Predicting Survival in the Titanic Data Set

We will be using a decision tree to make predictions about the Titanic data set from Kaggle. This data set provides information on the Titanic passengers and can be used to predict whether a passenger survived or not.

Loading Data and modules

import numpy as np import pandas as pd import seaborn as sb import matplotlib.pyplot as plt import sklearn from pandas import Series, DataFrame from pylab import rcParams from sklearn import preprocessing from sklearn.linear_model import LogisticRegression from sklearn.cross_validation import train_test_split from sklearn import metrics from sklearn.metrics import classification_report

Url= [https://raw.githubusercontent.com/BigDataGal/Python-for-DataScience/master/titanic-train.csv (https://raw.githubusercontent.com/BigDataGal/Python-for-DataScience/master/titanic-train.csv)](https://raw.githubusercontent.com/BigDataGal/Python-for-DataScience/master/titanic-train.csv)

titanic = pd.read_csv(url)

titanic.columns = ['PassengerId','Survived','Pclass','Name','Sex','Age','SibSp','Parch','Ticket','Fare','Cabin','E mbarked']

You use only Pclass, Sex, Age, SibSp (Siblings aboard), Parch (Parents/children aboard), and Fare to predict whether a passenger survived.

In [1]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_
import seaborn as sns
```

In [2]:
```
Data= sns.load_dataset('titanic')
Data.head()
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Na
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Na
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Na

In [3]:
```
#You use only Pclass, Sex, Age, SibSp (Siblings aboard), Parch (Parents/children
# to predict whether a passenger survived.
```

In [4]:
```
Data.drop(axis=1,columns=["embarked","class",'who','adult_male','deck','embark_to
Data.head()
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500

In [5]:
```
Y=Data["survived"]
Y.head()
```

Out[5]:
```
0    0
1    1
2    1
3    1
4    0
Name: survived, dtype: int64
```

In [6]:
```
Data['age'].fillna(method ='ffill',inplace = True)
Data
```

Out[6]:

	survived	pclass	sex	age	sibsp	parch	fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500
...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000
887	1	1	female	19.0	0	0	30.0000
888	0	3	female	19.0	1	2	23.4500
889	1	1	male	26.0	0	0	30.0000
890	0	3	male	32.0	0	0	7.7500

891 rows × 7 columns

In [7]:
```
Sex = Data["sex"]
Dummy = pd.get_dummies(Sex)
Dummy.drop(columns=["female"],inplace=True)
Dummy.rename(columns={"male": "sex"})
```

Out[7]:

	sex
0	1
1	0
2	0
3	0
4	1
...	...
886	1
887	0
888	0
889	1
890	1

891 rows × 1 columns

In [8]:
```
Data
```

Out[8]:

	survived	pclass	sex	age	sibsp	parch	fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500
...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000
887	1	1	female	19.0	0	0	30.0000
888	0	3	female	19.0	1	2	23.4500
889	1	1	male	26.0	0	0	30.0000
890	0	3	male	32.0	0	0	7.7500

891 rows × 7 columns

In [9]:
```
Data.drop(axis=1,columns=['sex',"survived"],inplace=True)
Data.head()
```

Out[9]:

	pclass	age	sibsp	parch	fare
0	3	22.0	1	0	7.2500
1	1	38.0	1	0	71.2833
2	3	26.0	0	0	7.9250
3	1	35.0	1	0	53.1000
4	3	35.0	0	0	8.0500

In [10]:
```
df = pd.merge(Data ,Dummy ,left_index=True,right_index=True)
df.rename(columns={'male':'sex'},inplace= True)
df.head()
```

Out[10]:

	pclass	age	sibsp	parch	fare	sex
0	3	22.0	1	0	7.2500	1
1	1	38.0	1	0	71.2833	0
2	3	26.0	0	0	7.9250	0
3	1	35.0	1	0	53.1000	0
4	3	35.0	0	0	8.0500	1

In [11]:
```
#checking wehther there is any null value or not
for i in df.columns:
    print(i,df[i].isnull().sum())
```

```
pclass 0
age 0
sibsp 0
parch 0
fare 0
sex 0
```

In [12]:
```
x_train,x_test,y_train,y_test = train_test_split(df,Y,test_size = 0.25, random_st
```

In [13]:
```
clf = DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

Out[13]:
```
DecisionTreeClassifier(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=False,
                       random_state=None, splitter='best')
```

In [14]:
```
feature_name=list(df.columns)
class_name = list(y_train.unique())
feature_name
```

Out[14]:
```
['pclass', 'age', 'sibsp', 'parch', 'fare', 'sex']
```

In [15]:
```
clf.score(x_train,y_train)
```

Out[15]:
```
0.9925149700598802
```

In [16]:
```
y_pred = clf.predict(x_test)
```

In [17]:
```
# accuracy of our classification tree
print("Accuracy of this model is {}%".format(round((100*clf.score(x_test,y_test)
```

```
Accuracy of this model is 74.888%
```

In [18]:
```
from sklearn import metrics
print("Confusion matrix:-\n",metrics.confusion_matrix(y_test, y_pred))
```

```
Confusion matrix:-
 [[110  23]
 [ 33  57]]
```

In [ ]: