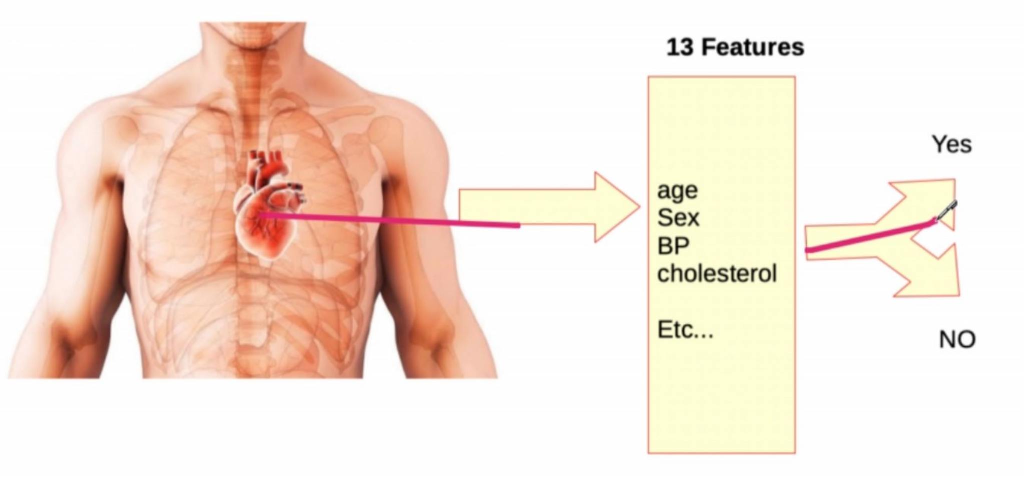
Report

Heart Disease Prediction

By: Sanjeet Pal Singh

Introduction

Heart disease is the leading cause of deaths for men and women and one person every 30 second is dead because of some heart diseases so it is one of the important problems to solve. The problem statement of the assignment is to build a machine model using the processed Cleveland data from the UCI website using a classifier around the features given in the processed cleveland file. Heart Disease Prediction using the processed-cleveland data which do have 14 different features and those features can be used as labels for the supervised learning algorithm. Also the problem is of classification type where the classification can be binary i.e. either the person can or cannot have heart disease. Whether the person can have heart disease or not , this classification can be done by building models using different classification algorithms.



Binary Classification either yes or no

Choosing the classifier

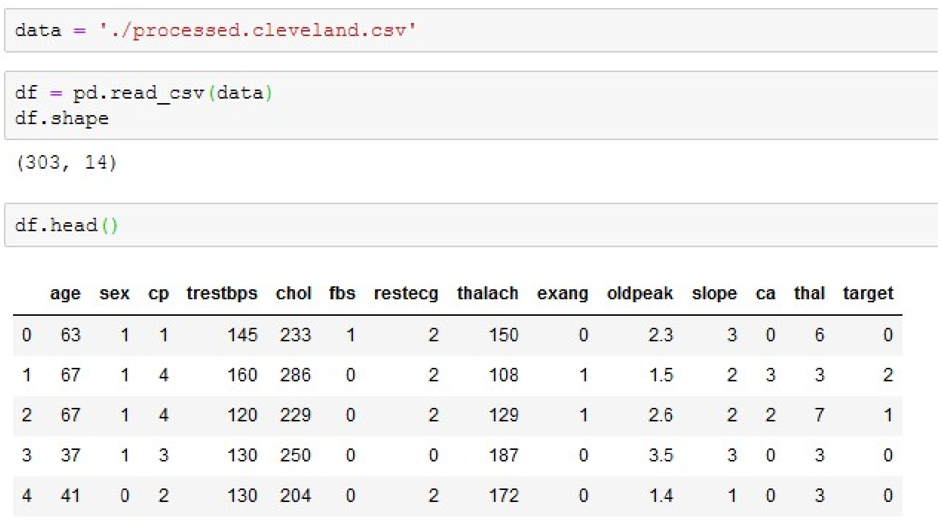
As the given problem is the classification problem so we are using four different classifier algorithms and will compare the accuracy rates of the classifiers on the basis of accuracy rate and confusion matrix results. And at the end of the report in the conclusion we will define the best classification algorithm having the accurate result. Algorithms that we are using are:

