IDS Project Report

Prediction of Heart Disease

Code Link: Github

Team:

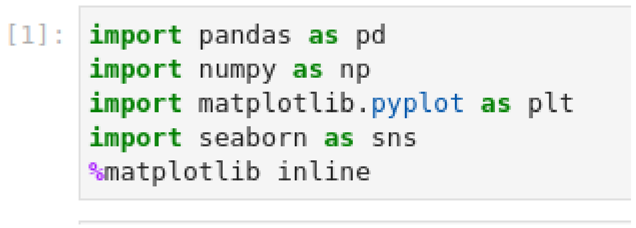
Vani Agarwal	18ucs098
Abhay Singhal	18ucs011
Hritwik Singhal	18ucs055
Sameer Gupta	18ucs008

Project Goal:-

The goal of this project is to visualise the dataset containing factors like age, sex, cholestrol level, and from these factors predict if a person is suffering from heart disease or not.

Plan of Working

1. **Importing Libraries**:- We import libraries like matplotlib, pandas, numpy, seaborn, sklearn. Here, numpy is used for the mathematical part of working with the dataset and matrices. Then pandas is required for performing various kinds of slices and selections which are inbuilt in it.. Matplotlib and seaborn will be used for visualizations, sklearn has important libraries related to applying machine learning algorithms and check for its accuracies.



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

2. **Data Description:-** We import the dataset and give a description of it. Also, we have taken a preprocessed dataset, so we have not found any missing value or null values.
3. **Data Inferences:-** We shall visualise the dataset and get inferences from it. The graphs we shall plot shall help us to see various trends as on what factors does our target variable actually depend on and how other values relate among themselves.
4. After getting inferences, we shall think of **choosing the correct algorithm** to solve the above problem.
5. Finally we have applied our algorithm and see its **result** on the testing dataset and see how well our algorithm has performed and end this project by looking at their confusion matrix and other related things.

Data Description

The dataset is taken from this link. The model we implemented is predictive and supervised. Output variable is binary (heart disease or not) and there are 14 input variables (you can see all of the variables below).

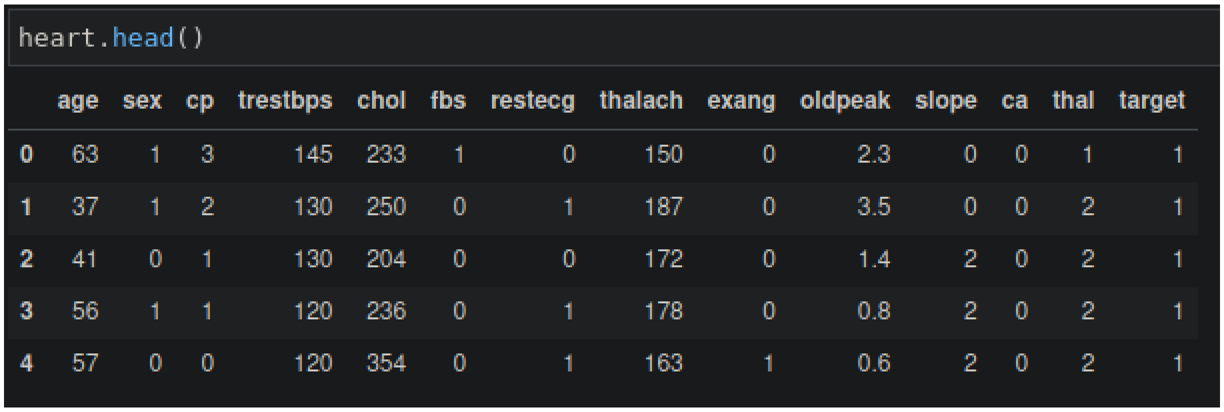
Output variables

1. Target (Heart disease)
   1 = Yes
   0 = no

Input variables

1. age (in years)
2. sex (<u>sex</u>)
   0 = female ,
   1 = male
3. cp: (chest pain type)
   <u>Value 1</u>: typical angina
   <u>Value 2</u>: atypical angina
   <u>Value 3</u>: non-anginal pain
   <u>Value 4</u>: asymptomatic
4. trestbps (resting blood pressure (in mm Hg on admission to the hospital))
5. chol (serum cholesterol in mg/dl)
6. fbs (fasting blood sugar (> 120 mg/dl))
   1 = true
   0 = false
7. restecg (resting electrocardiographic results)
   Value 0: normal,
   Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV),
   Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria)
8. thalach (maximum heart rate achieved)
9. exang (exang: exercise induced angina)
   1 = yes,
   0 = no
10. oldpeak (ST depression induced by exercise relative to rest)
11. slope (the slope of the peak exercise ST segment)
    Value 1: upsloping
    Value 2: flat
    Value 3: downsloping
12. ca (number of major vessels (0-3) colored by fluoroscopy)
13. Thal (Thalassemia: A blood disorder)
    3 = normal;
    6 = fixed defect;
    7 = reversible defect
14. target (the predicted attribute)
    0 = no heart disease ,
    1 = heart disease

