Internship Report

Heart Disease prediction

Name: Ankita Gupta

Course: ML1119

Duration: 2 Weeks

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

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has python 3.6 installed on it. you can refer to this url https://www.python.org/downloads/ to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

Dataset used

The data source used for this project is Heart.csv The heart.csv Data Set contains attributes like: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target.

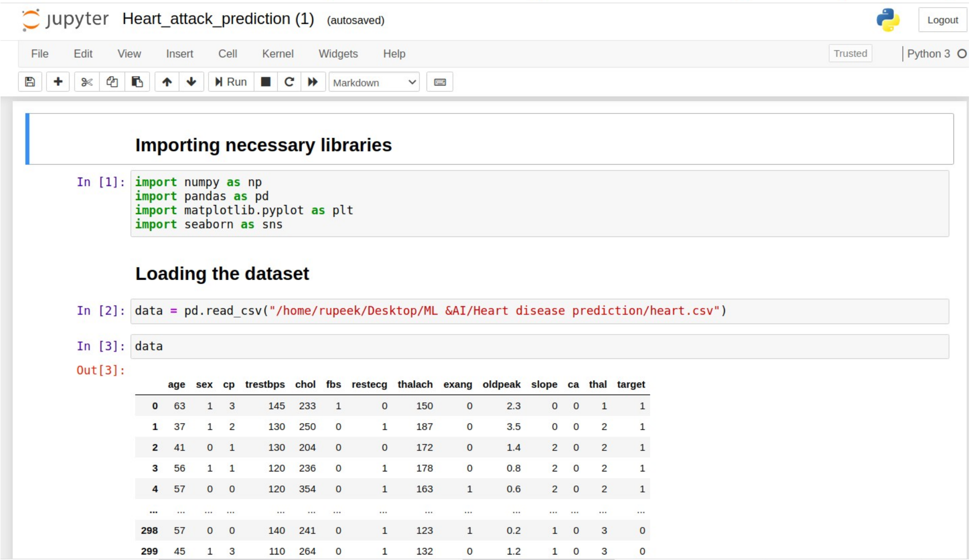
Applying algorithms

K Nearest Neighbors, Random Forest Classifier, Decision Tree Classifier, Support Vector Classifier, Naive Bayes

Accuracy comparision

According to the accuracy graph, random forest works the best.

Importing the libraries:



Importing necessary libraries

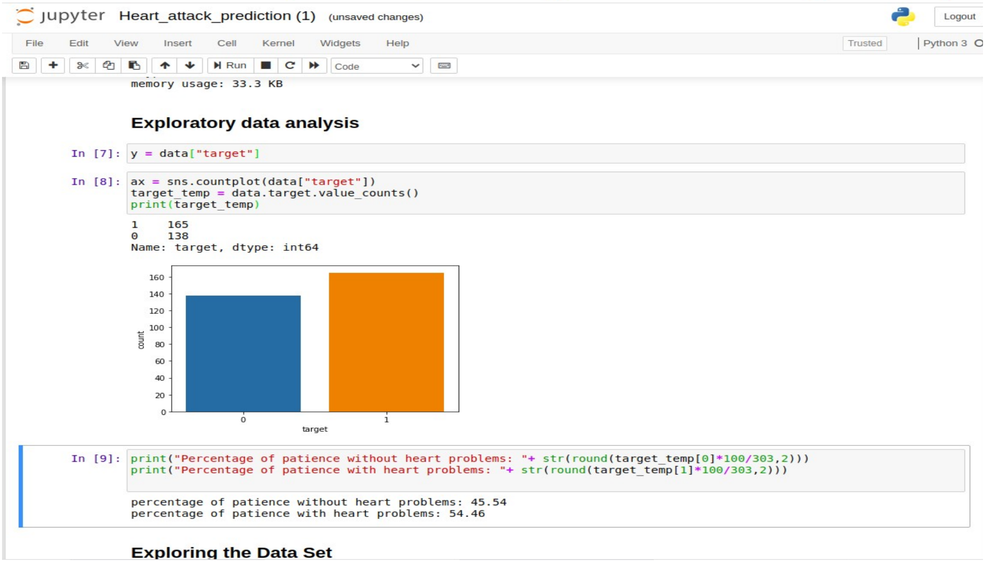
```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Loading the dataset

```
In [2]: data = pd.read_csv("/home/rupeek/Desktop/ML &AI/Heart disease prediction/heart.csv")

