Decision tree - Passenger Survival

## Classifying And Predicting the survivability of the passangers given in the dataset.[¶](#X673b89d998ece4704d81e39d6f37a00a52d0510)

### Classification using the decision tree[¶](#Classification-using-the-decision-tree)

In [1]:

import pandas as pd  
import numpy as np  
from sklearn import tree  
from sklearn.ensemble import RandomForestClassifier  
from sklearn import preprocessing as pp  
  
trainData = pd.read\_csv("Dataset/train.csv")  
  
trainData.isnull().sum()

Out[1]:

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

#### So no null cells in the required variable columns[¶](#Xc7d2a81e7650fd5bfc55c040881dac3e6c42814)

In [2]:

encoded\_gender = pp.LabelEncoder().fit\_transform(trainData['Sex'])

In [3]:

classify = pd.DataFrame([trainData['Pclass'],trainData['Age'],encoded\_gender,trainData['Fare']]).T  
  
tree\_model = tree.DecisionTreeClassifier(max\_depth=6)  
  
tree\_model.fit(X=classify,y=trainData['Survived'])

Out[3]:

DecisionTreeClassifier(max\_depth=6)

In [4]:

with open("Dtree.dot","w") as f:  
 f=tree.export\_graphviz(tree\_model,feature\_names=["Pclass","Age","Sex","Fare"],out\_file=f)

### Too big tree[¶](#Too-big-tree)

### We will use the Random forest to get the important variables to minimize the tree[¶](#X9907f94ba594ddb7220b59d8f5f2212d22fbb2e)

### After the Classification has done, the tree model is generated[¶](#Xac322525df2f4434b03457c294eb4ea1bdf417e)

#### Below we have the accuracy of the trained model[¶](#X489ca1366f8e4c7dfbadeebb2bd979782c5f76f)

In [5]:

tree\_model.score(X=classify,y=trainData["Survived"])

Out[5]:

0.8627671541057368

## =========================================================================[¶](#X858b65d38026defadccfbf786551abde88b717f)

## Now creating the predictive model using the test dataset[¶](#Xfac5820212a38b036e3fad08f430af585757c94)

In [6]:

testData = pd.read\_csv("Dataset/test.csv")  
  
testData['Sex'] = pp.LabelEncoder().fit\_transform(testData['Sex'])  
  
test\_features = pd.DataFrame([testData['Pclass'],testData['Age'],testData['Sex'],testData['Fare']]).T  
  
test\_predict = tree\_model.predict(X=test\_features)  
  
predicted\_data = pd.DataFrame({"PassengerId":testData['PassengerId'],"Survived":test\_predict})  
  
#Output file created  
predicted\_data.to\_csv("Predicted\_output.csv",index=False)  
  
predicted\_data.head()

Out[6]:

|  |  |  |
| --- | --- | --- |
|  | PassengerId | Survived |
| 0 | 892 | 0 |
| 1 | 893 | 1 |
| 2 | 894 | 0 |
| 3 | 895 | 0 |
| 4 | 896 | 0 |

## =========================================================================[¶](#X858b65d38026defadccfbf786551abde88b717f)

## Now applying RandomForestClassifier[¶](#Now-applying-RandomForestClassifier)

In [7]:

trainData['Sex'] = pp.LabelEncoder().fit\_transform(trainData['Sex'])  
trainData['Embarked'] = pp.LabelEncoder().fit\_transform(trainData['Embarked'])  
  
trainData['Age'] = np.round(trainData['Age'])  
trainData['Fare'] = np.round(trainData['Fare'])

In [8]:

rf\_model = RandomForestClassifier(n\_estimators=1000,max\_features=2,oob\_score=True)  
  
features=["Pclass","Sex","Age","SibSp","Fare","Embarked"]  
  
rf\_model.fit(X=trainData[features],y=trainData['Survived'])

Out[8]:

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

In [9]:

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

OOB Accuracy score: 0.8110236220472441

In [10]:

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

Pclass : 0.09664082784881883  
Sex : 0.2830930497424547  
Age : 0.3001951951608912  
SibSp : 0.05555661631118755  
Fare : 0.22640227784765757  
Embarked : 0.038112033088990055

### Only "Age", "Sex", "Fare" has the higher value compare to others so only these variables are the important ones.[¶](#X67d6b5d439db14d8d228ff7a321a15e45dfa321)

### Now only taking the important variables for Decision Tree[¶](#Xd48b19a67fb08d6d5d046dbb2c431fef3ab3a4c)

In [14]:

tree\_model = tree.DecisionTreeClassifier(max\_depth=6, max\_leaf\_nodes=12)  
  
cl\_data = pd.DataFrame([trainData["Age"],trainData["Sex"],trainData["Fare"]]).T  
  
tree\_model.fit(X=cl\_data,y=trainData["Survived"])

Out[14]:

DecisionTreeClassifier(max\_depth=6, max\_leaf\_nodes=12)

In [15]:

tree\_model.score(X=cl\_data,y=trainData["Survived"])

Out[15]:

0.8076490438695163

In [16]:

with open("Dtree\_Survived.dot","w") as f:  
 f=tree.export\_graphviz(tree\_model,feature\_names=["Age","Sex","Fare"],out\_file=f)

## Survival Conditions[¶](#Survival-Conditions)

##### 1. If the person is female and have paid fair more than 44.5, then the survivability is high.[¶](#X64a46e937aa83e732ea0f175623f186448a1f83)

##### 2. If the person is female and have paid fair more than 48.5 and age is more than 8 years, then the survivability is high.[¶](#Xa9d127e6a20b3e6248a39bb5c2f20958be4d2bc)

##### 3. If the person is male and the age is more than 6.5 years and paid fare more than 27, then the survivability is high.[¶](#X53ce9dc3f22fcbde631be3c28a24f503288381c)

##### 4. If the person is male and the age is more than 13.5 years and paid fare between 25.5 and 27 , then the survivability is high.[¶](#Xc7ab46cfe557f537ab3353c16560886820d5726)

##### 5. If the person is male and the age is more than 47.5 years and paid fare between 25.5 and 387.5, then the survivability is high.[¶](#X9b97ebd9607fb59eb56ea0993cd1b5d7be38fac)

##### 6. If the person is male and the age is more than 6.5 years and paid fare more than 387.5, then the survivability is high.[¶](#X0da7ae1719e2d48588e6fa45eca2e7921f3162f)

### ***\_\_***[¶](#Xcccc847b29090d252ba07a00934a0946c6a9d07)

In [ ]: