In [158]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import re

%matplotlib inline
```

In [159]:
```
training_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')
datasets=[training_data,test_data]
```

In [160]:
```
training_data.head()
```

Out[160]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nal
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Nal
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Nal

In [161]:
```
training_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

In [162]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

In [163]:
```
training_data.describe()
```

Out[163]:

	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 [164]:
```
training_data[['Pclass','Survived']].groupby(['Pclass']).mean()
```

Out[164]:

Pclass	Survived
1	0.629630
2	0.472826
3	0.242363

In [165]:
```
training_data[['Sex','Survived']].groupby(['Sex']).mean()
```

Out[165]:

Sex	Survived
female	0.742038
male	0.188908

In [166]:
```
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    # If the title exists, extract and return it.
    if title_search:
        return title_search.group(1)
    return ""
```

In [167]:
```
for dataset in datasets:
    # Mapping Sex to binary
    dataset['Sex'] = dataset['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

    # Fill Embarked missing data to Mode
    dataset['Embarked'] = dataset['Embarked'].fillna(dataset['Embarked'].mode().i

    # Mapping Embarked
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).ast


    # Fill Fare missing data to Mean
    dataset['Fare'] = dataset['Fare'].fillna(dataset['Fare'].mean())

    # Get avg std and null count of age
    age_avg = dataset['Age'].mean()
    age_std = dataset['Age'].std()
    age_null_count = dataset['Age'].isnull().sum()

    # Generate random age within 95% confidence interval
    age_null_random_list = np.random.randint(age_avg - 1.96*age_std/(age_null_cou
    dataset['Age'][np.isnan(dataset['Age'])] = age_null_random_list
    dataset['Age'] = dataset['Age'].astype(int)

    # Extract Title from Name, replace french title to uniform title, replace rar
    dataset['Title'] = dataset['Name'].apply(get_title)
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col'
                                                 'Don', 'Dr', 'Major', 'Rev', 'Si
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

    # Mapping titles
    title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:22: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [168]:
```
drop_elements = ['Cabin','Ticket','PassengerId','Name']
training_data = training_data.drop(drop_elements, axis = 1)
test_data = test_data.drop(drop_elements, axis=1)
```

In [173]:
```
test_data.head()
```

Out[173]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	3	1	34	0	0	7.8292	2	1
1	3	0	47	1	0	7.0000	0	3
2	2	1	62	0	0	9.6875	2	1
3	3	1	27	0	0	8.6625	0	1
4	3	0	22	1	1	12.2875	0	3

In [170]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
Pclass      418 non-null int64
Sex         418 non-null int32
Age         418 non-null int32
SibSp       418 non-null int64
Parch       418 non-null int64
Fare        418 non-null float64
Embarked    418 non-null int32
Title       418 non-null int64
dtypes: float64(1), int32(3), int64(4)
memory usage: 21.3 KB
```

In [171]:
```
train=training_data.values
X = train[0::, 1::]
Y = train[0::, 0]
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,random_state=1)
```

In [189]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(3)
fitted_knn=knn.fit(X_train,Y_train)
fitted_knn.score(X_test,Y_test)
```

Out[189]: 0.73991031390134532

In [190]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
fitted_dtc=dtc.fit(X_train,Y_train)
fitted_dtc.score(X_test,Y_test)
```

Out[190]: 0.7488789237668162

In [191]:
```
from sklearn.svm import SVC
svc=SVC(probability=True)
fitted_svc=svc.fit(X_train,Y_train)
fitted_svc.score(X_test,Y_test)
```

Out[191]: 0.69506726457399104

In [ ]:
```

```