A sample of five rows.



```
heart.head()
```

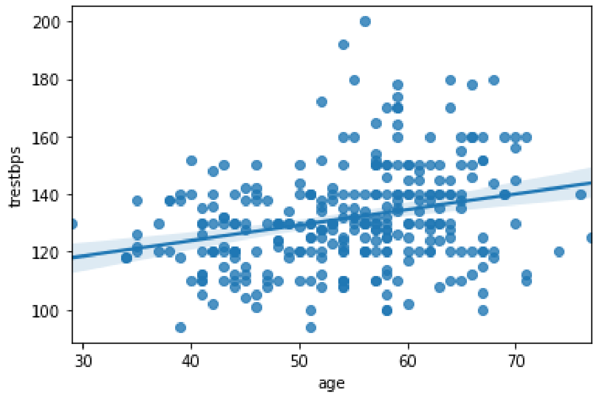
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Inferences from Dataset:

We analyse our dataset and get some inferences from it:-

1. The patients with age range (40,70) have resting blood pressure(trestbps) between 120 to140 mm Hg.

```
[25]: sns.regplot(x="age", y="trestbps", data=heart)

[25]: <matplotlib.axes._subplots.AxesSubplot at 0x25dc5fb5340>
```

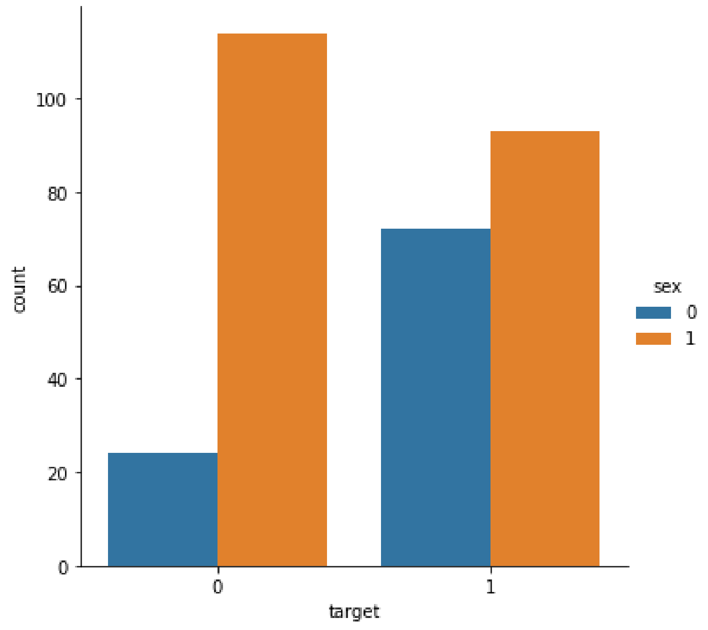


.

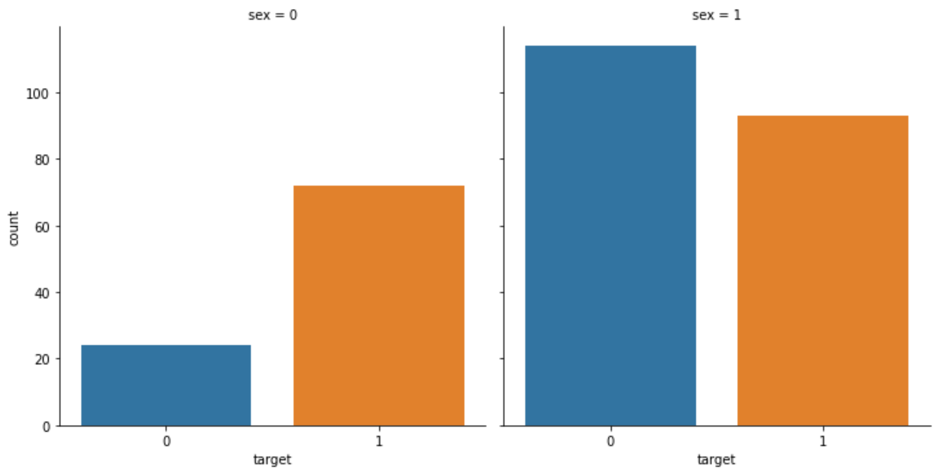
2. The ratio of heart disease / no heart disease is greater for females than males.

```
[13]: sns.catplot(x="target",  hue="sex", kind="count", data=heart)
```

```
[13]: <seaborn.axisgrid.FacetGrid at 0x7feed8521d60>
```

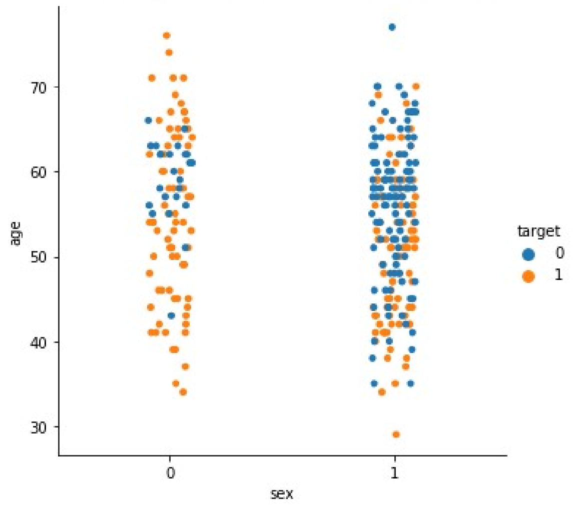


```
[14]: ax = sns.catplot(x="target", col="sex", data=heart, kind="count", height=5, aspect=1)
```

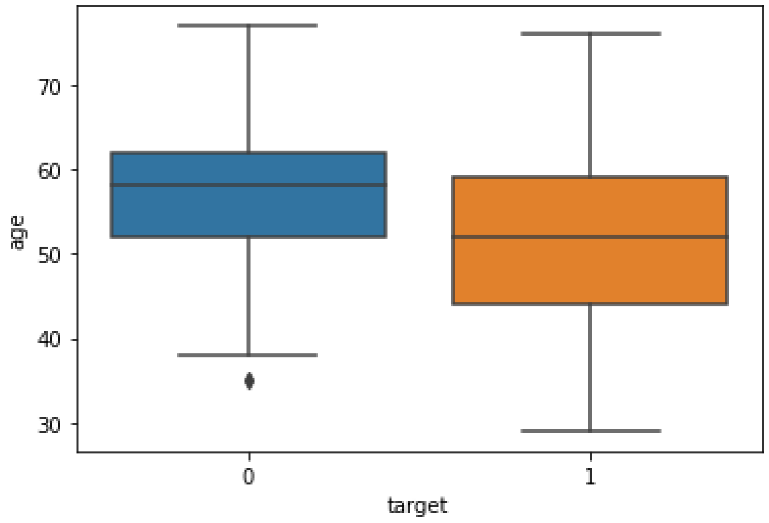


3. Heart disease occurs mostly to patients whose age is greater than 40 years.

```
[15]: sns.catplot(x='sex',y='age',hue='target',data=heart)

[15]: <seaborn.axisgrid.FacetGrid at 0x25dcd401eb0>
```



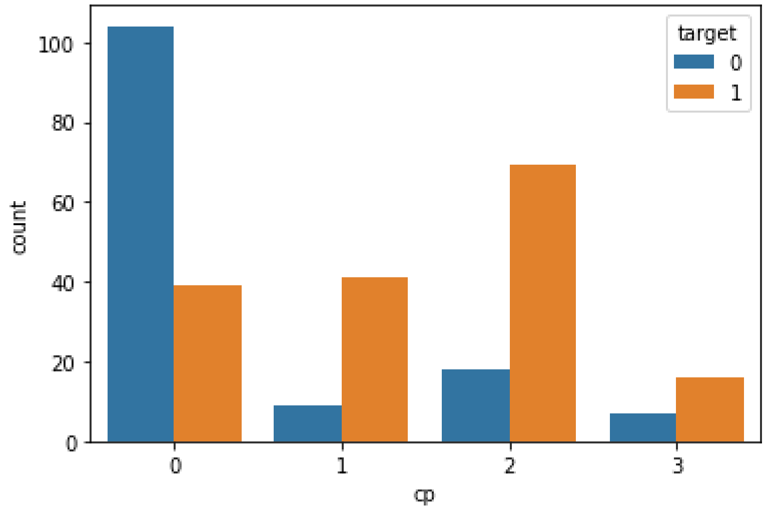
```
[24]: sns.boxplot(x="target", y="age", data=heart)

[24]: <matplotlib.axes._subplots.AxesSubplot at 0x25dc9d0c520>
```



4. The patients are reporting typical angina chest pain in a high number, but number of patients who are suffering from non-anginal pain has high rate of having heart disease

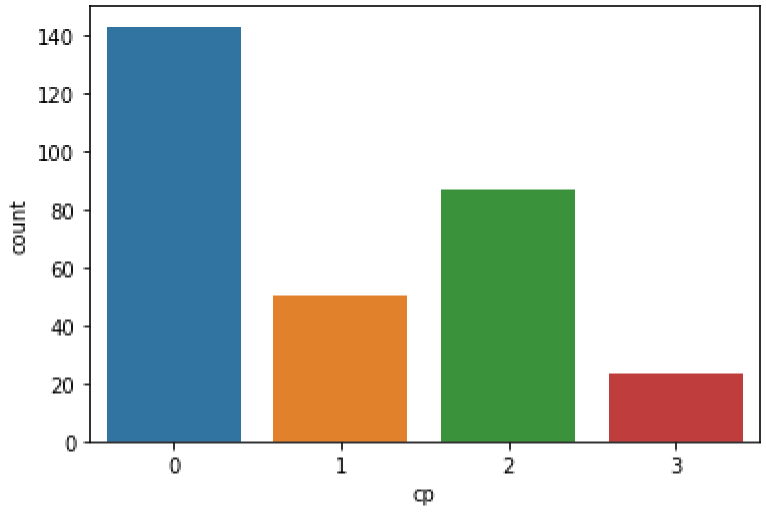
```
[19]: sns.countplot(x="cp", hue="target", data=heart)
```

```
[19]: <matplotlib.axes._subplots.AxesSubplot at 0x25dcd55f070>
```



```
[18]: sns.countplot(x="cp", data=heart)
```

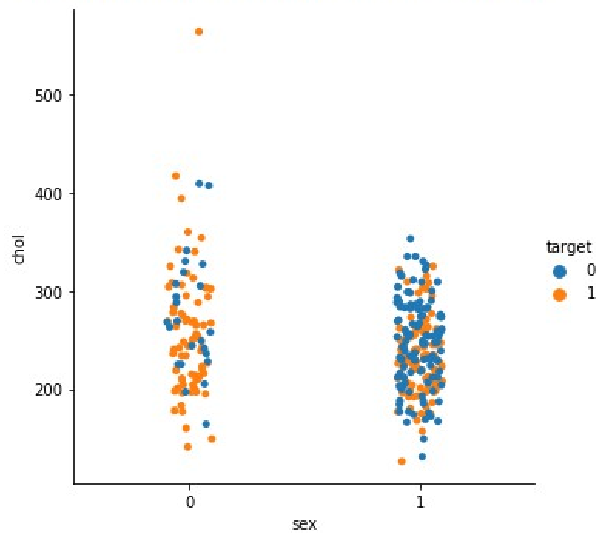
```
[18]: <matplotlib.axes._subplots.AxesSubplot at 0x25dcd50d850>
```



