**Titanic\_Dtree\_RF\_Prediction**

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn import preprocessing

**Loading Data and Data Treatment:**

titanic\_train = pd.read\_csv("train.csv")

titanic\_train.head()

Out[6]:

PassengerId Survived Pclass ... Fare Cabin Embarked

0 1 0 3 ... 7.2500 NaN S

1 2 1 1 ... 71.2833 C85 C

2 3 1 3 ... 7.9250 NaN S

3 4 1 1 ... 53.1000 C123 S

4 5 0 3 ... 8.0500 NaN S

[5 rows x 12 columns]

titanic\_train.isnull().sum()

Out[7]:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 0

dtype: int64

titanic\_train["Cabin"].mode()

Out[8]:

0 B96 B98

1 C23 C25 C27

2 G6

dtype: object

**Encoding Categorical Variables**

label\_encoder = preprocessing.LabelEncoder()

titanic\_train["Sex"] = label\_encoder.fit\_transform(titanic\_train["Sex"])

titanic\_train["Embarked"] = label\_encoder.fit\_transform(titanic\_train["Embarked"])

**Random Forest Algorithm to find imp Variables**

from sklearn.ensemble import RandomForestClassifier

features = ['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']

rf\_model = RandomForestClassifier(n\_estimators= 1000, max\_features= 2, oob\_score= True)

rf\_model.fit(X = titanic\_train[features], y = titanic\_train["Survived"])

Out[17]:

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

criterion='gini', max\_depth=None, max\_features=2,

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=1000,

n\_jobs=None, oob\_score=True, random\_state=None,

verbose=0, warm\_start=False)

print("Model Accuracy: ",rf\_model.oob\_score\_)

***Model Accuracy: 0.8087739032620922***

for feature,imp in zip(features,rf\_model.feature\_importances\_):

print(feature,imp)

Pclass 0.08674014645814597

Sex 0.26124666544869013

Age 0.25688283002534956

SibSp 0.04911199836747369

Parch 0.039625779248592244

Fare 0.2716301019408058

Embarked 0.03476247851094266

**Generating Decision Tree Model**

tree\_model = tree.DecisionTreeClassifier(max\_depth= 8)

predictors = titanic\_train[['Sex','Age','Fare']]

tree\_model.fit(X = predictors, y = titanic\_train['Survived'])

Out[22]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=8, 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')

with open("titanic\_DTree1.dot","w") as f:

f = tree.export\_graphviz(tree\_model,feature\_names=['Sex','Age','Fare'], out\_file= f)

print("DTree Model Accuracy: ", tree\_model.score(X = predictors, y = titanic\_train['Survived']))

***DTree Model Accuracy: 0.8706411698537683***

**Testing the Model**

titanic\_test = pd.read\_csv("test.csv")

titanic\_test.head()

Out[26]:

PassengerId Pclass ... Fare Embarked

0 892 3 ... 7.8292 Q

1 893 3 ... 7.0000 S

2 894 2 ... 9.6875 Q

3 895 3 ... 8.6625 S

4 896 3 ... 12.2875 S

[5 rows x 10 columns]

titanic\_test.isnull().sum()

Out[27]:

PassengerId 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

Embarked 0

dtype: int64

titanic\_test['Sex']= label\_encoder.fit\_transform(titanic\_test['Sex'])

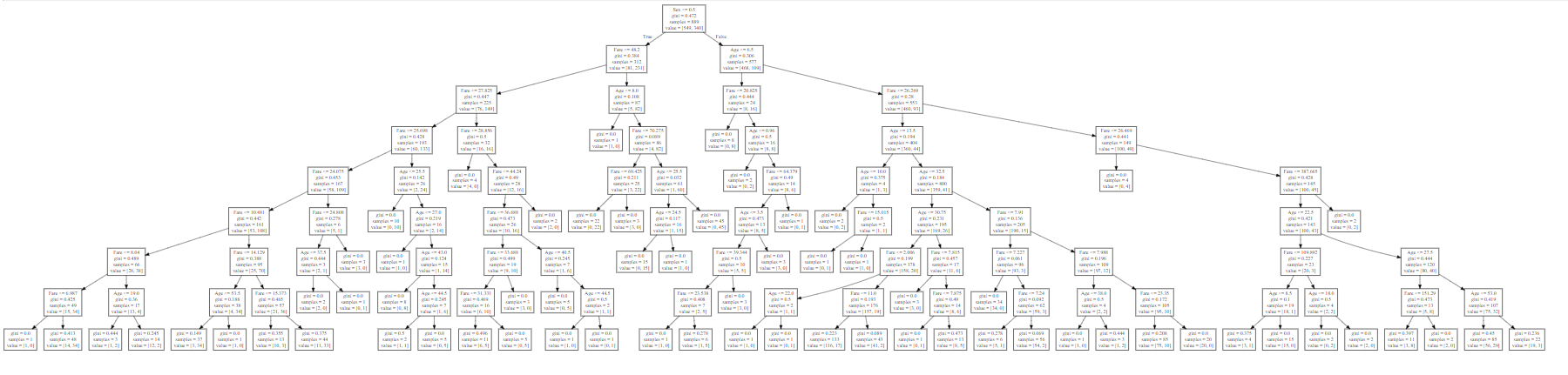
test\_features = titanic\_test[['Sex','Age','Fare']]

test\_pred = tree\_model.predict(X = test\_features)

Predicted\_output = pd.DataFrame({"PassengerId": titanic\_test["PassengerId"], "Name": titanic\_test["Name"], "Survived": test\_pred})

Predicted\_output.to\_csv("titanic\_testdata\_output1.csv", index= False)

**Decision Tree**



**Inference:**

1. Based on the importance value generated with Random forest algorithm, it is seen that the features **'Sex', 'Age' and 'Fare'** are more significant for decision tree generation.
2. Decision tree generated with these features and max-depth of 8 provides **87%** accuracy in classifying the record as Survived(Y/N) and also predicting the survival(Y/N) for any unseen record.