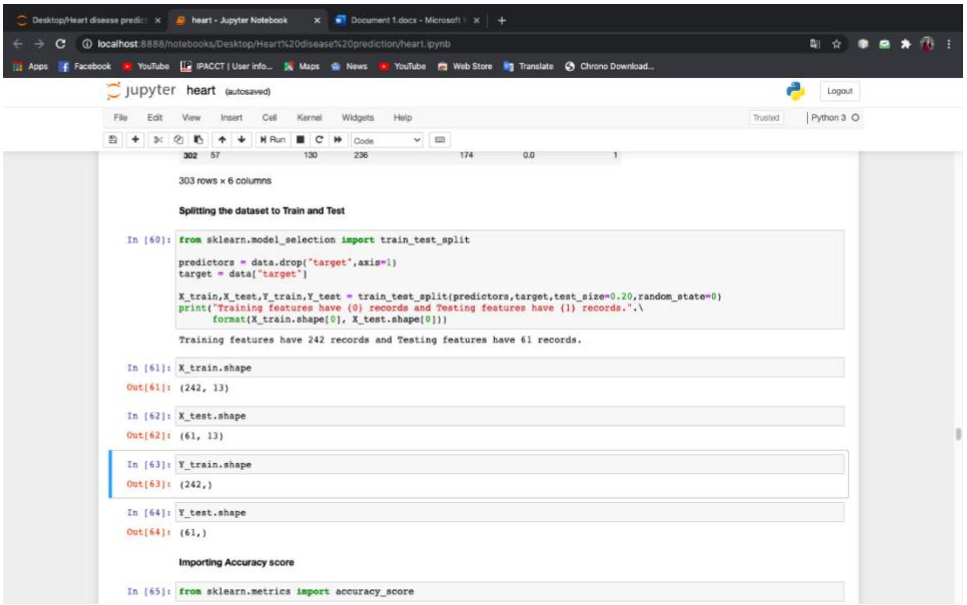
```
In [51]: plt.figure(figsize=(20,10))
         sns.scatterplot(x='cholesterol',y='thalassemia',data=data,hue='target')
         plt.show()
```

10. Correlation plot:


Desktop/Heart disease predict
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
Apps
Facebook
YouTube
IPACCT | User info...
Maps
News
YouTube
Web Store
Translate
Chrono Download...
jupyter heart (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
age 1 0.28 0.21 -0.4 0.21 0.28
resting_blood_pressure 0.28 1 0.12 -0.047 0.19 0.1
cholesterol 0.21 0.12 1 -0.0099 0.054 0.071
max_heart_rate_achieved -0.4 -0.047 -0.0099 1 -0.34 -0.21
st_depression 0.21 0.19 0.054 -0.34 1 0.22
num_major_vessels 0.28 0.1 0.071 -0.21 0.22 1
age
resting_blood_pressure
cholesterol
max_heart_rate_achieved
st_depression
num_major_vessels
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
         cnames=['age','resting_blood_pressure','cholesterol','max_heart_rate_achieved','st_depression','num_major_vessels']

In [58]: #Set the width and height of the plot
         f, ax = plt.subplots(figsize=(7, 5))

         #Correlation plot
         df_corr = data.loc[:,cnames]
         #Generate correlation matrix
         corr = df_corr.corr()

         #Plot using seaborn library
         sns.heatmap(corr, annot = True, cmap='coolwarm',linewidths=.1)
         plt.show()
```

11. Splitting the dataset to Train and Test:



