Titanic Dataset Analysis

November 13, 2017

1 Predict survial of the Titanic

1.0.1 Fundamental analysis on the famous Titanic Dataset using the logistic Regression and Visualization with seaborn.

Importing the train and test dataset

```
In [1]: import numpy as np
        import pandas as pd
        import csv

        train = pd.read_csv('G:/My Data Science files/Logistic Regression/train.csv')
        test = pd.read_csv('G:/My Data Science files/Logistic Regression/test.csv')

In [2]: train.head()

Out[2]:    PassengerId  Survived  Pclass  \
        0            1         0       3
        1            2         1       1
        2            3         1       3
        3            4         1       1
        4            5         0       3

                                                        Name     Sex   Age  SibSp  \
        0                            Braund, Mr. Owen Harris    male  22.0      1
        1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
        2                             Heikkinen, Miss. Laina  female  26.0      0
        3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
        4                           Allen, Mr. William Henry    male  35.0      0

           Parch            Ticket     Fare Cabin Embarked
        0      0         A/5 21171   7.2500   NaN        S
        1      0          PC 17599  71.2833   C85        C
        2      0  STON/O2. 3101282   7.9250   NaN        S
        3      0            113803  53.1000  C123        S
        4      0            373450   8.0500   NaN        S

In [3]: train.describe()
```

Out[3]:

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

In [4]:

```python
# To find the number of missing values in a dataframe

train.isnull().sum()
```

Out[4]:

```
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 [5]:

```python
# dropping the irrelevant columns
train1 = train.drop(['PassengerId','Name','Ticket','Cabin'], axis=1)

# nice way to encode the categorical values
cleanup_nums = {"Pclass":     {1: "High Class", 2:"Medium Class" , 3:"Lower Class"}}
train1.replace(cleanup_nums, inplace=True)
train1.head()
```

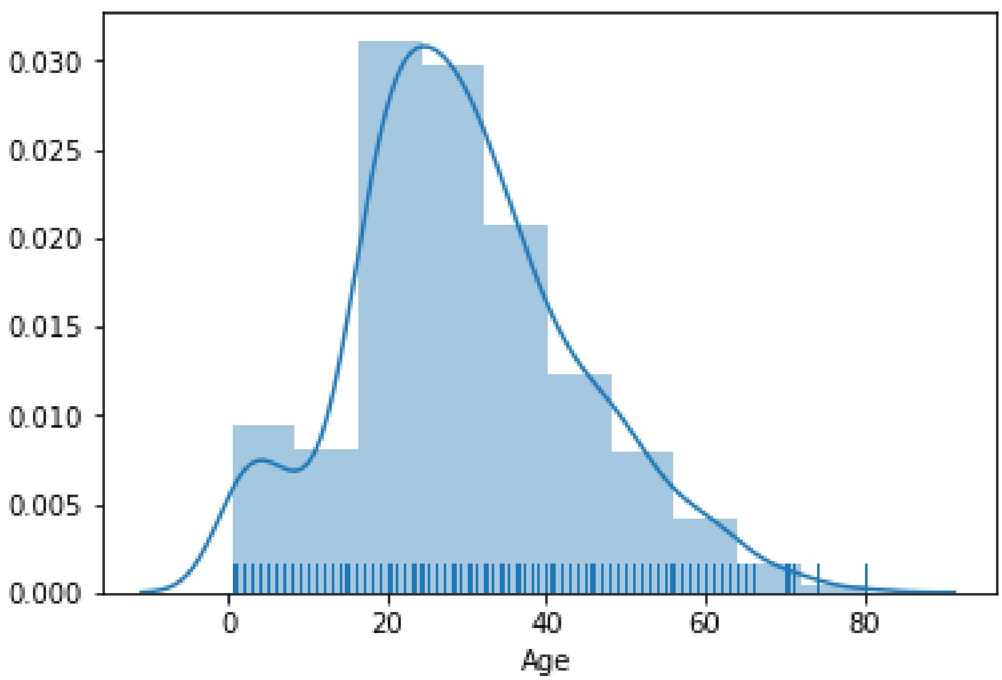
Out[5]:

```
   Survived       Pclass     Sex   Age  SibSp  Parch     Fare Embarked
0         0  Lower Class    male  22.0      1      0   7.2500        S
```

```
1          1   High Class  female  38.0      1      0  71.2833        C
2          1  Lower Class  female  26.0      0      0   7.9250        S
3          1   High Class  female  35.0      1      0  53.1000        S
4          0  Lower Class    male  35.0      0      0   8.0500        S
```

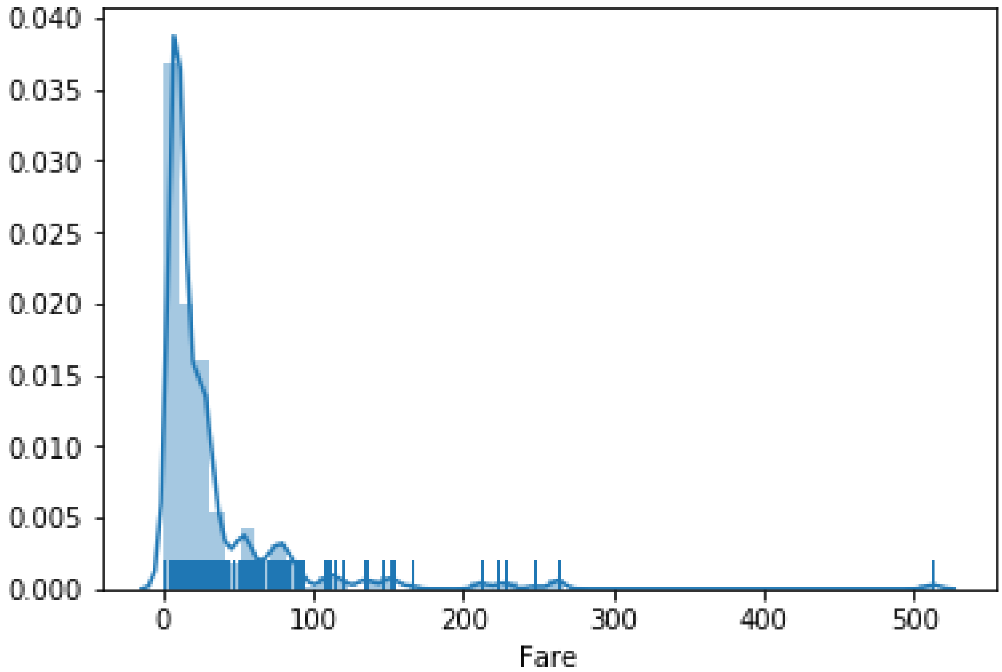
In [6]:
```
# grouping the age and fare

