**Heart Disease Dataset**

**Using**

**Decision Tree, Random forest, Adaboosting**

The heart as we know is an important organ of the human body that pumps blood through the body. If circulation of blood in the body is inefficient the organs like the brain suffer and if the heart stops working altogether, death occurs within minutes. Life is completely dependent on efficient functioning of the heart. The term Heart disease refers to disease of the heart. A number of factors have been shown in the data set that increases the risk of Heart disease:

* age
* sex
* chest pain type (4 values)
* 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

**Import libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# To generate confusion matrix

from sklearn.metrics import confusion\_matrix

*#To split dataset to training and test set‘*

from sklearn.model\_selection import train\_test\_split

*#classifiers*

from tree import DecisionTreeClassifier

*#to obtain accuracy score of model*

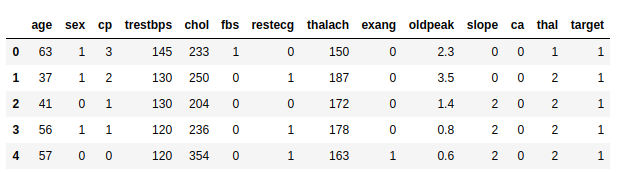
from sklearn.metrics import accuracy\_score

*#to generate classification report*

from sklearn.metrics import classification\_report

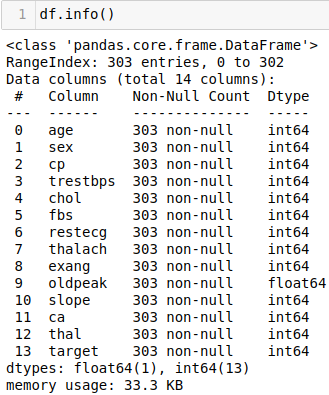
**Load Dataset**





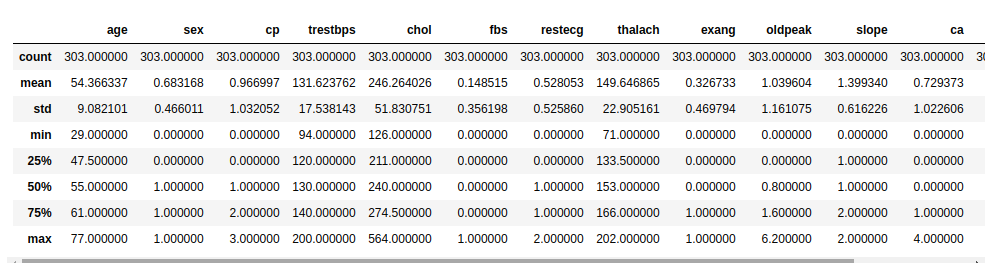
**Data exploration: Data analysis and visualization**

The data set we have consists of factors that might lead to heart disease. Information of dataset, It shows we have 13 features that will help to predict target i.e heart disease true and false

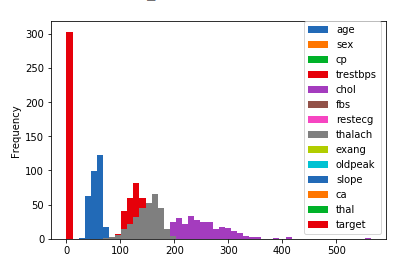


Obtaining further description on the data set

|  |
| --- |
| df.describe() |

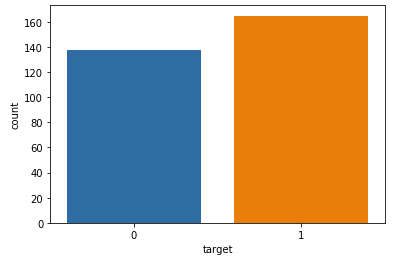


|  |
| --- |
| df.plot.hist(bins=50) |



The above histogram shows the frequency plot diagram of the features

|  |
| --- |
| sns.countplot(x="target", data=df) |



Countplot diagram shows about 160 of the record is Positive cases and around 140 cases Negetive cases

**SPlit data to train and test set**

|  |
| --- |
| X = df.drop(columns='target')  y = df['target']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1, stratify=y |

Splitting the data to train and test set considering 80% of data to train and 20% to test to predict target i.e

0: Has Disease

1: No Disease

**Features: Extraction and normalization**

### **OneHotEncoding**

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. The categorical value represents the numerical value of the entry in the dataset

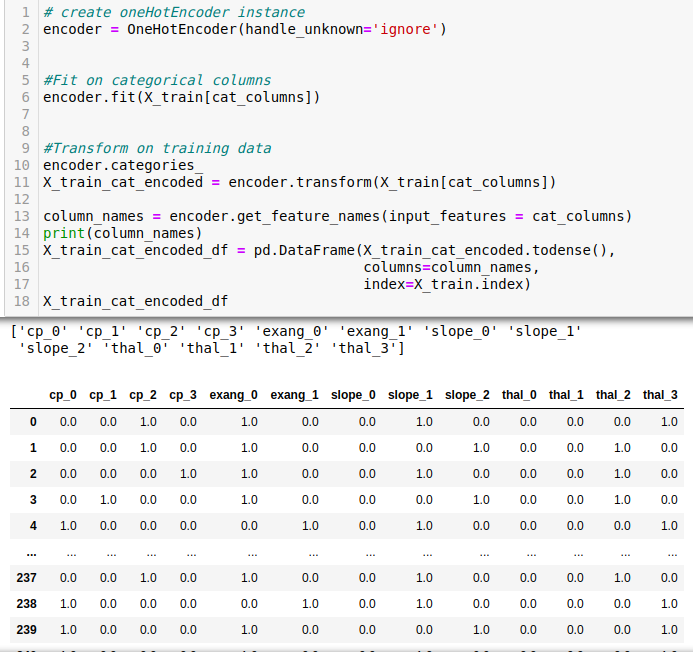


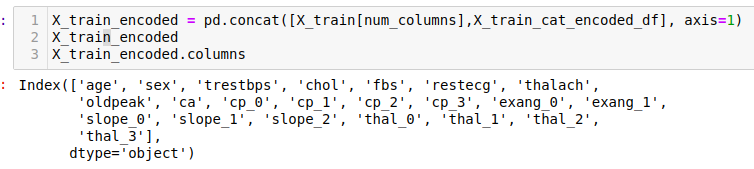
**Categorical columns** : ['cp', 'exang', 'slope', 'thal']

**Num\_columns:** ['age', 'sex', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'oldpeak', 'ca']

Importing OneHotEncoder library using:



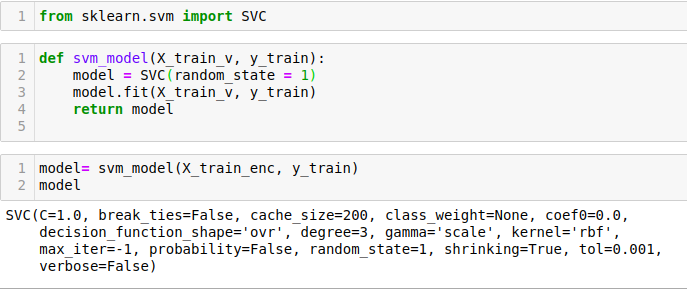




Similarly encoding the test data sets using one hotEncoder.

**Model Build**

**SVM Model**



**Decision Tree classifier**

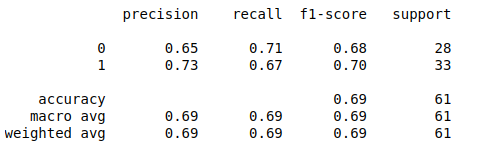
**1. Simple Decision Tree classifier**

model = DecisionTreeClassifier(random\_state = 1)

model.fit(X\_train\_encoded, y\_train)

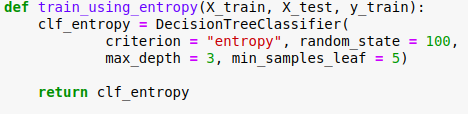
preds = model.predict(X\_test\_encoded)

print(classification\_report(y\_test, preds))



**2. Using entropy**

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of data. Entropy is the parameter used in Decision tree classifier, a function used to measure the quality of split. Higher the entropy more information content.



**Plot Decision Tree:**

from sklearn.tree import export\_graphviz

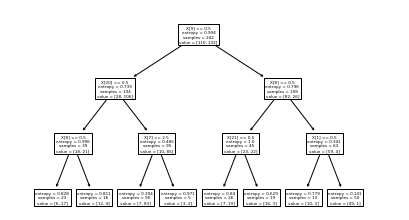
from sklearn import tree

export\_graphviz(model, out\_file='tree.dot', feature\_names = X\_train\_encoded.columns,

class\_names = ['0', '1'], rounded = True, proportion = False,

precision = 2, filled = True)

tree.plot\_tree(clf\_entropy.fit(X\_train\_encoded, y\_train))

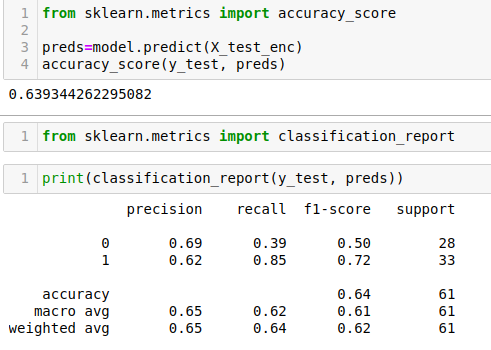


**Model evaluation using the test set**.

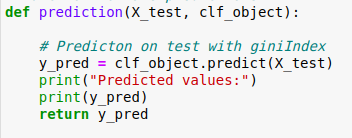
Now, will run the model on the train and test set using SVC( Support Vector Classifier) ,

and then use the test set to see what kind of prediction results we get

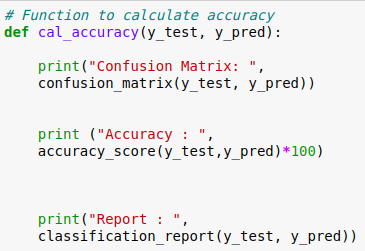
using the test data set for the Support Vector Machines as well as the Random Forest Mode ab other models

  
Above classification report is from SVM model evaluation with accuracy score of **63.93%**

Prediction Model:



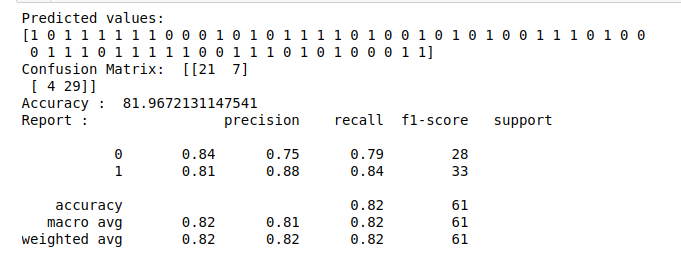
Accuracy check model:



clf\_entropy = train\_using\_entropy(X\_train\_encoded, X\_test\_encoded, y\_train)

y\_pred\_entropy = prediction(X\_test\_encoded, clf\_entropy)

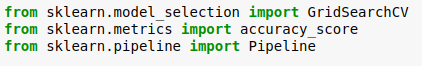
cal\_accuracy(y\_test, y\_pred\_entropy)



**Accuracy: 81.967%**

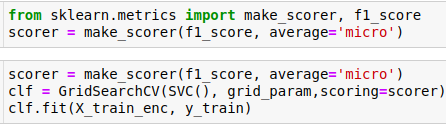
**Model Selection**

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. In this case parameters we use are max\_tree\_depth, depth, CV to fit the model and train the data and select the best model to predict the output

****

Parameter tuning for SVM model

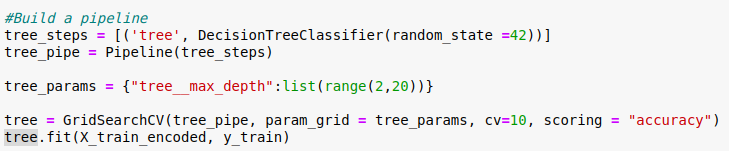




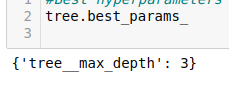


Model selection shows that it is 81.4% accurate and best parameters is C:1, kernel: linear