- Logistic Regression
- XGBoost
- Random Forest
- And Support Vector Machine

1. Model Building

a. Importing the data

For data preparation we are **using pandas** to import the dataset or data values to further use for visualization, plotting graphs and implementing algorithms in order to build the models. After getting the processed cleveland data which is comma separated list file we added the labels to those 14 features in order to format the data for further steps especially visualization to see any patterns and co-relation between these features.



```
data = './processed.cleveland.csv'
```

```
df = pd.read_csv(data)
df.shape
```

```
(303, 14)
```

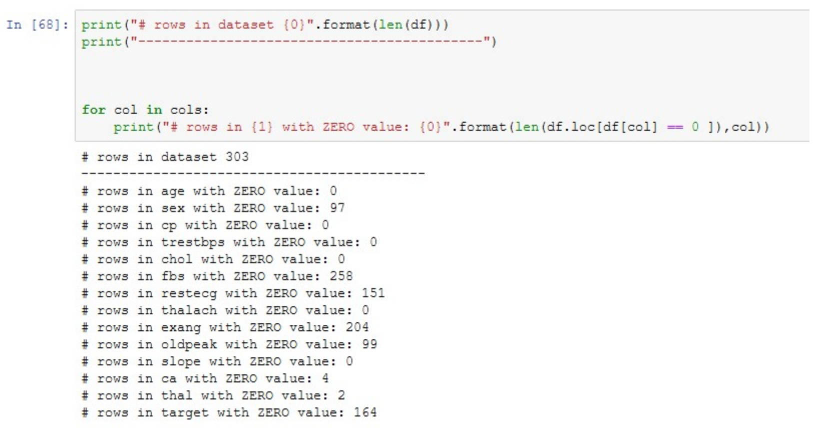
```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0

Data prepartion- Picture showing the data sets that 303 rows of data is there and 14 columns / features

Further we did checked for the null values , number of zeroes (in certain column it should not be there like age) we are having in the dataset which can cause problem while the model building and visualization. There was no null or missing value for the features to correct.

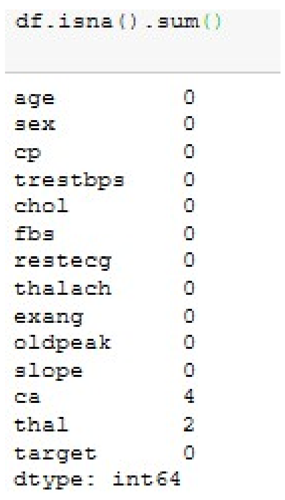
The fields like restecg and target/diagnosis should only contains the value 0 for negative or normal while 1 if the abnormality is there, so in the given datasets values for these fields were other 1 and 0, so we did converted 2's in restecg to 1 as that also means abnormality. Also for the diagnosis the values were 0,1,2,3,4 so we did replaced the values 2,3,4 using pandas library to 1 as that also means positive case of heart disease.



```
In [68]: print("# rows in dataset {0}".format(len(df)))
         print("-------------------------------------------")



         for col in cols:
             print("# rows in {1} with ZERO value: {0}".format(len(df.loc[df[col] == 0 ]),col))
```

```
# rows in dataset 303
-------------------------------------------
# rows in age with ZERO value: 0
# rows in sex with ZERO value: 97
# rows in cp with ZERO value: 0
# rows in trestbps with ZERO value: 0
# rows in chol with ZERO value: 0
# rows in fbs with ZERO value: 258
# rows in restecg with ZERO value: 151
# rows in thalach with ZERO value: 0
# rows in exang with ZERO value: 204
# rows in oldpeak with ZERO value: 99
# rows in slope with ZERO value: 0
# rows in ca with ZERO value: 4
# rows in thal with ZERO value: 2
# rows in target with ZERO value: 164
```

Only ca and thal are the fields found where the value of the zero is irrelevant has been corrected in further process by taking mean imputer using sklearn library before building models



```
df.isna().sum()
```

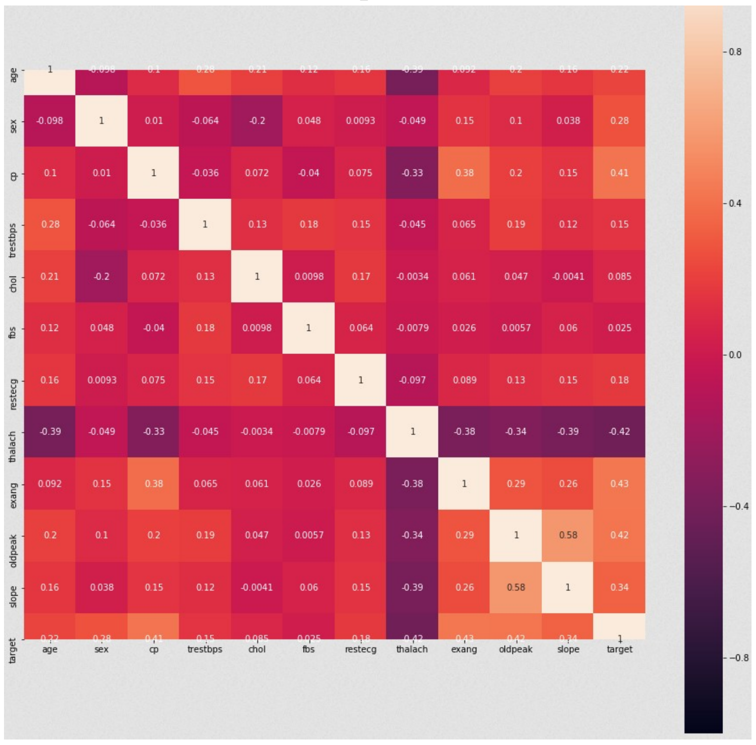
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          4
thal        2
target      0
dtype: int64
```

null value found in ca and thal column

b. Data Visualization for understanding the data

We did used the following types of graphs in order to understand and visualize the data to further find out any relation or pattern in the data. Using **matplotlib and seaborn libraries** we have plotted the following types of graphs in order do the visualization for having a looking on the data to see the relations

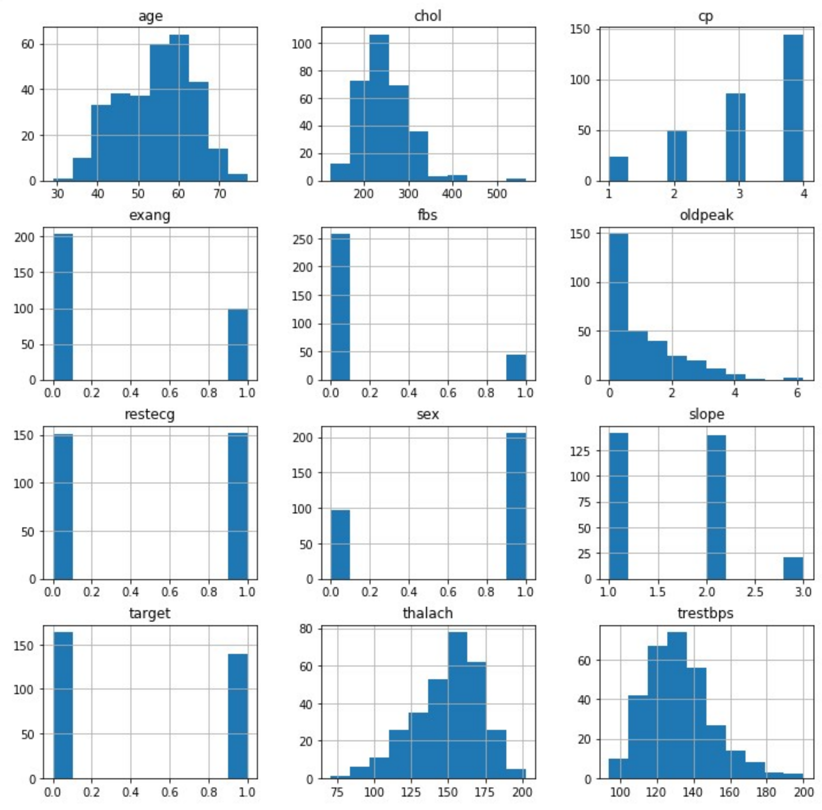
1. **Correlation Matrix Heat Map between all the features**