In [3]: data
Out[3]:
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0

2. Exploratory Data Analysis (EDA):



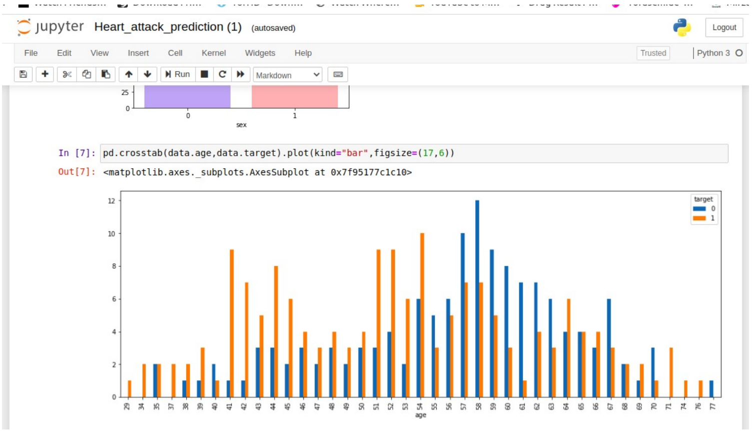
```
memory usage: 33.3 KB
```

Exploratory data analysis

```
In [7]: y = data["target"]

In [8]: ax = sns.countplot(data["target"])
        target_temp = data.target.value_counts()
        print(target_temp)

        1    165
        0    138
        Name: target, dtype: int64

In [9]: 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
```

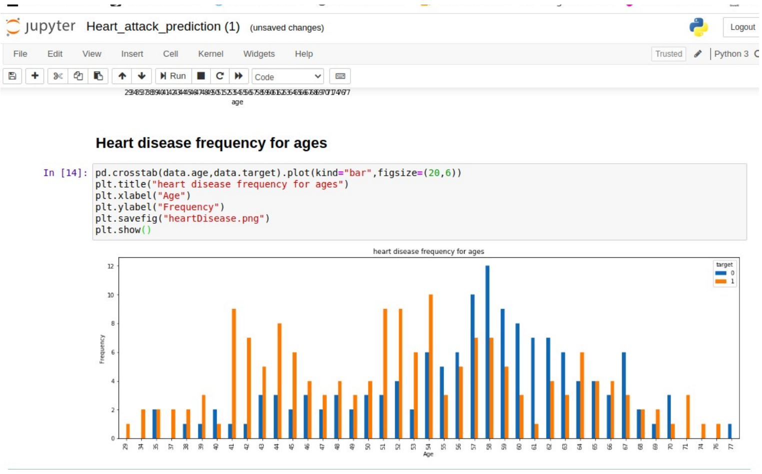
Exploring the Data Set

3. Target v/s age:



```
In [7]: pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(17,6))
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f95177c1c10>
```

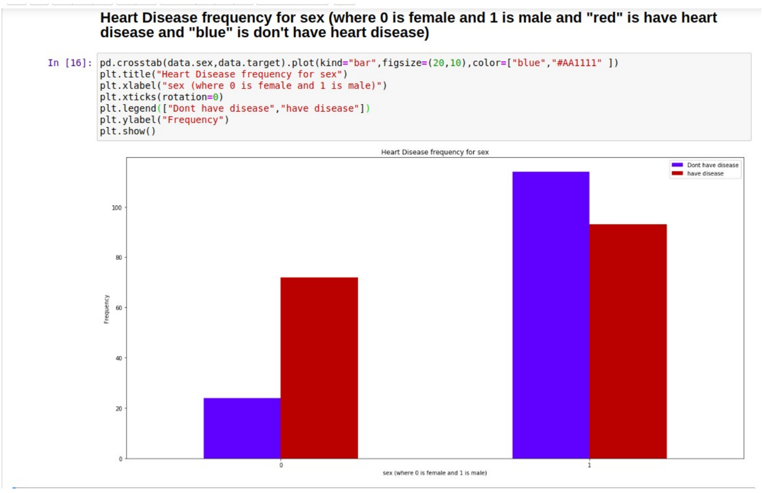
4. Heart Disease Frequency for ages:



Heart disease frequency for ages

```
In [14]: 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("heartDisease.png")
         plt.show()
```

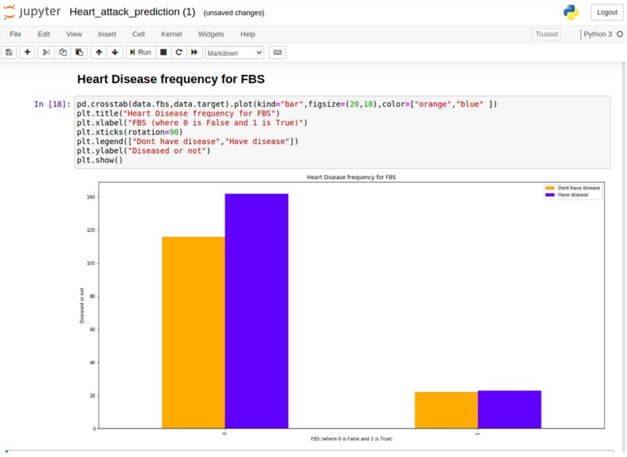
5. Heart Disease frequency for sex (where 0 is female and 1 is male and "red" is have heart disease and "blue" is don't have heart disease):


Heart Disease frequency for sex
Dont have disease
have disease
Frequency
100
80
60
40
20
0
0
1
sex (where 0 is female and 1 is male)