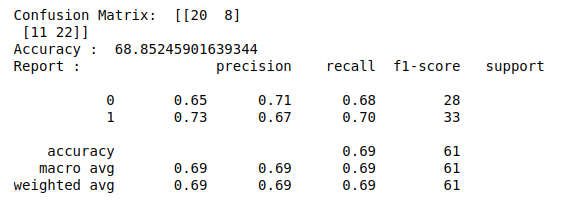
Hyper parameter tunning Decision Tree Classifier



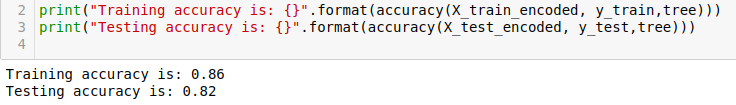
Best Parameter:



Classification report:



**Accuracy : 68.85%**

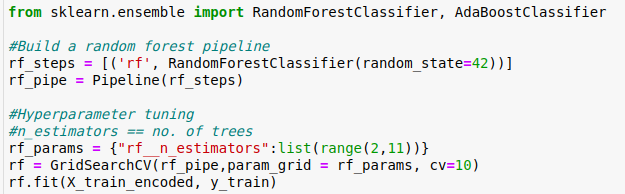
****

**Training accuracy: 86%**

**Testing accuracyL 82%**

**Random Forest**

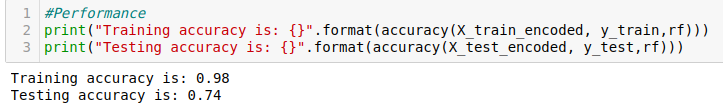
**Model1:**



Best parameter:

rf.best\_params

**{'rf\_\_n\_estimators': 9}**

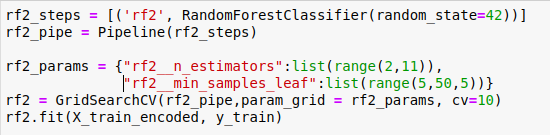


**Training accuracy: 98%**

**Testing accuracyL 74%**

Above accuracy result showed that , random forest model overfit on the training data. As we are not controlling the maximum depth of each tree in the ensemble like in simple decision tree model.

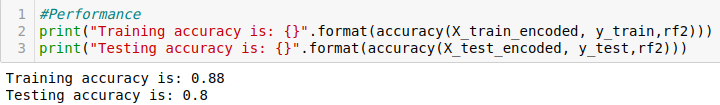
**Model 2:**

****

Best parameter:

rf2.best\_params

{'rf2\_\_min\_samples\_leaf': 5, 'rf2\_\_n\_estimators': 9}

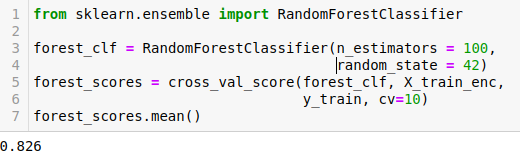


**Training accuracy: 88%**

**Testing accuracyL 80%**

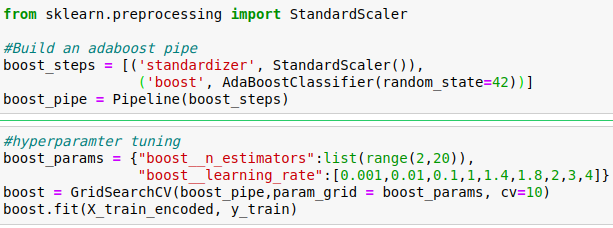
By controlling the minimum number of leaves in each tree limits the depth of the tree. With a classification accuracy of 80% on the test set,

Model3:



**AdaBoosting**

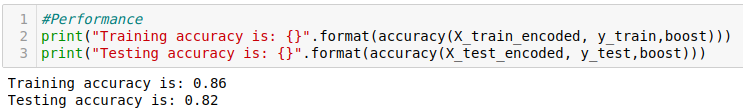
Boosting is an ensemble method that is fundamentally different from bagging and random forests. RF combines decision trees in parallel, boosting combines models additively. The ensemble model in boosting is a linear combination of simpler trees. AdaBoost, which stands for Adaptive Boosting. It helps to improve the performance of weak base learners by training new trees on points that were incorrectly classified. Thus, the complex model at the end of this iterative process delivers higher classification accuracy.



Best parameter:

boost.best\_params

{'boost\_\_learning\_rate': 1, 'boost\_\_n\_estimators': 3}

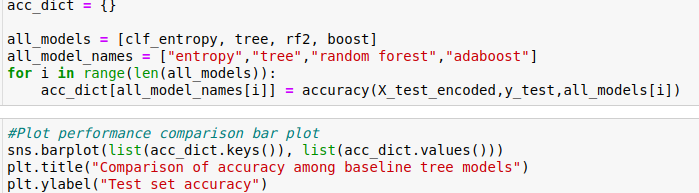


**Training accuracy: 86%**

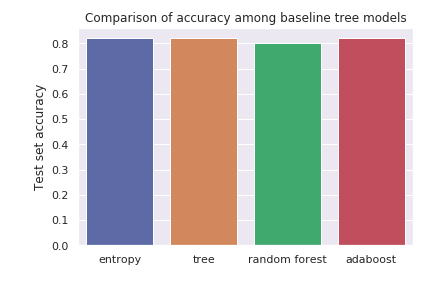
**Testing accuracyL 82%**

Accuracy obtained from AdaBoost classification is similar to simple decision tree.

**Comparing the models**

****

Below is the bar plot of model accuracy

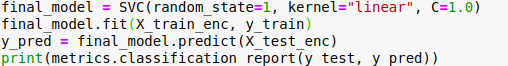


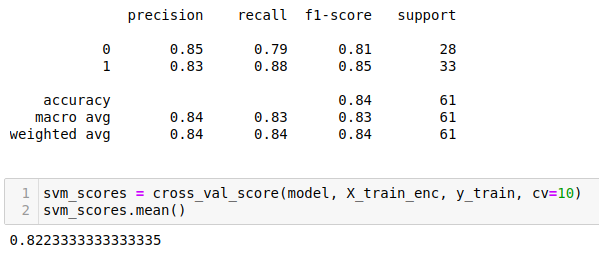
**Final Model**

Comparing the models from the above bar plot, Simple decision tree, decision tree with entropy and adaboost had almost the same accuracy.

So using adaboost for final model prediction

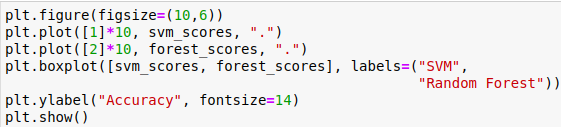
And considering SVM model using best parameters we get:

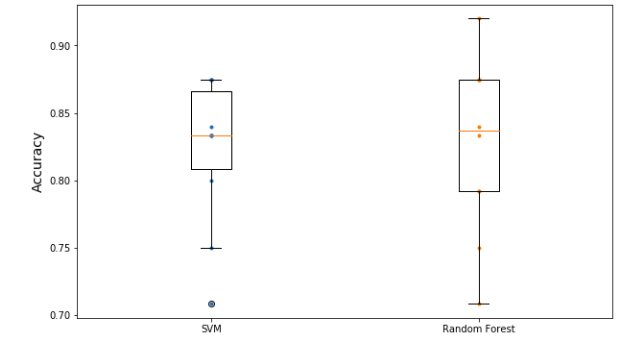




Accuracy of model : 82.23%

As we can see that the from Random Forests model 3 is slightly better than that of the SVM Model and below graphically as well.





l

**Code link**

<https://github.com/Kristiee/Hear_disease_classifier/blob/master/DT_RF_Adaboost.ipynb>

[**https://github.com/Kristiee/Hear\_disease\_classifier/blob/master/SVM\_final\_Heart\_Disease.ipynb**](https://github.com/Kristiee/Hear_disease_classifier/blob/master/SVM_final_Heart_Disease.ipynb)