So from the above co-relational matrix heat plot we can see clearly that there is no high co-relation between the features due to further which we can skip the feature selection and it will also save our lot of computational power. As if the co-related features would be there we might need to put some operations to select the one feature out of the 'n' features having the co-relation, so now we can skip that part.

2. **Histograms of all the numerical data**

```
df.hist(figsize=(12,12))
plt.show()
```

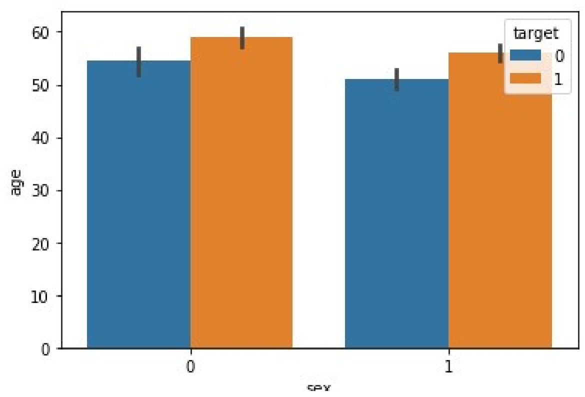


Using the histogram plot we can see that how values of different numerical values are distributed over the scale. Here cp, exlang, fbs, restecg, sex, slope, target are the categorical values but in our case we will take them as numerical values.

3. **Bar graph** (between sex vs age)

```
sns.barplot(x="sex", y="age", hue="target", data=df)
```

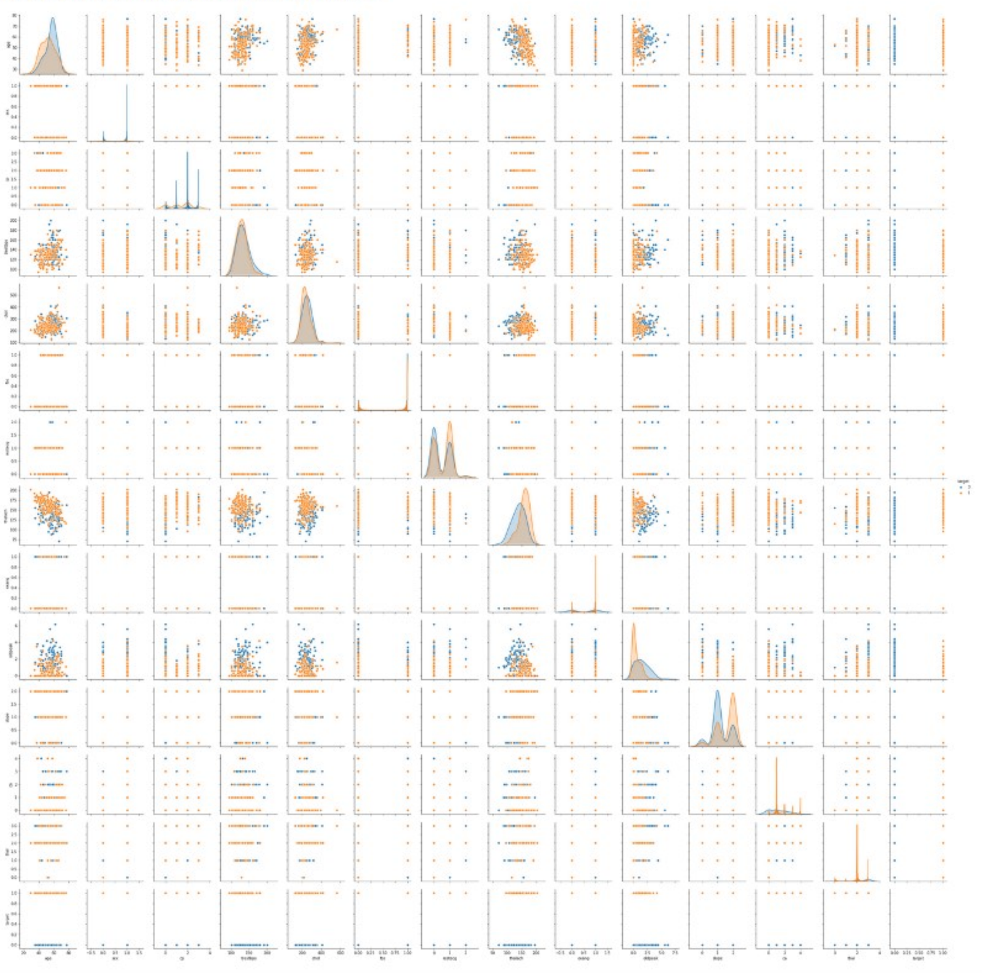
```
<matplotlib.axes._subplots.AxesSubplot at 0x25db6ceb908>
```



In the above bar graph we can see that non heart diseases is less in the cases of male(zero is male) of high ages and vice versa, means females (one is female) at a lower age catch this disease. Orange bar represent the heart diseases in male and female genders.

4. **Pair Grid Plot to see the trend or any pattern.**

```
<seaborn.axisgrid.PairGrid at 0x2cd406cdc08>
```



So pair plot is the extensive graph as it is clear from the above image as , it is the combination of various plots and we did used it in order to see the relations between the various features that we had so as we did not had any much co-relation between our features so we can see that there is no much of the trend or pattern in the plots, any upward trend or downward trend or any kind of co-relation here. Because there was very co-relation between the features that is why there is no visible trend in the scatter plot as most of the values are binary or categorical.

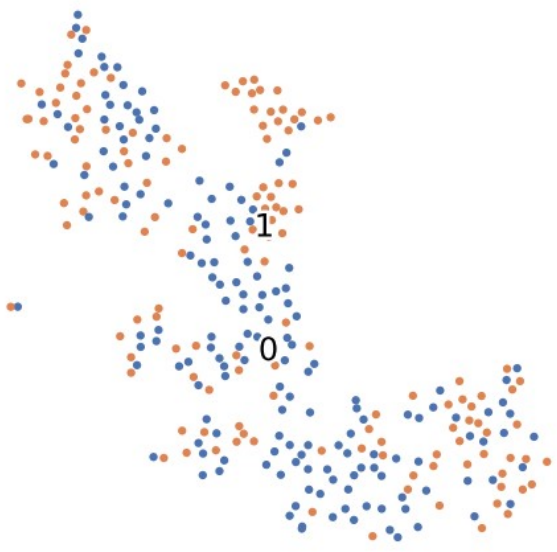
5. **Scatter Plot Graph to check the bifurcation**

In order to plot the scatter plot graph **using matplotlib** first we need to do either PCA (Principal Component Analysis) or TSNE to do the dimensional reduction, but in our case we are doing TSNE to do the dimensional reduction.

