In [1]:
```
# Author: Chidura Santosh
# Importing required Libraries

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.model_selection import train_test_split
from sklearn import metrics
```

In [2]:
```
# Reading the data into data frame df
df=pd.read_csv('https://raw.githubusercontent.com/BigDataGal/Python-for-Data-Science/master/titanic-train.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```
df.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:
```
# as asked to use only few classes
df = df.drop(columns = ['PassengerId','Ticket','Name','Cabin','Embarked'])
```

In [6]:
```
df.head()
```

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

In [7]:
```
df.isnull().values.sum()
```

Out[7]: 177

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Sex         891 non-null object
Age         714 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
dtypes: float64(2), int64(4), object(1)
memory usage: 48.8+ KB
```

In [9]:
```
#age has many null

def removenullfromAge(df):
    df.Age = df.Age.fillna(-0.5)
    bins = (-1, 0, 5, 12, 18, 25, 35, 60, 120)
    group_names = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Senior']
    categories = pd.cut(df.Age, bins, labels=group_names)
    df.Age = categories
    return df
df=removenullfromAge(df)
```

In [10]:
```
df.isnull().values.sum()
```

Out[10]:
```
0
```

In [11]:
```
bins = (-1, 0, 8, 15, 31, 1000)
group_names = ['Unknown', 'Low_Fare', 'Med_fare', 'High_Fare', 'Very_High_Fare']
categories = pd.cut(df.Fare, bins, labels=group_names)
df.Fare = categories
```

In [12]:
```
df.head()
```

Out[12]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	Student	1	0	Low_Fare
1	1	1	female	Adult	1	0	Very_High_Fare
2	1	3	female	Young Adult	0	0	Low_Fare
3	1	1	female	Young Adult	1	0	Very_High_Fare
4	0	3	male	Young Adult	0	0	Med_fare

In [13]:
```
df.Pclass.unique()
```

Out[13]:
```
array([3, 1, 2], dtype=int64)
```

In [14]:
```
df['Sex'],uniq = pd.factorize(df['Sex'])
df['Fare'],uniq = pd.factorize(df['Fare'])
df['Age'],uniq = pd.factorize(df['Age'])
```

In [15]:
```
df.head()
```

Out[15]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	0	0	1	0	0
1	1	1	1	1	1	0	1
2	1	3	1	2	0	0	0
3	1	1	1	2	1	0	1
4	0	3	0	2	0	0	2

Data Split

In [16]:
```
y=df['Survived']
X=df.drop('Survived',axis=1)
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.30, random_state=1234)
```

In [18]:
```
y_train.shape
```

Out[18]:
```
(623,)
```

Using Random Forest for predictions

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umat
h_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [20]:
```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
```

Out[20]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', 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, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [21]:
```
prediction=random_forest.predict(X_test)
```

In [30]:
```
from sklearn import metrics
print("RandomForest regressor accuracy is",metrics.accuracy_score(y_test,prediction))
```

```
RandomForest regressor accuracy is 0.8208955223880597
```

In [23]:
```
(y_test != prediction).sum()
```

Out[23]:
```
48
```

In [24]:
```
from sklearn.metrics import classification_report
target_names = ['0','1']
print(classification_report(y_test, prediction, target_names=target_names))
```

```
             precision    recall  f1-score   support

          0       0.84      0.89      0.86       166
          1       0.79      0.72      0.75       102

avg / total       0.82      0.82      0.82       268
```