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 [16]: 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 (where 0 is female and 1 is male)")
         plt.xticks(rotation=0)
         plt.legend(["Dont have disease","have disease"])
         plt.ylabel("Frequency")
         plt.show()
```

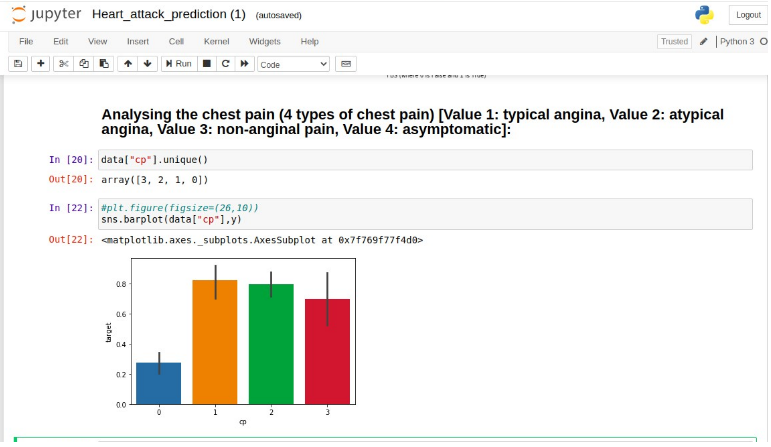
6. Heart disease according to Fasting Blood sugar:



Heart Disease frequency for FBS

```
In [18]: pd.crosstab(data.fbs,data.target).plot(kind="bar",figsize=(20,10),color=["orange","blue" ])
         plt.title("Heart Disease frequency for FBS")
         plt.xlabel("FBS (where 0 is False and 1 is True)")
         plt.xticks(rotation=90)
         plt.legend(["Dont have disease","Have disease"])
         plt.ylabel("Diseased or not")
         plt.show()
```

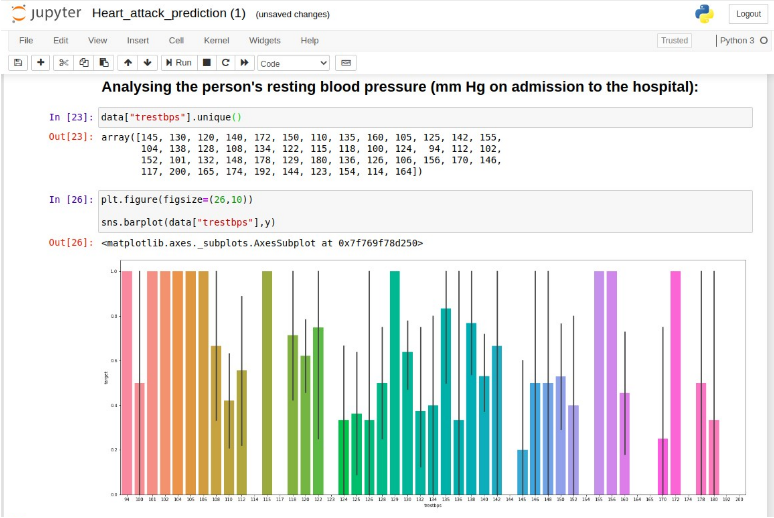
7. Analysing the chest pain (4 types of chest pain) [Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic]:



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 [20]: data["cp"].unique()

Out[20]: array([3, 2, 1, 0])

In [22]: #plt.figure(figsize=(26,10))
         sns.barplot(data["cp"],y)

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769f77f4d0>
```

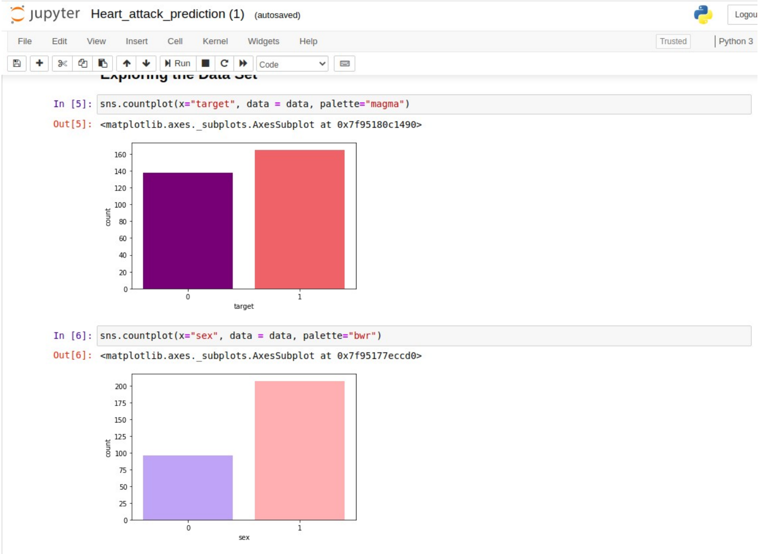
8. Analysing the person's resting blood pressure (mm Hg on admission to the hospital):