As we have more than 2 features (we have 14 in our case) which act as dimensions for the graphs so we cannot plot the 2d graph with the 14 dimensions so in order to reduce the number of dimensions to 2 in order to draw the 2d graph we are using TSNE to further draw the scatter plot.

```
fashion_scatter(df_tsne, df.target)
```

```
(<Figure size 576x576 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x25dbe0b27c8>,
 <matplotlib.collections.PathCollection at 0x25dbeb2ee48>,
 [Text(-0.33564666, -5.022967, '0'), Text(-0.6622662, 6.618767, '1')])
```



Scatter plot graph after TSNE – 0 is no heart disease (blue color) and 1 is heart disease (orange color)

The above scattered graph has been plotted is showing two different clusters one with the heart diseases and one with the non heart disease. So here we just want to see whether there is clear bifurcation or not, If there is bifurcation it means that we will achieve high accuracy for sure otherwise not. The above graph don't show clear bifurcations in the clusters so our model can have bit ambiguous behavior because it has some values which might confuse our machine learning model.

c. Feature Engineering

Before starting the feature engineering first we will start by checking whether the dataset is balanced or unbalanced.

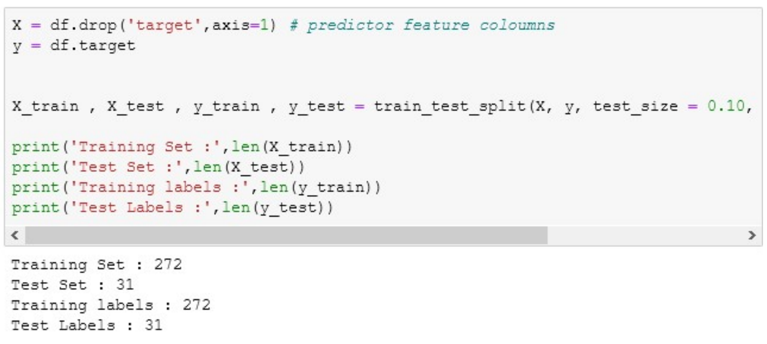
1. **Checking the result is balance or not.**

```
df.target.value_counts()
```

```
0    164
1    139
Name: target, dtype: int64
```

The number of results for the positive and negative cases should be in the same ratio so here in the above image result is slightly unbalanced.

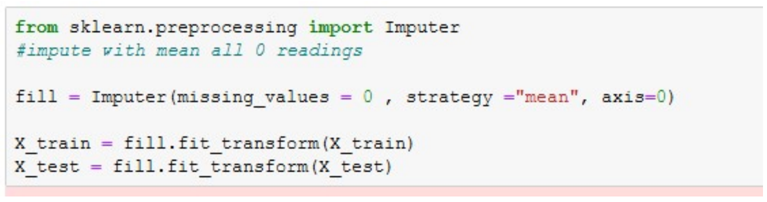
2. **Dividing data in training and test sets**



```
X = df.drop('target',axis=1) # predictor feature coloumns
y = df.target


X_train , X_test , y_train , y_test = train_test_split(X, y, test_size = 0.10,

print('Training Set :',len(X_train))
print('Test Set :',len(X_test))
print('Training labels :',len(y_train))
print('Test Labels :',len(y_test))
```

```
Training Set : 272
Test Set : 31
Training labels : 272
Test Labels : 31
```

After creating the training and test sets , we also did used the Imputer from the **sklearn preprocessing** library to impute the values by taking mean of the other values of the column. And the imputing always should be done after creating the training and test sets because the mean of the training set should always should be different then the mean of the test set (Last two lines in the below code) as compared to the mean of the complete data frame.

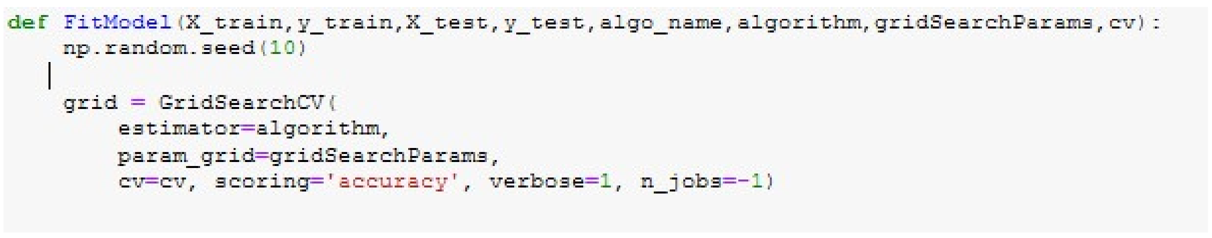


```
from sklearn.preprocessing import Imputer
#impute with mean all 0 readings

fill = Imputer(missing_values = 0 , strategy ="mean", axis=0)

X_train = fill.fit_transform(X_train)
X_test = fill.fit_transform(X_test)
```

d. Model Building and Evaluation

In feature engineering we don't have to do much operations because data is already in the numeral values. After training and test sets we can do model building and evaluate the result.

In this heart disease assignment we have to functionalize the machine learning model

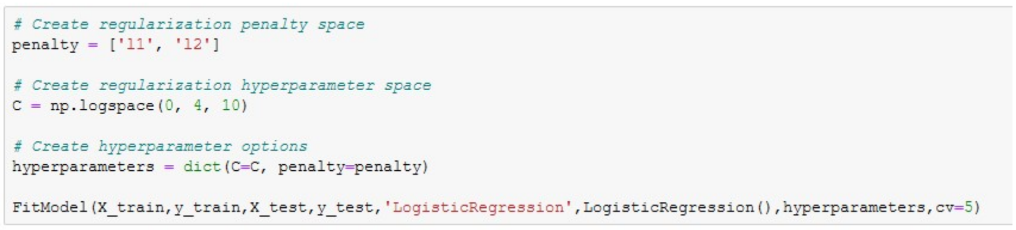


```
def FitModel(X_train,y_train,X_test,y_test,algo_name,algorithm,gridSearchParams,cv):
    np.random.seed(10)

    grid = GridSearchCV(
        estimator=algorithm,
        param_grid=gridSearchParams,
        cv=cv, scoring='accuracy', verbose=1, n_jobs=-1)
```

function defination

where function is taking the X_train, y_train,X_test, y_test, algo_name, algorithm (itself as object), gridSearchParams, cv(cross validation parameter normally value is 5 or 10). In the fitmodel function the values are getting passed to the grid search cross validation function because instead of brute forcing the values or doing hit and trial method instead we need to fine tune hyperparameters value need to be fine tuned when the accuracy score is higher, instead of taking scoring parameter as accuracy, we can also consider the other arguments such as recall, precision, f1-score etc. The hyperparameters value would be fixed when the accuracy score would be higher that's the way the gridsearch will work. It also check for cross validation while setting the hyperparameters values. Here n_jobs=-1 is basically in order to implement the grid search on all the available processors.

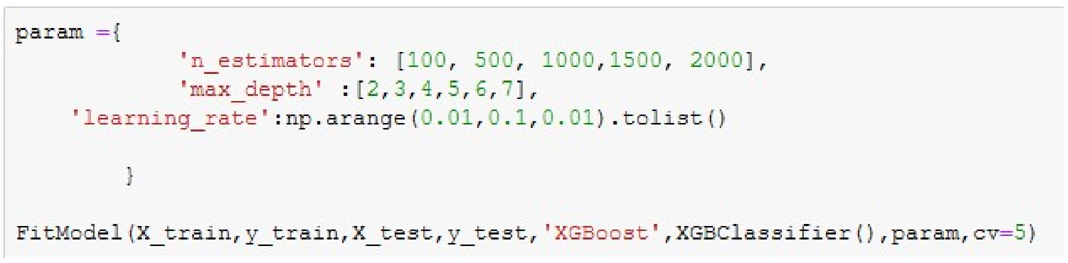
1. **Logistic Regression (General Fact: considered as the king of binary classification)**



```
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)

FitModel(X_train,y_train,X_test,y_test,'LogisticRegression',LogisticRegression(),hyperparameters,cv=5)
```

In the logistic regression we are passing two different hyper parameters one is penalty having the value L1 and L2 that is ridge and lasso regularization and c value of np.logspace which returns number distributed evenly over the logarithmic scale.

