***Titanic Survival Model***

**Code**

**#importing required files**

In [1]: import pandas as pd

In [2]: import numpy as np

In [4]: from sklearn import tree

In [5]: from sklearn import preprocessing

In [6]: dataset=pd.read\_csv("train.csv")

**#Finding average to fill all null values**

In [7]: dataset["Age"].mean()

Out[7]: 32.69985376827896

**#Filling Null values**

In [8]: new\_age=np.where(dataset["Age"].isnull(),32,dataset["Age"])

In [9]: dataset["Age"]=new\_age

**#Changing string to numerical values**

In [11]: label\_encoder=preprocessing.LabelEncoder()

In [16]: gender = label\_encoder.fit\_transform(dataset["Sex"])

**#creating model**

In [18]: tree\_model=tree.DecisionTreeClassifier()

In [20]: tree\_model.fit(X=pd.DataFrame(gender),y=dataset["Survived"])

Out[20]:

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')

**#Exporting decision tree**

In [23]: with open("Dtree3.dot","w") as f:

...: f=tree.export\_graphviz(tree\_model,feature\_names=["Sex"],out\_file=f);

**# adding extra columns to model**

In [27]: predictor=pd.DataFrame([gender,dataset["Age"]]).T

In [28]: tree\_model.fit(X=predictor,y=dataset["Survived"])

Out[28]:

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')

**#exporting decision tree again**

In [29]: with open("Dtree4.dot","w") as f:

...: f=tree.export\_graphviz(tree\_model,feature\_names=["Sex","Age"],out\_file=f);

**# adding third required column**

In [30]: predictor=pd.DataFrame([gender,dataset["Age"],dataset["Fare"]]).T

In [31]: tree\_model.fit(X=predictor,y=dataset["Survived"])

Out[31]:

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')

**# Exporting decision tree for three columns**

In [32]: with open("Dtree5.dot","w") as f:

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

**# calculate and print score**

In [33]: tree\_model.score(X=predictor,y=dataset["Survived"])

Out[33]: 0.9763779527559056

**#testing**

In [34]: dataset2=pd.read\_csv("test.csv")

In [36]: dataset2["Age"].mean()

Out[36]: 33.23741007194245

In [37]: new\_data2=np.where(dataset2["Age"].isnull(),33,dataset2["Age"])

In [38]: dataset2["Age"]=new\_data2

In [39]: encoded\_sex\_test=label\_encoder.fit\_transform(dataset2["Sex"])

In [41]: test\_features=pd.DataFrame([encoded\_sex\_test,dataset2["Age"],dataset2["Fare"]]).T

In [43]: test\_preds=tree\_model.predict(X=test\_features)

In [44]: predicted\_output=pd.DataFrame({"PassengerId":dataset2["PassengerId"],"Survived":test\_preds})

**#exporting prediction output**

In [45]: predicted\_output.to\_csv("Output1.csv",index=False);

**# applying random forest**

In [46]: from sklearn.ensemble import RandomForestClassifier

In [47]: dataset["Sex"]=label\_encoder.fit\_transform(dataset["Sex"])

In [48]: rf\_model=RandomForestClassifier(n\_estimators=1000,max\_features=2,oob\_score=True)

In [49]: features=["Sex","Age","Fare"]

In [50]: rf\_model.fit(X=dataset[features],y=dataset["Survived"])

Out[50]:

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)

**# Accuracy Score**

In [51]: print("OOb Accuracy")

OOb Accuracy

In [52]: print("OOb Accuracy", rf\_model.oob\_score\_)

**OOb Accuracy 0.7874015748031497**

**Decision tree**

Decision tree info given in the file “Titanic\_Survival\_Dtree”