Titanic_Dtree_RF_Prediction

Data Launching and Data Treatement :

```
import pandas as pd

import numpy as np

from sklearn import tree

from sklearn import preprocessing

titanic_train = pd.read_csv("train.csv")

titanic_train.head()

Out[2]:

   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]


n [3]: titanic_train.isnull().sum()

Out[3]:

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[4]:

```
0       B96 B98
1   C23 C25 C27
2            G6
dtype: object
```

```
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[6]:

```
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:  0.8031496062992126
```

```
 for feature,imp in zip(features,rf_model.feature_importances_):
   print(feature,imp)
```

Pclass 0.0855065169311984

Sex 0.2629397870398745

Age 0.2558301125811531

SibSp 0.0512276152100811

Parch 0.03963163239755314

Fare 0.27023471611003264

Embarked 0.03462961973010705

Generating Decision Tree Model:

```
tree_model = tree.DecisionTreeClassifier(max_depth= 6, max_leaf_nodes= 10)
predictors = titanic_train[['Sex','Age','Fare']]
tree_model.fit(X = predictors, y = titanic_train['Survived'])
Out[9]:
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                max_depth=6, max_features=None, max_leaf_nodes=10,
                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:  0.8020247469066367
```

Testing the Model:

```
titanic_test = pd.read_csv("test.csv")
titanic_test.head()
Out[11]:
   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[12]:
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'])

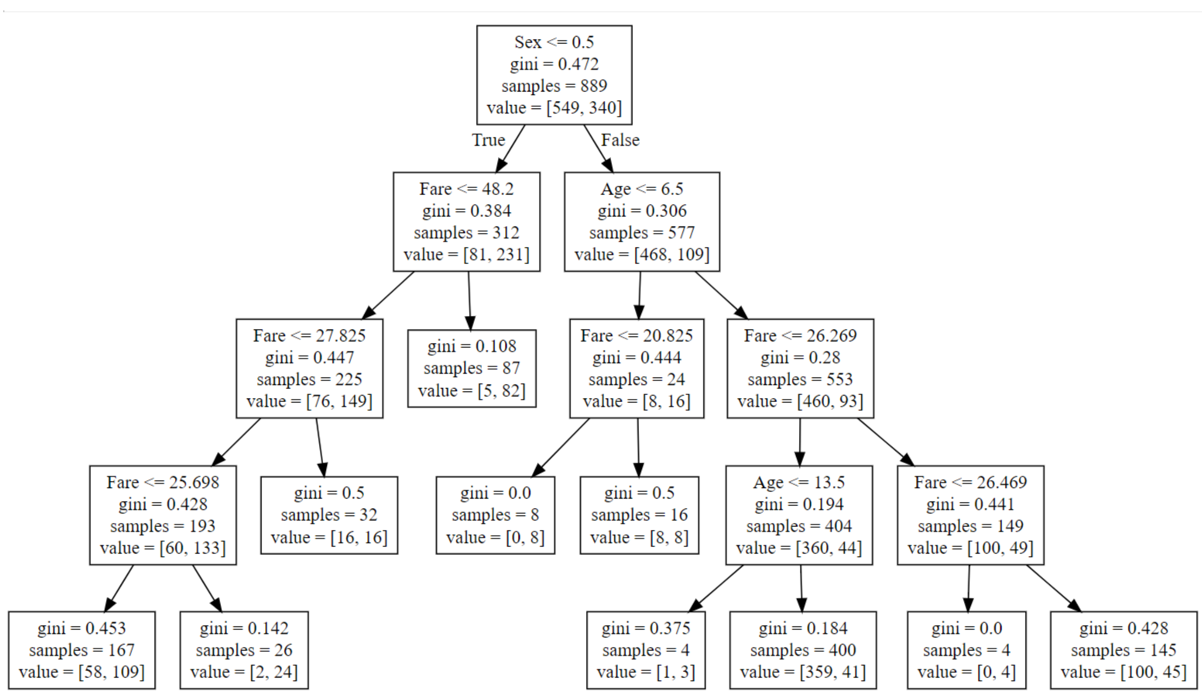
test_features = titanic_test[['Sex','Age','Fare']]

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)

<u>Decision Tree:</u>



Rules:

Survived- YES

1. If the person is a female and fare greater than 48.2 then there is a high probability that the person survived(Y)
2. If the person is a female and fare less than 25.69 then there is a high probability that the person survived(Y)
3. If the person is a female and fare ranges between 25.69.8 to 27. then there is a high probability that the person survived(Y)
4. If the person is a male with age less than 6.5 and fare less than 20.8. then there is a high probability that the person survived(Y)
5. If the person is a male with age in range of 6.5 to 13.5 and fare less than 26.2. then there is a high probability that the person survived(Y)
6. If the person is a male with age greater than 6.5 and fare in range 26.2 to 26.4. then there is a high probability that the person survived(Y)
7. If the person is a male with age less than 6.5 and fare greater than 20.82 then there is a equal probability of that person surviving and dying
8. If the person is a female and fare ranges between 27.8 to 48.2 then there is a equal probability of that person surviving and dying

Survived- NO

1. If the person is a male with age is greater than 6.5 and fare greater than 26.4. then there is a high probability that the person has not survived(N)
2. If the person is a male with age is greater than 13.5 and fare less than 26.2. then there is a high probability that the person has not survived(N)

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 6 and 10 leaf nodes provides **80.2%** accuracy in classifying the record as Survived(Y/N) and also predicting the survival(Y/N) for any unseen record.