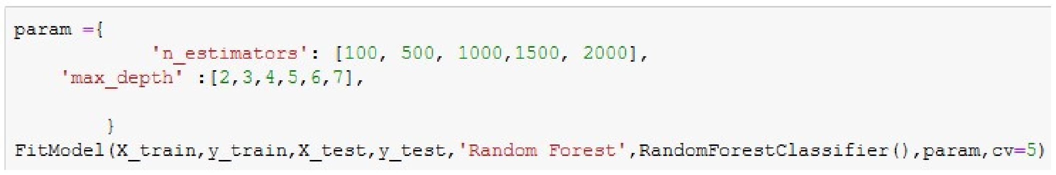
2. **XGBoost**



```
param ={
            'n_estimators': [100, 500, 1000,1500, 2000],
            'max_depth' :[2,3,4,5,6,7],
    'learning_rate':np.arange(0.01,0.1,0.01).tolist()

        }

FitModel(X_train,y_train,X_test,y_test,'XGBoost',XGBClassifier(),param,cv=5)
```

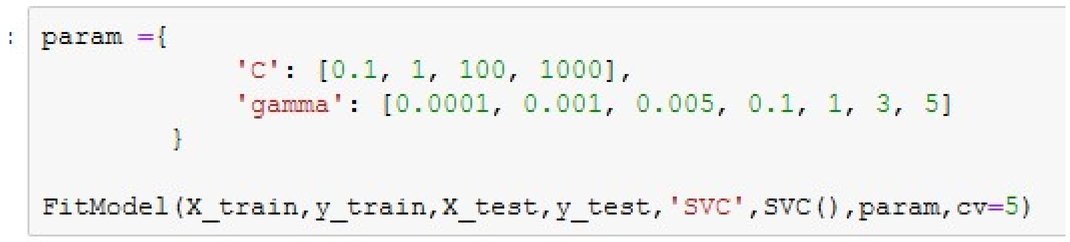
In XGBoost the hyperparameter are n_estimators, max depth and learning rate.

3. **Random Forest**



```
param ={
            'n_estimators': [100, 500, 1000,1500, 2000],
    'max_depth' :[2,3,4,5,6,7],

        }
FitModel(X_train,y_train,X_test,y_test,'Random Forest',RandomForestClassifier(),param,cv=5)
```

4. **Support Vector Component – SVC**



```
param ={
            'C': [0.1, 1, 100, 1000],
            'gamma': [0.0001, 0.001, 0.005, 0.1, 1, 3, 5]
        }

FitModel(X_train,y_train,X_test,y_test,'SVC',SVC(),param,cv=5)
```

e. Results and Conclusion

Results:

1. **Logistic Regression**

```
Accuracy Score : 0.8064516129032258
        Confusion Matrix :
             [[14  2]
             [ 4 11]]
```

2. **XGBoost**

```
Accuracy Score : 0.5806451612903226
        Confusion Matrix :
             [[ 6 10]
             [ 3 12]]
```

3. **Random Forest**

```
Accuracy Score : 0.8064516129032258
        Confusion Matrix :
             [[12  4]
             [ 2 13]]
```

4. **SVC**

```
Accuracy Score : 0.6774193548387096
        Confusion Matrix :
             [[11  5]
             [ 5 10]]
```

Conclusion:

From the above results one can see that the accuracy score of the logistic and random forest classifiers is almost same, while XGBoost and SVC did not provided the expected result performance. So we can say the one should go with the logistic or random forest classification algorithms in order to get the best results. Normally the SVC and XGBoost do require large dataset in order to do well and give the better accuracy score but the logistic and random forest classifier is good for the smaller datasets.