```
302  57    130  236    174    0.0    1
```

303 rows × 6 columns

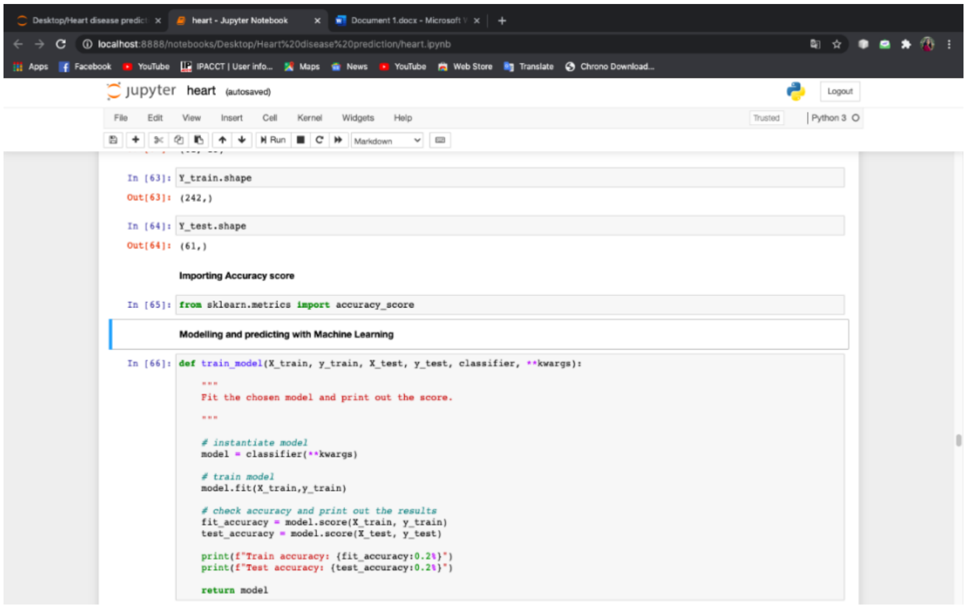
Splitting the dataset to Train and Test

```
In [60]: from sklearn.model_selection import train_test_split

         predictors = data.drop("target",axis=1)
         target = data["target"]

         X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_size=0.20,random_state=0)
         print("Training features have {0} records and Testing features have {1} records.".\
               format(X_train.shape[0], X_test.shape[0]))

         Training features have 242 records and Testing features have 61 records.

In [61]: X_train.shape

Out[61]: (242, 13)

In [62]: X_test.shape

Out[62]: (61, 13)

In [63]: Y_train.shape

Out[63]: (242,)

In [64]: Y_test.shape

Out[64]: (61,)
```

Importing Accuracy score

```
In [65]: from sklearn.metrics import accuracy_score
```

12. Modelling and predicting with Machine Learning:



```
In [63]: Y_train.shape

Out[63]: (242,)

In [64]: Y_test.shape

Out[64]: (61,)
```

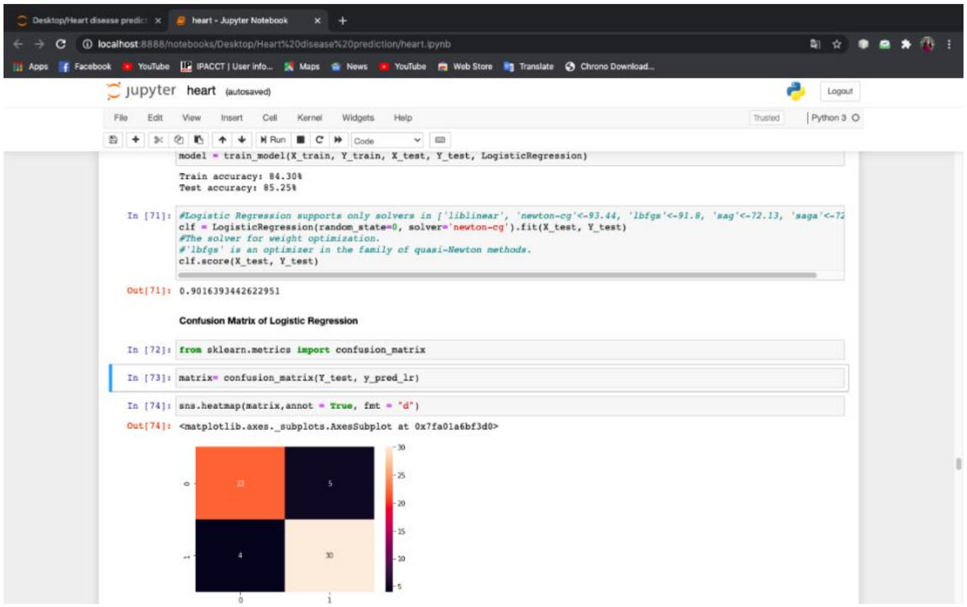
Importing Accuracy score

```
In [65]: from sklearn.metrics import accuracy_score
```

Modelling and predicting with Machine Learning

```
In [66]: def train_model(X_train, y_train, X_test, y_test, classifier, **kwargs):

             """
             Fit the chosen model and print out the score.

             """

             # instantiate model
             model = classifier(**kwargs)

             # train model
             model.fit(X_train,y_train)

             # check accuracy and print out the results
             fit_accuracy = model.score(X_train, y_train)
             test_accuracy = model.score(X_test, y_test)

             print(f"Train accuracy: {fit_accuracy:0.2%}")
             print(f"Test accuracy: {test_accuracy:0.2%}")

             return model
```

13. Confusion Matrix of Logistic Regression:



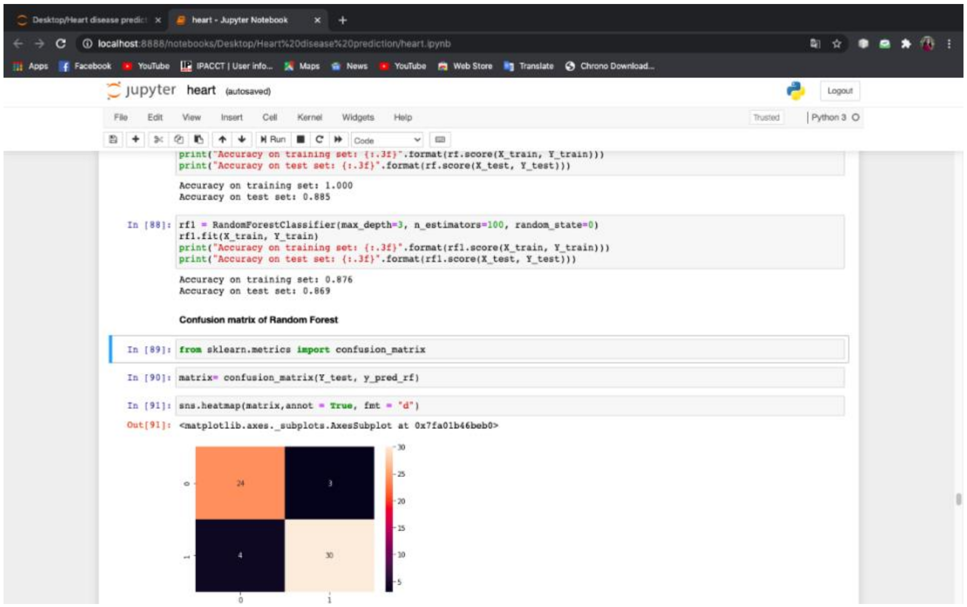
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
jupyter heart (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
22
5
4
30
0
1
0
1
30
25
20
15
10
5

```
model = train_model(X_train, Y_train, X_test, Y_test, LogisticRegression)

Train accuracy: 84.30%
Test accuracy: 85.25%

In [71]: #Logistic Regression supports only solvers in ['liblinear', 'newton-cg'<-93.44, 'lbfgs'<-91.8, 'sag'<-72.13, 'saga'<-72
         clf = LogisticRegression(random_state=0, solver='newton-cg').fit(X_test, Y_test)
         #The solver for weight optimization.
         #'lbfgs' is an optimizer in the family of quasi-Newton methods.
         clf.score(X_test, Y_test)

Out[71]: 0.9016393442622951
```

Confusion Matrix of Logistic Regression

```
In [72]: from sklearn.metrics import confusion_matrix

In [73]: matrix= confusion_matrix(Y_test, y_pred_lr)

In [74]: sns.heatmap(matrix,annot = True, fmt = "d")
Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01a6bf3d0>
```

14. Confusion Matrix of Random forest:



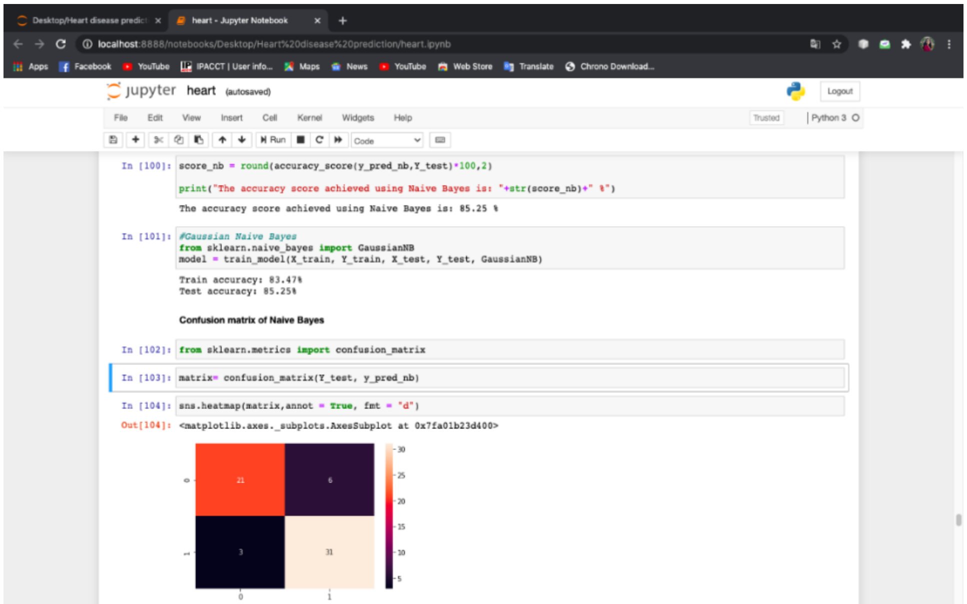
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
jupyter heart (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
24
3
4
30
0
1
0
1
30
25
20
15
10
5

```
print("Accuracy on training set: {:.3f}".format(rf.score(X_train, Y_train)))
print("Accuracy on test set: {:.3f}".format(rf.score(X_test, Y_test)))

Accuracy on training set: 1.000
Accuracy on test set: 0.885

In [88]: rf1 = RandomForestClassifier(max_depth=3, n_estimators=100, random_state=0)
         rf1.fit(X_train, Y_train)
         print("Accuracy on training set: {:.3f}".format(rf1.score(X_train, Y_train)))
         print("Accuracy on test set: {:.3f}".format(rf1.score(X_test, Y_test)))

Accuracy on training set: 0.876
Accuracy on test set: 0.869
```

Confusion matrix of Random Forest

```
In [89]: from sklearn.metrics import confusion_matrix

In [90]: matrix= confusion_matrix(Y_test, y_pred_rf)

In [91]: sns.heatmap(matrix,annot = True, fmt = "d")
Out[91]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01b46beb0>
```

15. Confusion Matrix of Naive Bayes:

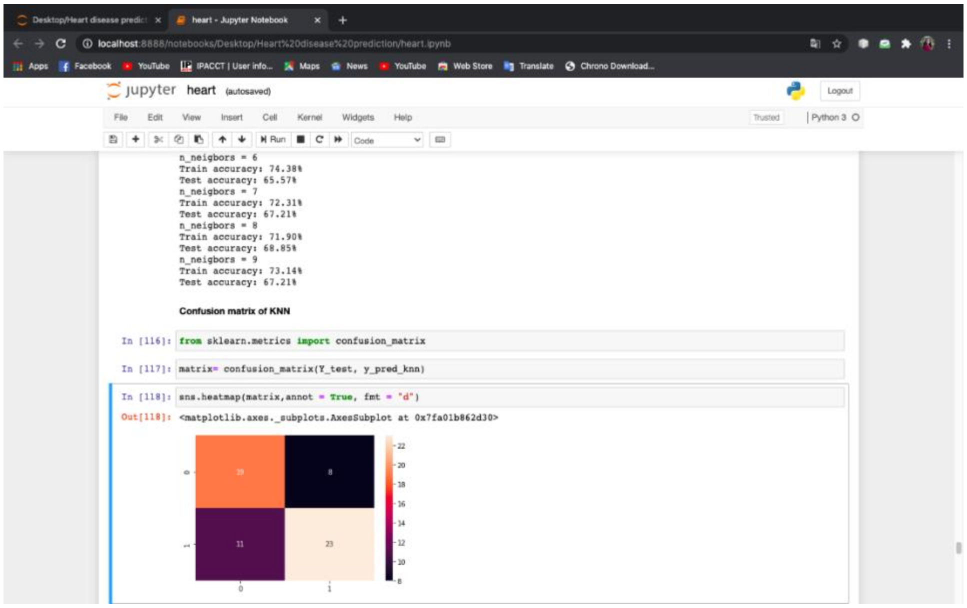


```
In [100]: score_nb = round(accuracy_score(y_pred_nb,Y_test)*100,2)

          print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")

          The accuracy score achieved using Naive Bayes is: 85.25 %

In [101]: #Gaussian Naive Bayes
          from sklearn.naive_bayes import GaussianNB
          model = train_model(X_train, Y_train, X_test, Y_test, GaussianNB)

          Train accuracy: 83.47%
          Test accuracy: 85.25%
```

Confusion matrix of Naive Bayes

```
In [102]: from sklearn.metrics import confusion_matrix

In [103]: matrix= confusion_matrix(Y_test, y_pred_nb)

In [104]: sns.heatmap(matrix,annot = True, fmt = "d")