5. Patients with cholesterol higher than 180 have very high rates of heart attack. Further categorizing them on basis of sex, we see that males have much much higher heart attack rates than females in any segment.

```
[16]: sns.catplot(x='sex',y='chol',hue='target',data=heart)
```

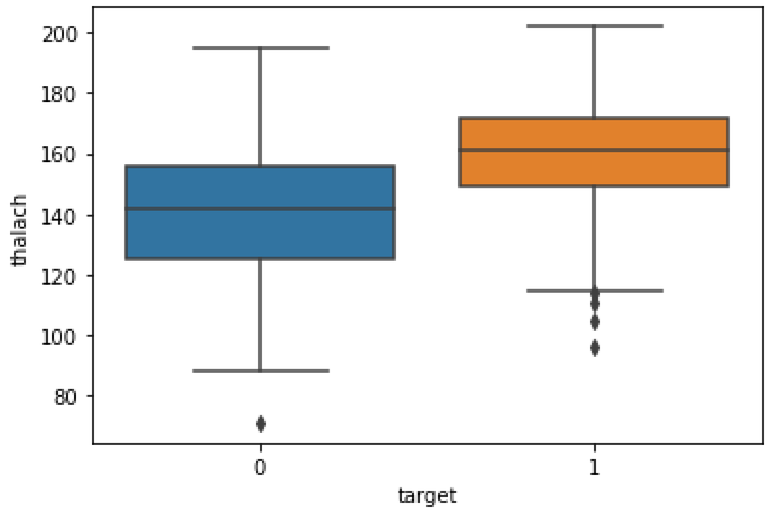
```
[16]: <seaborn.axisgrid.FacetGrid at 0x25dcd459af0>
```



6. The people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

```
[22]: sns.boxplot(x="target", y="thalach", data=heart)
```

```
[22]: <matplotlib.axes._subplots.AxesSubplot at 0x25dcd6b3580>
```

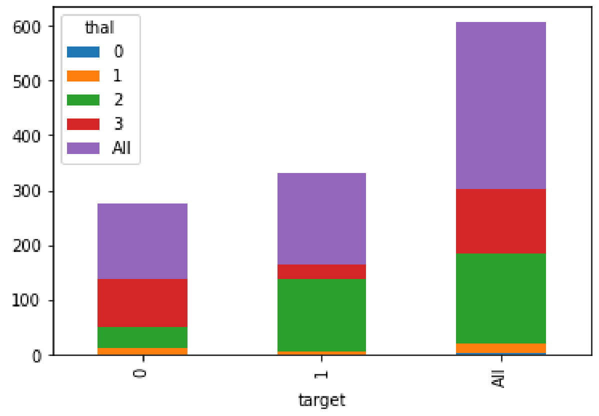


7. People suffering from heart disease generally diagnosed with fixed defect thalassemia.

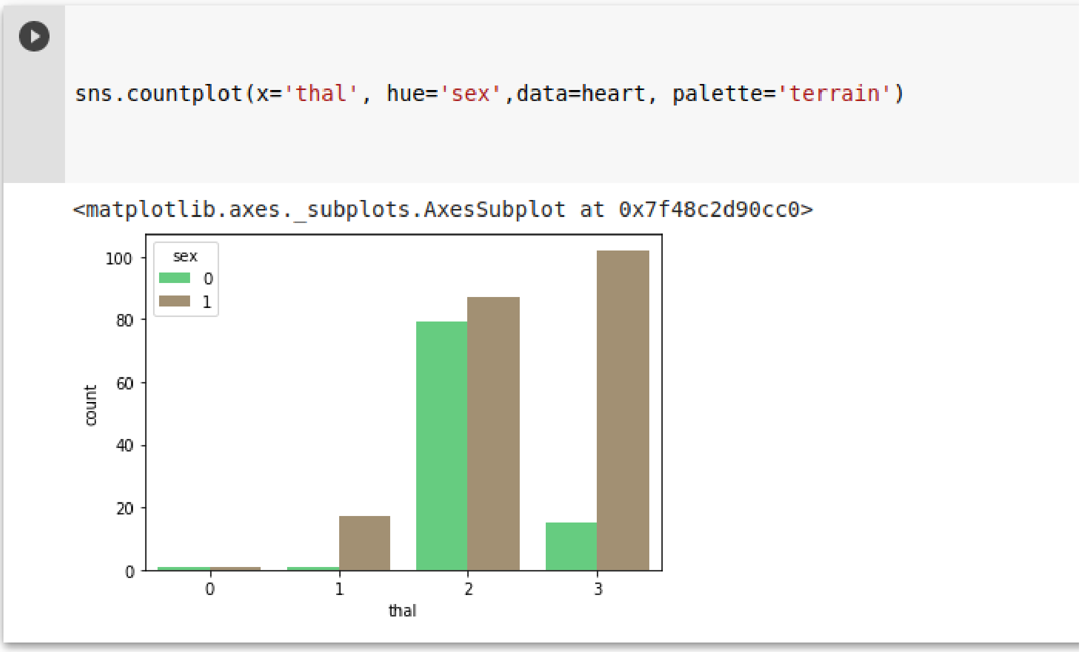
```
[60]
    temp=pd.crosstab(index=heart['target'],
                columns=[heart['thal']],
                margins=True)
    temp
```

thal	0	1	2	3	All
target					
0	1	12	36	89	138
1	1	6	130	28	165
All	2	18	166	117	303

```
temp.plot(kind='bar', stacked=True)
plt.show()
```



8. The point 2 on X-axis shows fixed defect thalassemia and point 3 shows reversible defect thalassemia. So females have fixed defect thalassemia more than any other type. Males have reversible and fixed defect thalassemia more than females and reversible defect thalassemia more than any other type.

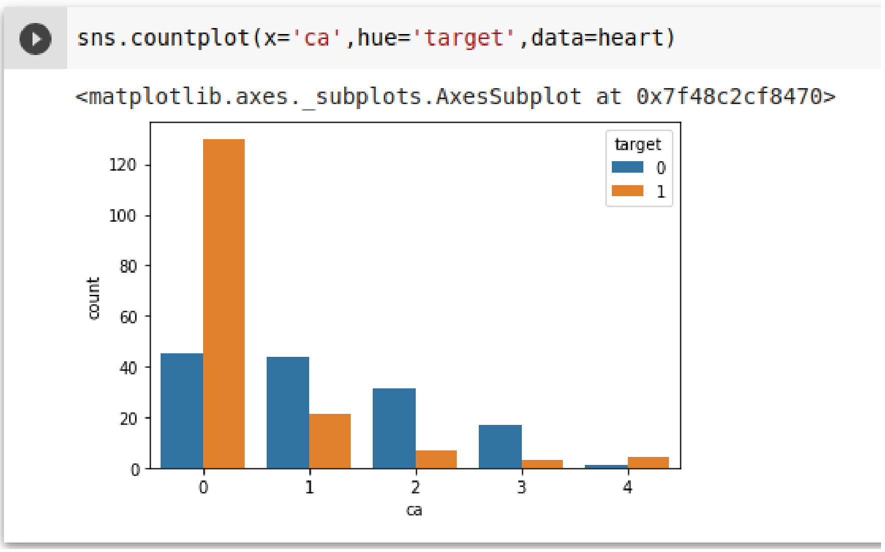


```
sns.countplot(x='thal', hue='sex',data=heart, palette='terrain')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48c2d90cc0>
```

9. People with a lesser number of major vessels have higher chances of suffering from heart disease.

sex

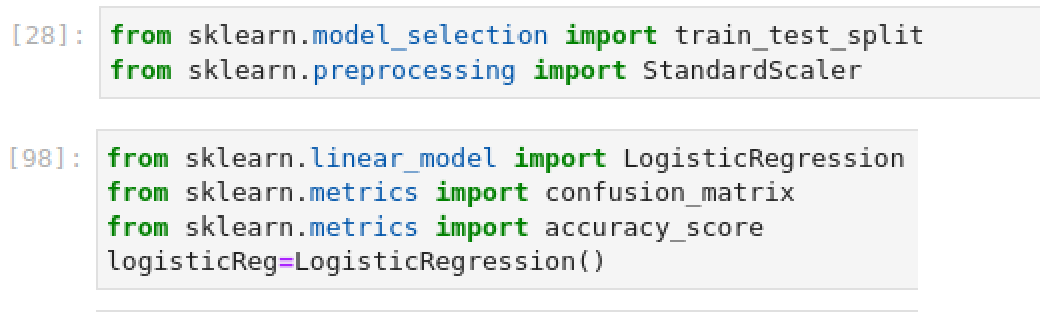


```
sns.countplot(x='ca',hue='target',data=heart)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48c2cf8470>
```

Algorithms

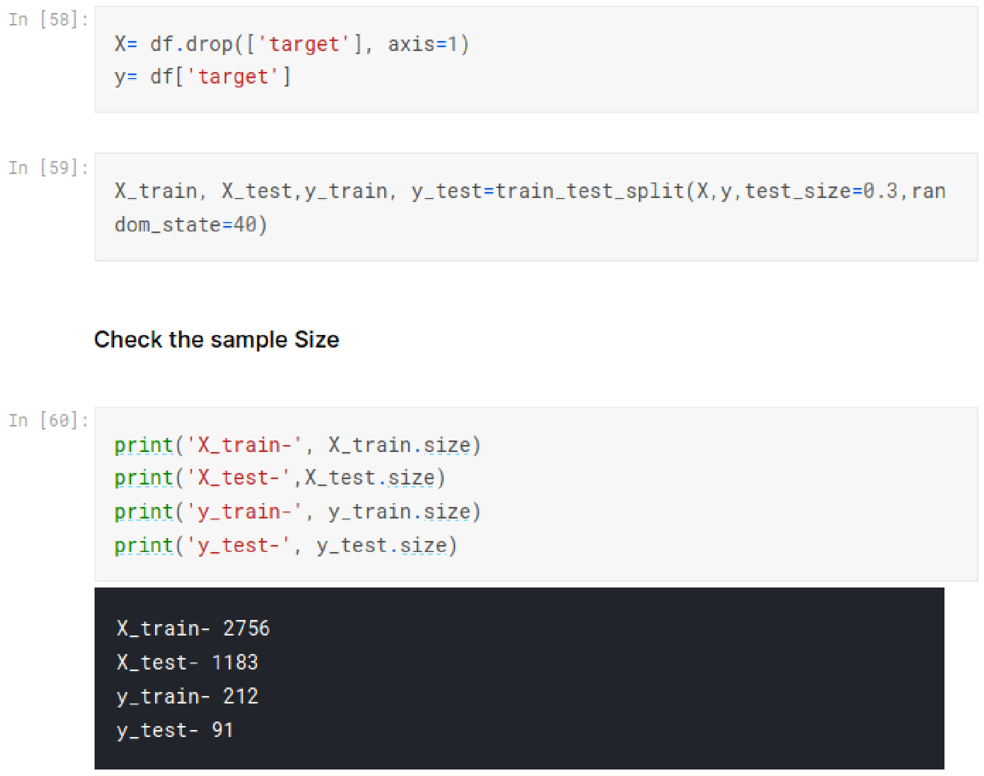
Importing Libraries



```
[28]: from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
```

```
[98]: from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import confusion_matrix
      from sklearn.metrics import accuracy_score
      logisticReg=LogisticRegression()
```

Testing DataSet



```
In [58]: X= df.drop(['target'], axis=1)
         y= df['target']
```

```
In [59]: X_train, X_test,y_train, y_test=train_test_split(X,y,test_size=0.3,ran
         dom_state=40)
```

Check the sample Size

```
In [60]: print('X_train-', X_train.size)
         print('X_test-',X_test.size)
         print('y_train-', y_train.size)
         print('y_test-', y_test.size)
```

```
X_train- 2756
X_test- 1183
y_train- 212
y_test- 91
```

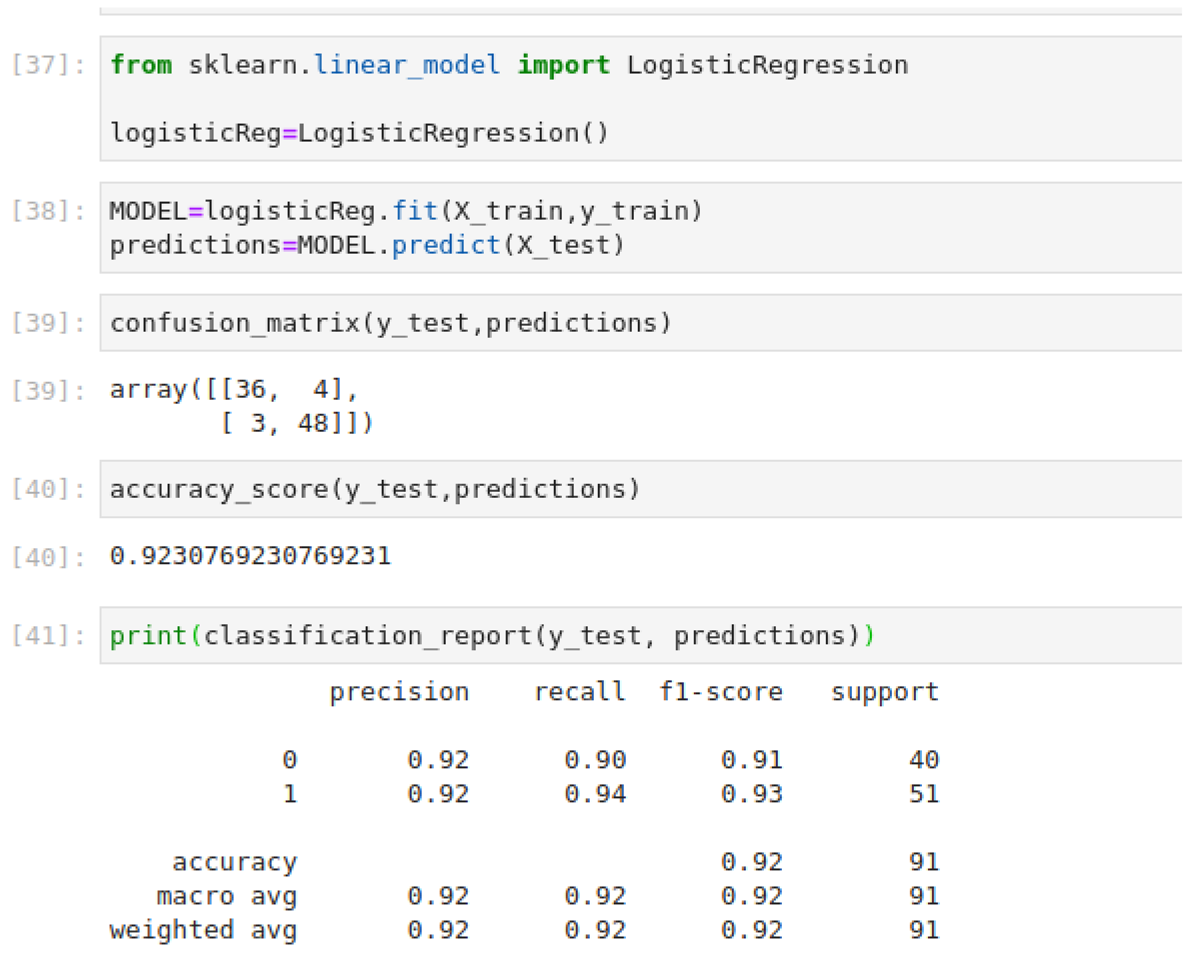
For X we have considered ‘age’, ‘oldpeak’, ‘trestbps’, ‘chol’, ‘thalach’ and y has a target, which is basically an output variable.