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

```
In [23]: data["trestbps"].unique()

Out[23]: 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 [26]: plt.figure(figsize=(26,10))

         sns.barplot(data["trestbps"],y)

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769f78d250>
```

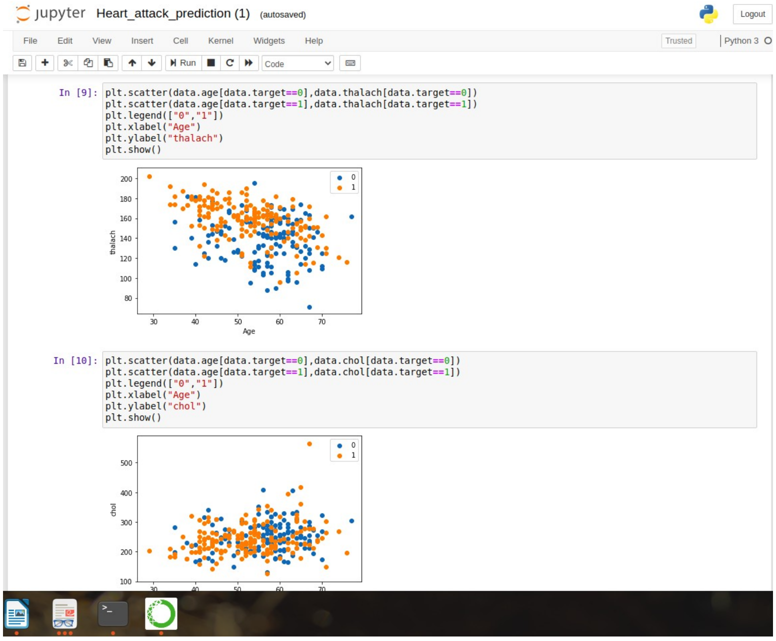
9. Count of Target and sex barplot:



```
In [5]: sns.countplot(x="target", data = data, palette="magma")
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f95180c1490>
```

```
In [6]: sns.countplot(x="sex", data = data, palette="bwr")
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f95177eccd0>
```

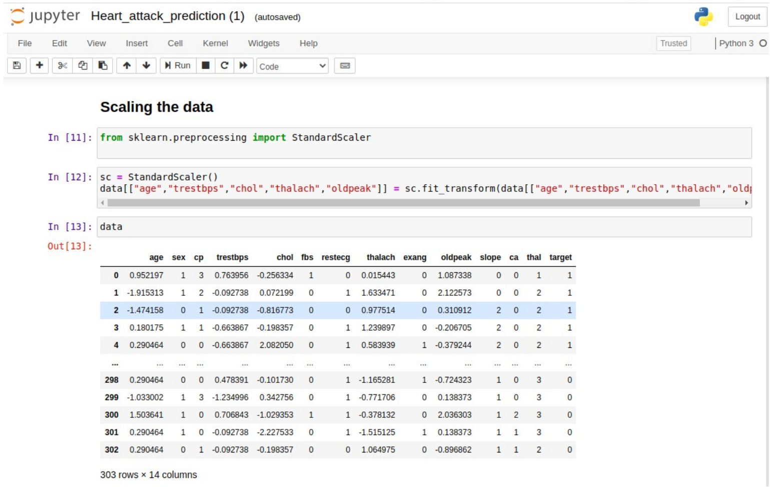
10. Scatter plot of various features:



```
In [9]: plt.scatter(data.age[data.target==0],data.thalach[data.target==0])
        plt.scatter(data.age[data.target==1],data.thalach[data.target==1])
        plt.legend(["0","1"])
        plt.xlabel("Age")
        plt.ylabel("thalach")
        plt.show()
```

```
In [10]: plt.scatter(data.age[data.target==0],data.chol[data.target==0])
         plt.scatter(data.age[data.target==1],data.chol[data.target==1])
         plt.legend(["0","1"])
         plt.xlabel("Age")
         plt.ylabel("chol")
         plt.show()
```

11. Scaling the data



Scaling the data

```
In [11]: from sklearn.preprocessing import StandardScaler
```

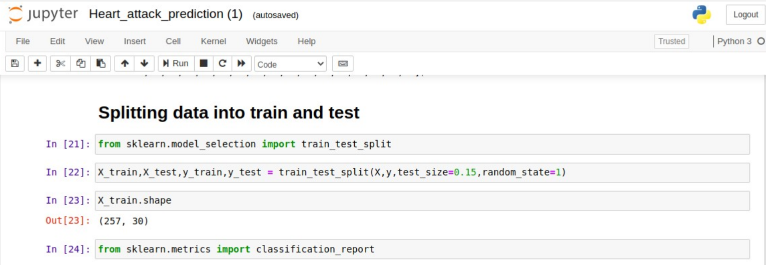
```
In [12]: sc = StandardScaler()
         data[["age","trestbps","chol","thalach","oldpeak"]] = sc.fit_transform(data[["age","trestbps","chol","thalach","oldp
```

```
In [13]: data
Out[13]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0.952197	1	3	0.763956	-0.256334	1	0	0.015443	0	1.087338	0	0	1	1
1	-1.915313	1	2	-0.092738	0.072199	0	1	1.633471	0	2.122573	0	0	2	1
2	-1.474158	0	1	-0.092738	-0.816773	0	0	0.977514	0	0.310912	2	0	2	1
3	0.180175	1	1	-0.663867	-0.198357	0	1	1.239897	0	-0.206705	2	0	2	1
4	0.290464	0	0	-0.663867	2.082050	0	1	0.583939	1	-0.379244	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	0.290464	0	0	0.478391	-0.101730	0	1	-1.165281	1	-0.724323	1	0	3	0
299	-1.033002	1	3	-1.234996	0.342756	0	1	-0.771706	0	0.138373	1	0	3	0
300	1.503641	1	0	0.706843	-1.029353	1	1	-0.378132	0	2.036303	1	2	3	0
301	0.290464	1	0	-0.092738	-2.227533	0	1	-1.515125	1	0.138373	1	1	3	0
302	0.290464	0	1	-0.092738	-0.198357	0	0	1.064975	0	-0.896862	1	1	2	0

303 rows × 14 columns

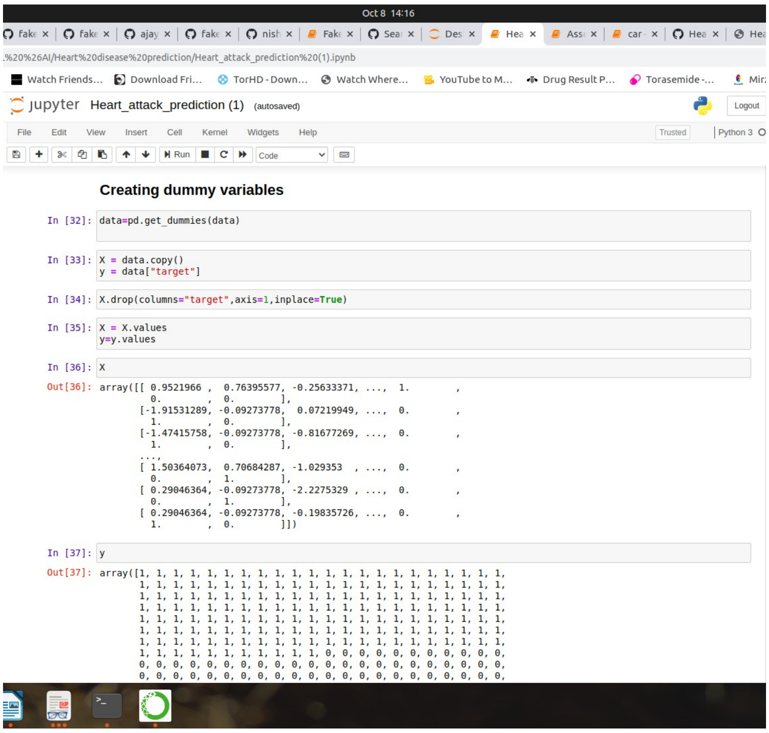
12. Splitting the dataset to Train and Test:


jupyter Heart_attack_prediction (1) (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Splitting data into train and test

```
In [21]: from sklearn.model_selection import train_test_split

In [22]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.15,random_state=1)

In [23]: X_train.shape
Out[23]: (257, 30)

In [24]: from sklearn.metrics import classification_report
```

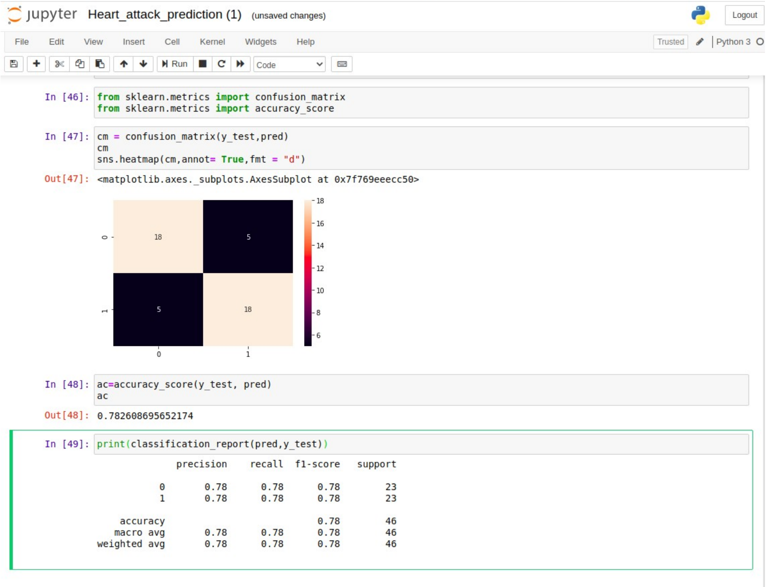
13. Creating dummies:



Creating dummy variables

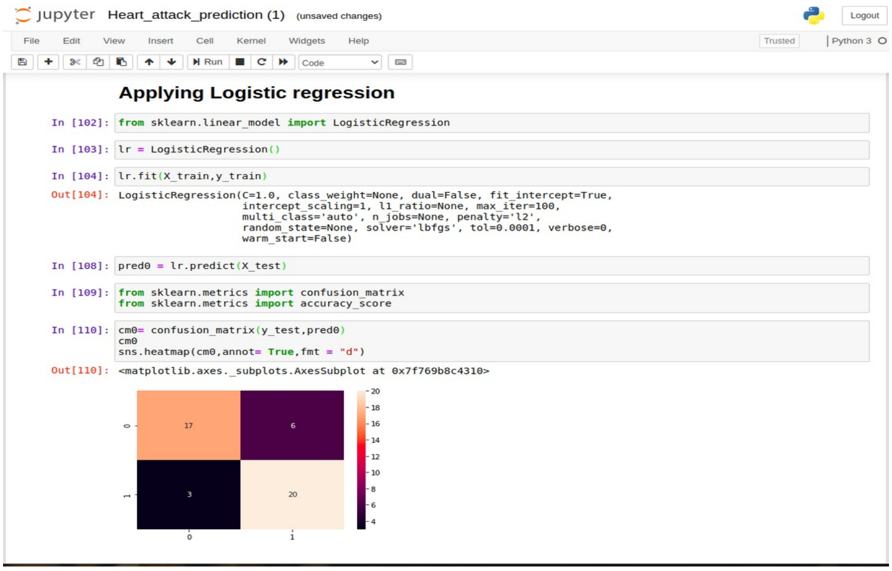
```
In [32]: data=pd.get_dummies(data)

In [33]: X = data.copy()
         y = data["target"]

In [34]: X.drop(columns="target",axis=1,inplace=True)

In [35]: X = X.values
         y=y.values

In [36]: X
Out[36]: array([[ 0.9521966 ,  0.76395577, -0.25633371, ...,  1.        ,
                  0.        ,  0.        ],
                [-1.91531289, -0.09273778,  0.07219949, ...,  0.        ,
                  1.        ,  0.        ],
                [-1.47415758, -0.09273778, -0.81677269, ...,  0.        ,
                  1.        ,  0.        ],
                ...,
                [ 1.50364073,  0.70684287, -1.029353  , ...,  0.        ,
                  0.        ,  1.        ],
                [ 0.29046364, -0.09273778, -2.2275329 , ...,  0.        ,
                  0.        ,  1.        ],
                [ 0.29046364, -0.09273778, -0.19835726, ...,  0.        ,
                  1.        ,  0.        ]])

In [37]: y
Out[37]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

14. Confusion Matrix of KNN:


jupyter Heart_attack_prediction (1) (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
0
1
0
1
18
5
5
18
18
16
14
12
10
8
6

```
In [46]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score

In [47]: cm = confusion_matrix(y_test,pred)
         cm
         sns.heatmap(cm,annot= True,fmt = "d")
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769eeecc50>

In [48]: ac=accuracy_score(y_test, pred)
         ac
Out[48]: 0.782608695652174

In [49]: print(classification_report(pred,y_test))
                       precision    recall  f1-score   support

                    0       0.78      0.78      0.78        23
                    1       0.78      0.78      0.78        23

             accuracy                           0.78        46
            macro avg       0.78      0.78      0.78        46
         weighted avg       0.78      0.78      0.78        46
```

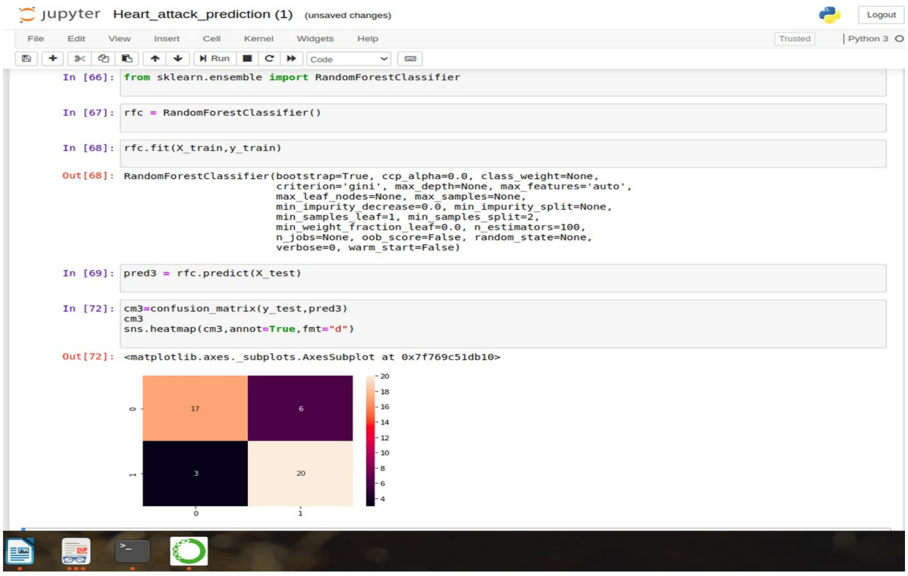
15.Confusion Matrix of Logistic Regression:



Applying Logistic regression

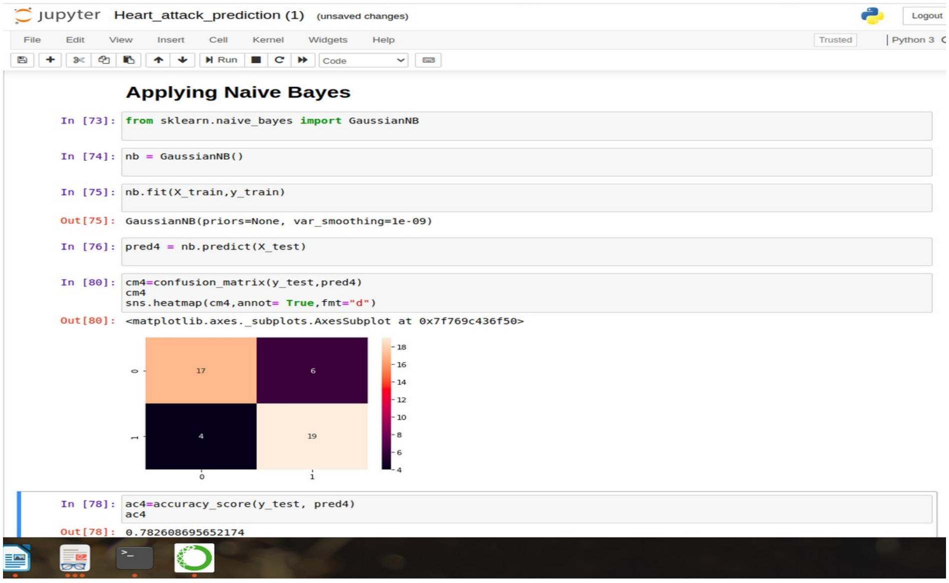
```
In [102]: from sklearn.linear_model import LogisticRegression

In [103]: lr = LogisticRegression()

In [104]: lr.fit(X_train,y_train)

Out[104]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=100,
                             multi_class='auto', n_jobs=None, penalty='l2',
                             random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                             warm_start=False)

In [108]: pred0 = lr.predict(X_test)

In [109]: from sklearn.metrics import confusion_matrix
          from sklearn.metrics import accuracy_score

In [110]: cm0= confusion_matrix(y_test,pred0)
          cm0
          sns.heatmap(cm0,annot= True,fmt = "d")

Out[110]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769b8c4310>
```

16. Confusion Matrix of Random forest:



```
In [66]: from sklearn.ensemble import RandomForestClassifier

In [67]: rfc = RandomForestClassifier()

In [68]: rfc.fit(X_train,y_train)

Out[68]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)

In [69]: pred3 = rfc.predict(X_test)

In [72]: cm3=confusion_matrix(y_test,pred3)
         cm3
         sns.heatmap(cm3,annot=True,fmt="d")

Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769c51db10>
```

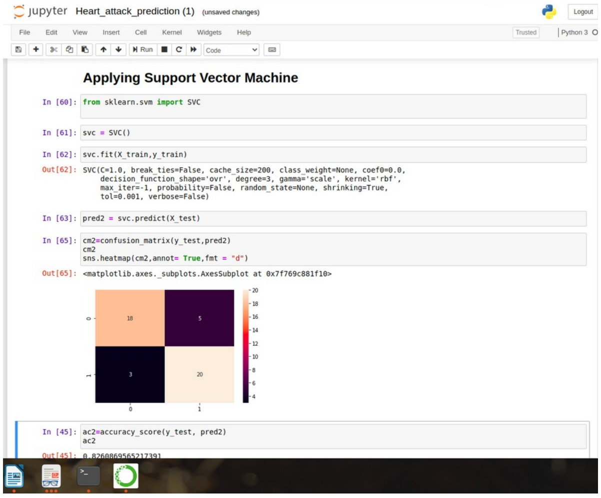
17. Confusion Matrix of Naive Bayes:



Applying Naive Bayes

```
In [73]: from sklearn.naive_bayes import GaussianNB

In [74]: nb = GaussianNB()

In [75]: nb.fit(X_train,y_train)

Out[75]: GaussianNB(priors=None, var_smoothing=1e-09)

In [76]: pred4 = nb.predict(X_test)

In [80]: cm4=confusion_matrix(y_test,pred4)
         cm4
         sns.heatmap(cm4,annot= True,fmt="d")

Out[80]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769c436f50>

In [78]: ac4=accuracy_score(y_test, pred4)
         ac4

Out[78]: 0.782608695652174
```