Out[104]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01b23d400>
```

16. Confusion Matrix of K Nearest Neighbors:

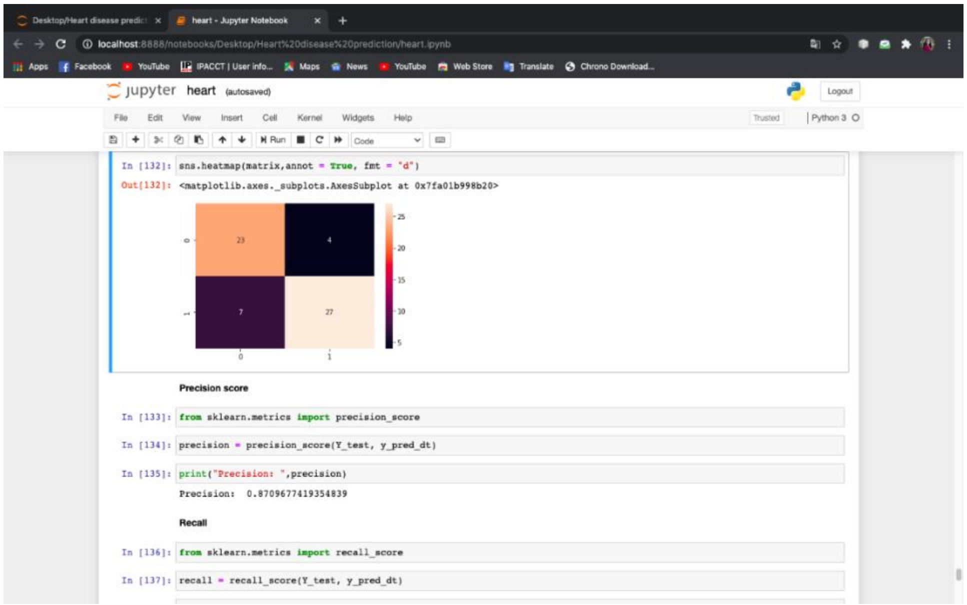


```
n_neigbors = 6
Train accuracy: 74.38%
Test accuracy: 65.57%
n_neigbors = 7
Train accuracy: 72.31%
Test accuracy: 67.21%
n_neigbors = 8
Train accuracy: 71.90%
Test accuracy: 68.85%
n_neigbors = 9
Train accuracy: 73.14%
Test accuracy: 67.21%
```

Confusion matrix of KNN

```
In [116]: from sklearn.metrics import confusion_matrix

In [117]: matrix= confusion_matrix(Y_test, y_pred_knn)

In [118]: sns.heatmap(matrix,annot = True, fmt = "d")

Out[118]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01b862d30>
```

17. Confusion Matrix of Decision Tree:



```
In [132]: sns.heatmap(matrix,annot = True, fmt = "d")
Out[132]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01b998b20>
```

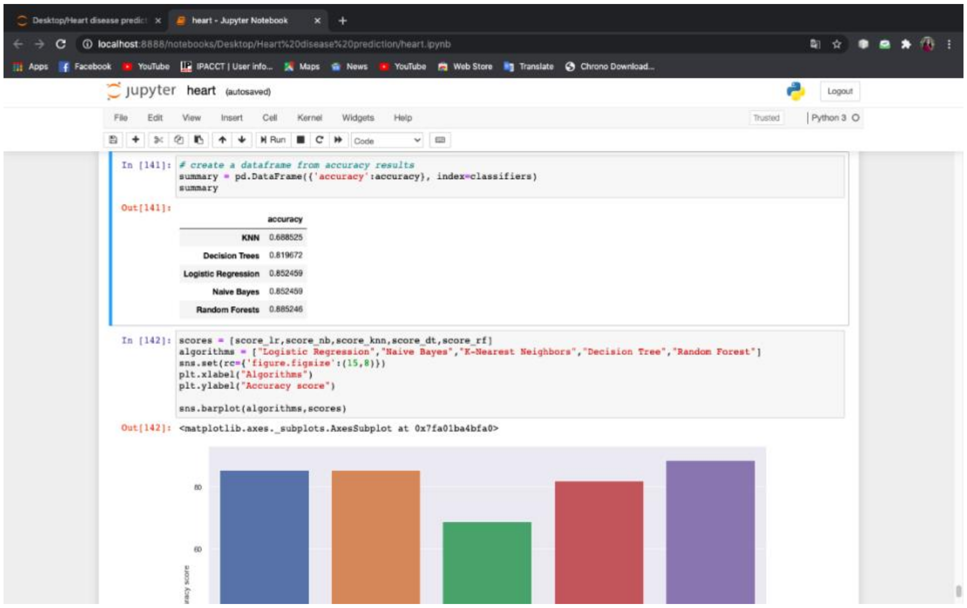
Precision score