We have Compared the following classifiers:

1. Logistic Regression
    a. Logistic regression works well for cases where the dataset is linearly separable.
    b. Logistic regression not only gives a measure of how relevant an independent variable is, but also tells us about the direction of the relationship.
2. Random Forest
    a. This algorithm can be used for both classifications and regression tasks. It provides higher accuracy through cross validation.
    b. This classifier will handle the missing values and maintain the accuracy of a large proportion of data
3. Decision tree
    a. Decision tree implicitly perform variable screening or feature selection. Non-linear relationships between parameters do not affect the tree performance.
    b. Another main reason was that decision trees are very easy to interpret.

Implementing the algorithms:

Logistic Regression:



```
[37]: from sklearn.linear_model import LogisticRegression

      logisticReg=LogisticRegression()

[38]: MODEL=logisticReg.fit(X_train,y_train)
      predictions=MODEL.predict(X_test)

[39]: confusion_matrix(y_test,predictions)

[39]: array([[36,  4],
             [ 3, 48]])

[40]: accuracy_score(y_test,predictions)

[40]: 0.9230769230769231

[41]: print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91        40
           1       0.92      0.94      0.93        51

    accuracy                           0.92        91
   macro avg       0.92      0.92      0.92        91
weighted avg       0.92      0.92      0.92        91
```

Random Forest:

```
[48]: from sklearn.ensemble import RandomForestClassifier
```

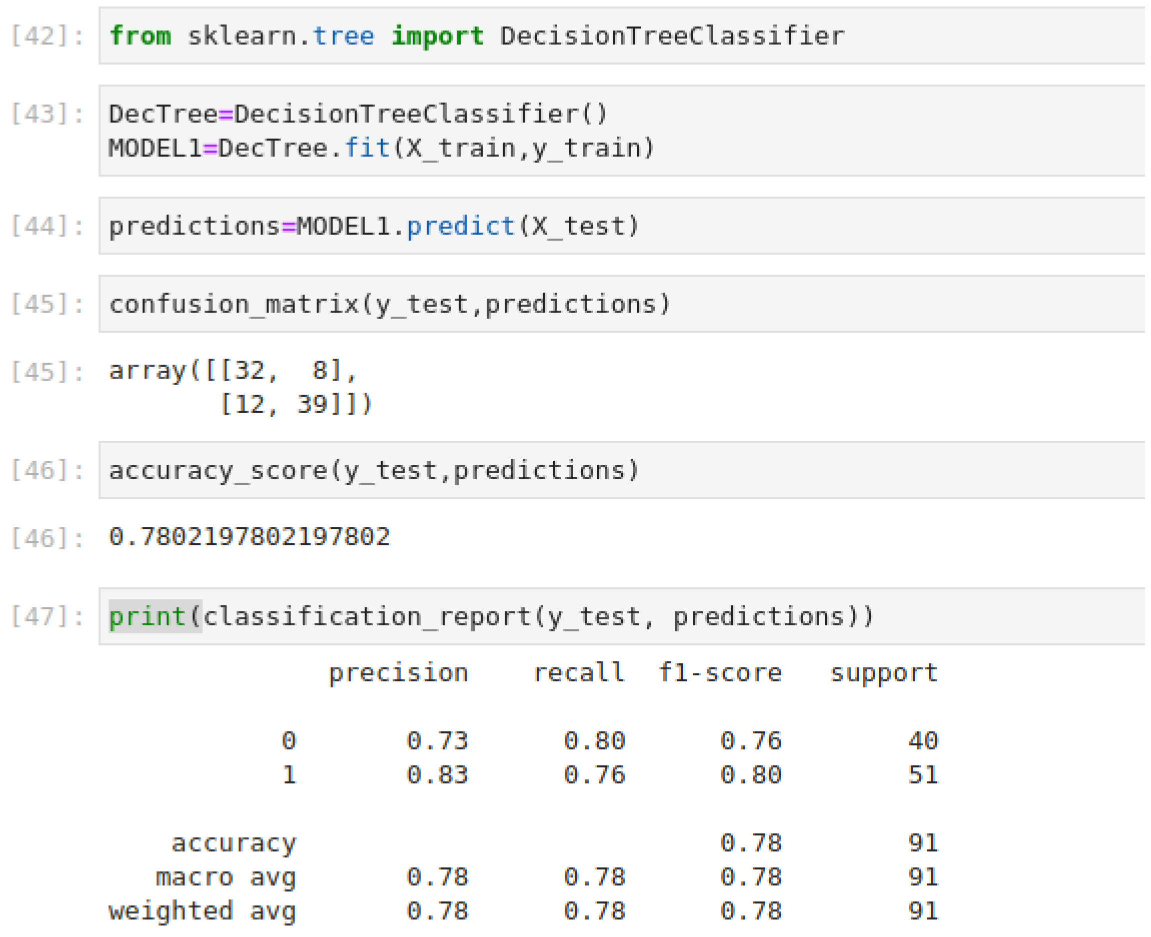
```
[49]: RandFor=RandomForestClassifier()
      MODEL2 = RandFor.fit(X_train, y_train)
      predictions = MODEL2.predict(X_test)
```

```
[50]: confusion_matrix(y_test, predictions)

[50]: array([[33,  7],
             [ 7, 44]])
```

```
[51]: accuracy_score(y_test, predictions)

[51]: 0.8461538461538461
```

```
[52]: print(classification_report(y_test, predictions))

                    precision    recall  f1-score   support

                 0       0.82      0.82      0.82        40
                 1       0.86      0.86      0.86        51

          accuracy                           0.85        91
         macro avg       0.84      0.84      0.84        91
      weighted avg       0.85      0.85      0.85        91
```

