In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
import numpy
import seaborn as sns
```

In [2]:
```
df=pd.read_csv('C:/Users/user/Downloads/Train.csv')
```

In [3]:
```
df
```

Out[3]:

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

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

891 rows × 12 columns

In [4]:
```
g=df.Age.groupby(df['Pclass'])
```

In [5]:
```
g.get_group(3).mean()
```

Out[5]: 25.14061971830986

In [6]:
```
g.get_group(1).mean()
```

Out[6]: 38.233440860215055

In [7]:
```
g.get_group(2).mean()
```

Out[7]: 29.87763005780347

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [9]:
```
df=df.drop(columns="Cabin")
```

In [10]:
```
def age_approx(x):
    Age=x[0]
    Pclass=x[1]
    if pd.isnull(Age):
        if Pclass==1:
            return 38
        elif Pclass==2:
            return 29
        elif Pclass==3:
            return 25
    else:
        return Age
```

In [11]:
```
df.Age=df[['Age','Pclass']].apply(age_approx, axis=1)
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
PassengerId    0
Survived       0
```

```
Pclass          0
Name            0
Sex             0
Age             0
SibSp           0
Parch           0
Ticket          0
Fare            0
Embarked        2
dtype: int64
```

In [13]:
```
df.groupby(df['Embarked']).mean()
```

Out[13]:

Embarked	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
C	445.357143	0.553571	1.886905	30.398333	0.386905	0.363095	59.954144
Q	417.896104	0.389610	2.909091	26.175325	0.428571	0.168831	13.276030
S	449.527950	0.336957	2.350932	29.257376	0.571429	0.413043	27.079812

In [14]:
```
def em_approx(x):
    Embarked=x[0]
    Fare=x[1]
    if pd.isnull(Embarked):
        if Fare<=14:
            return 'Q'
        elif Fare<=28:
            return 'S'
        else:
            return 'C'
    else:
        return Embarked
```

In [15]:
```
df.Embarked=df[['Embarked','Fare']].apply(em_approx, axis=1)
```

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [17]:
```
df=df.drop(["PassengerId","Name","Ticket"],axis=1)
```

In [18]:
```
df
```

Out[18]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S
887	1	1	female	19.0	0	0	30.0000	S
888	0	3	female	25.0	1	2	23.4500	S
889	1	1	male	26.0	0	0	30.0000	C
890	0	3	male	32.0	0	0	7.7500	Q

891 rows × 8 columns

In [19]:
```
df
```

Out[19]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S
887	1	1	female	19.0	0	0	30.0000	S
888	0	3	female	25.0	1	2	23.4500	S
889	1	1	male	26.0	0	0	30.0000	C
890	0	3	male	32.0	0	0	7.7500	Q

891 rows × 8 columns

In [20]:
```
df.dtypes
```

Out[20]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
dtype: object
```

In [21]:
```
df_dummy=pd.get_dummies(df,columns=['Sex'])
```

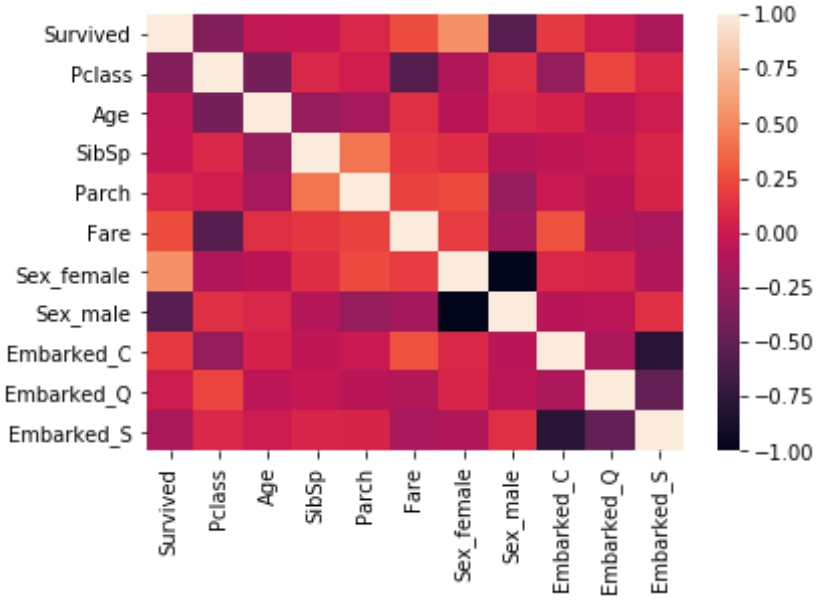
In [22]:
```
df_dummy=pd.get_dummies(df_dummy,columns=["Embarked"])
```