16. Confusion Matrix of SVM:



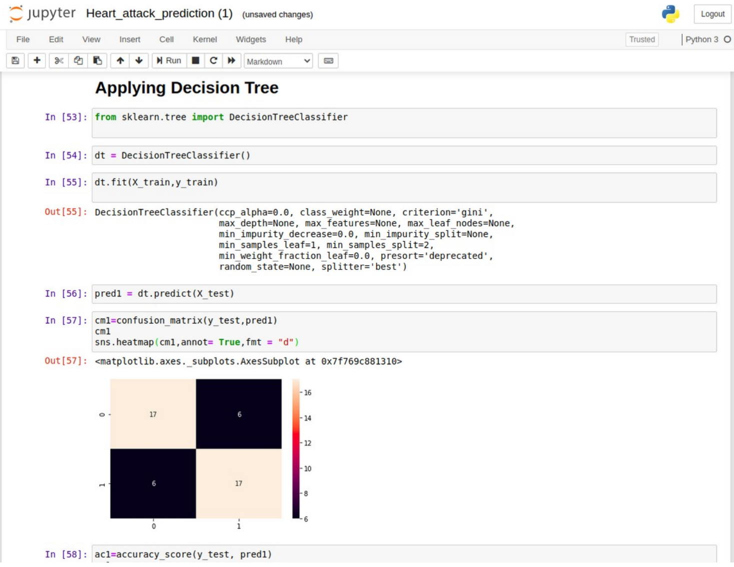
jupyter Heart_attack_prediction (1) (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Applying Support Vector Machine

```
In [60]: from sklearn.svm import SVC

In [61]: svc = SVC()

In [62]: svc.fit(X_train,y_train)

Out[62]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)

In [63]: pred2 = svc.predict(X_test)

In [65]: cm2=confusion_matrix(y_test,pred2)
         cm2
         sns.heatmap(cm2,annot= True,fmt = "d")

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769c881f10>
```

```
In [45]: ac2=accuracy_score(y_test, pred2)
         ac2

Out[45]: 0.8260869565217391
```

17. Confusion Matrix of Decision Tree:



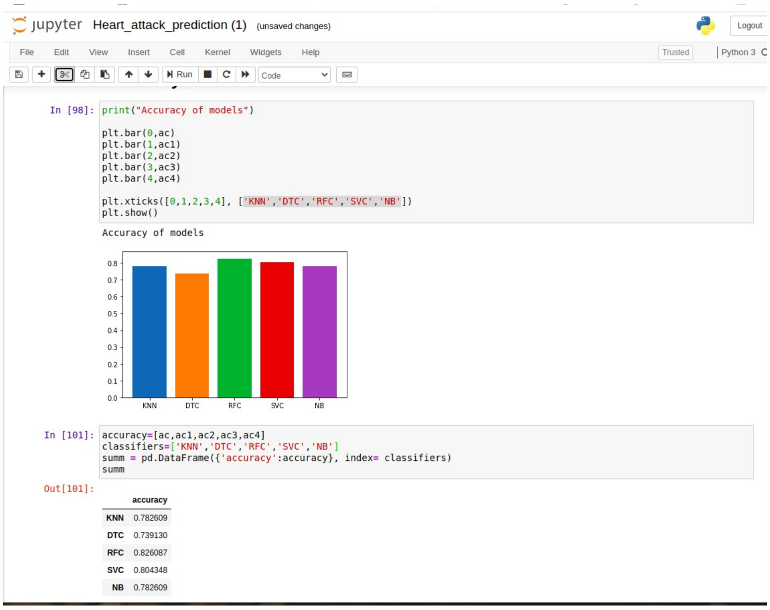
jupyter Heart_attack_prediction (1) (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Applying Decision Tree

```
In [53]: from sklearn.tree import DecisionTreeClassifier

In [54]: dt = DecisionTreeClassifier()

In [55]: dt.fit(X_train,y_train)

Out[55]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')

In [56]: pred1 = dt.predict(X_test)

In [57]: cm1=confusion_matrix(y_test,pred1)
         cm1
         sns.heatmap(cm1,annot= True,fmt = "d")

Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x7f769c881310>
```

```
In [58]: ac1=accuracy_score(y_test, pred1)
```

FINAL OUTPUT-



jupyter Heart_attack_prediction (1) (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [98]: print("Accuracy of models")

         plt.bar(0,ac)
         plt.bar(1,ac1)
         plt.bar(2,ac2)
         plt.bar(3,ac3)
         plt.bar(4,ac4)

         plt.xticks([0,1,2,3,4], ['KNN','DTC','RFC','SVC','NB'])
         plt.show()

         Accuracy of models
```

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
KNN
DTC
RFC
SVC
NB

```
In [101]: accuracy=[ac,ac1,ac2,ac3,ac4]
          classifiers=['KNN','DTC','RFC','SVC','NB']
          summ = pd.DataFrame({'accuracy':accuracy}, index= classifiers)
          summ

Out[101]:
```

	accuracy
KNN	0.782609
DTC	0.739130
RFC	0.826087
SVC	0.804348
NB	0.782609