Decision Tree:



```
[42]: from sklearn.tree import DecisionTreeClassifier

[43]: DecTree=DecisionTreeClassifier()
      MODEL1=DecTree.fit(X_train,y_train)

[44]: predictions=MODEL1.predict(X_test)

[45]: confusion_matrix(y_test,predictions)

[45]: array([[32,  8],
             [12, 39]])

[46]: accuracy_score(y_test,predictions)

[46]: 0.7802197802197802

[47]: print(classification_report(y_test, predictions))
                    precision    recall  f1-score   support

                 0       0.73      0.80      0.76        40
                 1       0.83      0.76      0.80        51

          accuracy                           0.78        91
         macro avg       0.78      0.78      0.78        91
      weighted avg       0.78      0.78      0.78        91
```

Detailed Results:-

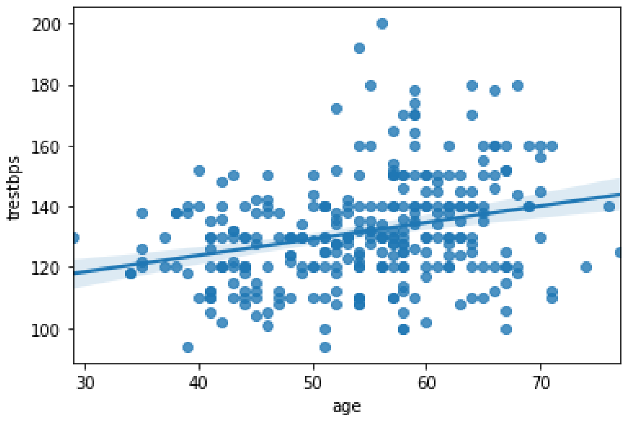
	Decision Tree	**Logistic Regression**	**Random Forest**
Accuracy	**0.78**	**0.92**	**0.85**
Precision	**0.78**	**0.92**	**0.84**
Recall	**0.78**	**0.92**	**0.84**
f1 score	**0.78**	**0.92**	**0.84**

<u>Our Choice of algorithm</u>: **Logistic Regression**:

We wanted a classification algorithm which would give us a value of 0 or 1. Logistic regression is used to predict an outcome variable that is in numerical form, predictor variables that are either continuous or in categorical form. This was exactly our case as many of our predictor variables were numerical.Logistic regression deals with the problem by using a logarithmic transformation, called logit function on the outcome variable which allows us to model a nonlinear relation in a linear way. It expresses the linear regression equation in logarithmic terms. Also, from the graphs between target(output variable) and most correlated input variables, As there we can see that it has only one major cluster, we were expecting the best accuracy to be on Logistic Regression, we tried random forest and decision trees but the best accuracy was on Logistic Regression only.

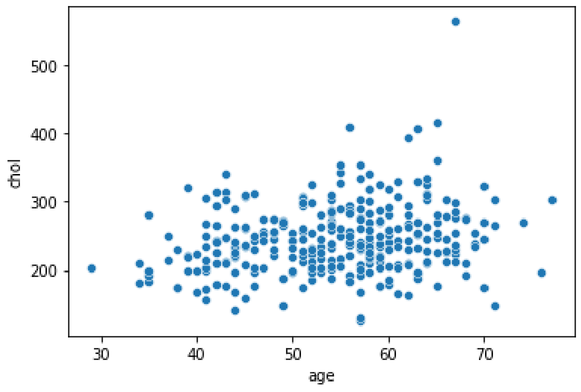
```
[25]: sns.regplot(x="age", y="trestbps", data=heart)
```

```
[25]: <AxesSubplot:xlabel='age', ylabel='trestbps'>
```



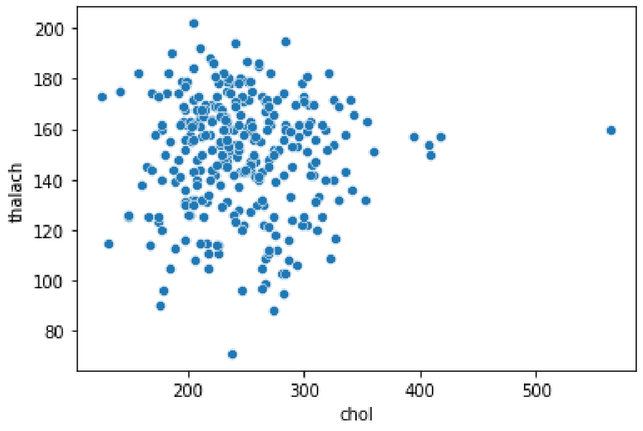
```
[26]: sns.scatterplot(x="age", y="chol", data=heart)
```

```
[26]: <AxesSubplot:xlabel='age', ylabel='chol'>
```



```
[27]: sns.scatterplot(x="chol", y = "thalach", data=heart)
```

```
[27]: <AxesSubplot:xlabel='chol', ylabel='thalach'>
```



References:-

1. Stackoverflow.com
2. Researchgate.net
3. Careerfoundry.com
4. Official documentations of Numpy, Pandas, Seaborn, Matplotlib
5. Medium.org
6. Kaggle
7. archive.ics.uci.edu/ml/datasets/heart+disease