In [23]:
```
df_dummy.dtypes
```

Out[23]:
```
Survived        int64
Pclass          int64
Age           float64
SibSp           int64
Parch           int64
Fare          float64
Sex_female      uint8
Sex_male        uint8
Embarked_C      uint8
Embarked_Q      uint8
Embarked_S      uint8
dtype: object
```

In [24]:
```
sns.heatmap(df_dummy.corr())
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26fdd0ea0c8>
```



In [25]:
```
df_dummy.corr()
```

Out[25]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_m
Survived	1.000000	-0.338481	-0.050118	-0.035322	0.081629	0.257307	0.543351	-0.543
Pclass	-0.338481	1.000000	-0.403923	0.083081	0.018443	-0.549500	-0.131900	0.131
Age	-0.050118	-0.403923	1.000000	-0.243110	-0.174824	0.121503	-0.079949	0.079
SibSp	-0.035322	0.083081	-0.243110	1.000000	0.414838	0.159651	0.114631	-0.114
Parch	0.081629	0.018443	-0.174824	0.414838	1.000000	0.216225	0.245489	-0.245
Fare	0.257307	-0.549500	0.121503	0.159651	0.216225	1.000000	0.182333	-0.182
Sex_female	0.543351	-0.131900	-0.079949	0.114631	0.245489	0.182333	1.000000	-1.000
Sex_male	-0.543351	0.131900	0.079949	-0.114631	-0.245489	-0.182333	-1.000000	1.000
Embarked_C	0.174718	-0.251139	0.050608	-0.061970	-0.013725	0.273614	0.090223	-0.090
Embarked_Q	0.003650	0.221009	-0.071679	-0.026354	-0.081228	-0.117216	0.074115	-0.074
Embarked_S	-0.155660	0.081720	0.000570	0.070941	0.063036	-0.166603	-0.125722	0.125

In [26]:
```
X=df_dummy.drop(columns='Survived').values
```

In [27]:
```
Y=df_dummy.Survived
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [30]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.33,rando
m_state=1)
```

In [31]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(596, 10)
(295, 10)
(596,)
(295,)
```

In [32]:
```
X.shape
```

Out[32]:
```
(891, 10)
```

In [35]:
```
from sklearn.linear_model import LogisticRegression
LGR=LogisticRegression()
```

In [36]:
```
LGR.fit(x_train,y_train)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logisti
c.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[36]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbo
se=0,
                   warm_start=False)
```

In [37]:
```
y_pred=LGR.predict(x_test)
```

In [51]:
```
import sklearn.metrics
sklearn.metrics.confusion_matrix(y_test,y_pred)
```

Out[51]:
```
array([[146,  28],
       [ 39,  82]], dtype=int64)
```

In [53]:
```
sklearn.metrics.accuracy_score(y_test,y_pred)
```

Out[53]:
```
0.7728813559322034
```

In [57]:
```
print(sklearn.metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.84      0.81       174
           1       0.75      0.68      0.71       121

    accuracy                           0.77       295
   macro avg       0.77      0.76      0.76       295
weighted avg       0.77      0.77      0.77       295
```

In [ ]:
```

```