```
In [133]: from sklearn.metrics import precision_score

In [134]: precision = precision_score(Y_test, y_pred_dt)

In [135]: print("Precision: ",precision)
          Precision:  0.8709677419354839
```

Recall

```
In [136]: from sklearn.metrics import recall_score

In [137]: recall = recall_score(Y_test, y_pred_dt)
```

FINAL OUTPUT-



Desktop/Heart disease predict
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
Apps Facebook YouTube IPACCT | User info... Maps News YouTube Web Store Translate Chrono Download...
jupyter heart (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

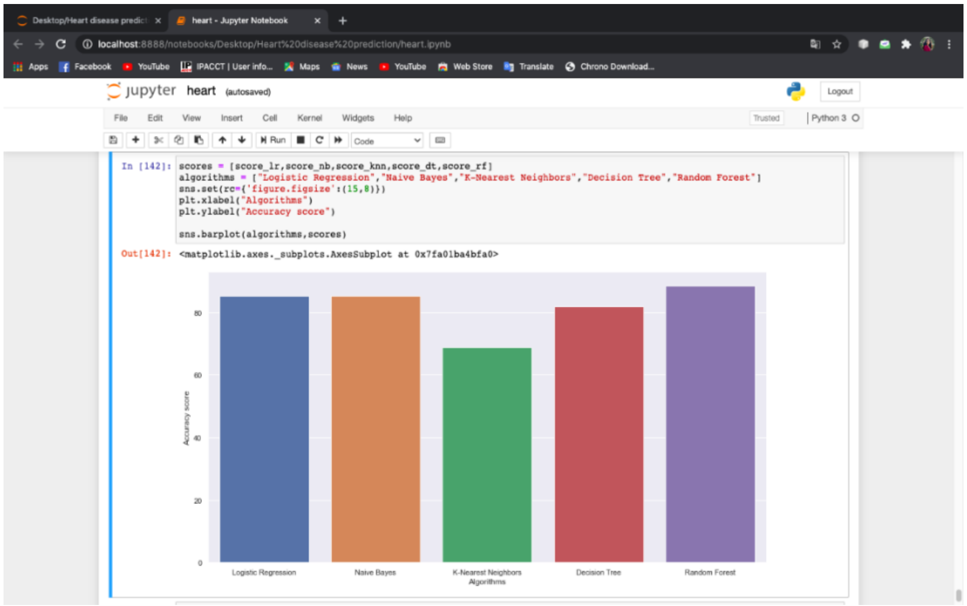
```
In [141]: # create a dataframe from accuracy results
          summary = pd.DataFrame({'accuracy':accuracy}, index=classifiers)
          summary
```

Out[141]:

	accuracy
KNN	0.688525
Decision Trees	0.819672
Logistic Regression	0.852459
Naive Bayes	0.852459
Random Forests	0.885246

```
In [142]: scores = [score_lr,score_nb,score_knn,score_dt,score_rf]
          algorithms = ["Logistic Regression","Naive Bayes","K-Nearest Neighbors","Decision Tree","Random Forest"]
          sns.set(rc={'figure.figsize':(15,8)})
          plt.xlabel("Algorithms")
          plt.ylabel("Accuracy score")

          sns.barplot(algorithms,scores)
```

```
Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01ba4bfa0>
```



Desktop/Heart disease predict
heart - Jupyter Notebook
localhost:8888/notebooks/Desktop/Heart%20disease%20prediction/heart.ipynb
Apps Facebook YouTube IPACCT | User info... Maps News YouTube Web Store Translate Chrono Download...
jupyter heart (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [142]: scores = [score_lr,score_nb,score_knn,score_dt,score_rf]
          algorithms = ["Logistic Regression","Naive Bayes","K-Nearest Neighbors","Decision Tree","Random Forest"]
          sns.set(rc={'figure.figsize':(15,8)})
          plt.xlabel("Algorithms")
          plt.ylabel("Accuracy score")

          sns.barplot(algorithms,scores)
```

```
Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa01ba4bfa0>
```

80
60
40
20
0
Accuracy score
Logistic Regression
Naive Bayes
K-Nearest Neighbors
Decision Tree
Random Forest
Algorithms