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

age = train1['Age'].dropna(axis=0)
age.to_string

sns.distplot(age,bins=10,kde=True,rug=True)

# there are two ways to handle this missing values either we can impute the data using
# we can remove those values
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x28e311ec0f0>



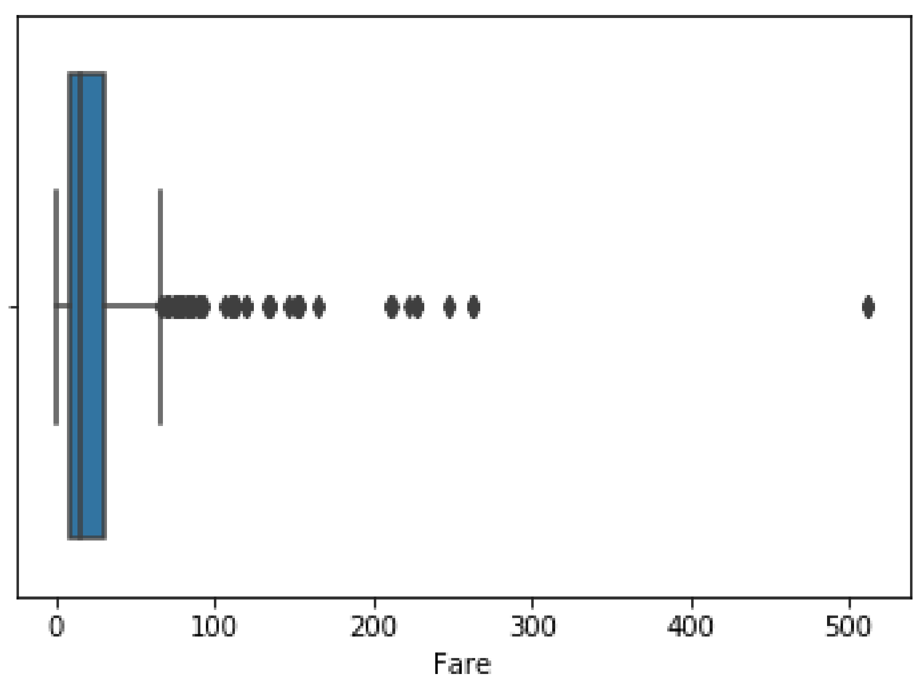
In [7]:
```
# distribution of the fare
fare = train1['Fare']
fare.to_string
sns.distplot(fare,kde=True,rug=True)
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x28e315111d0>

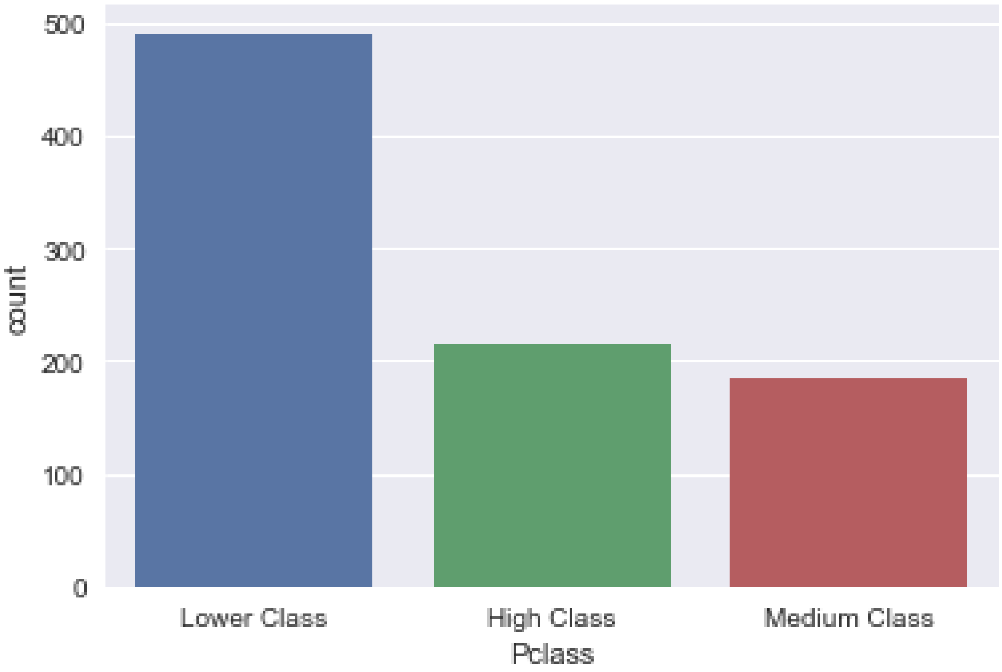


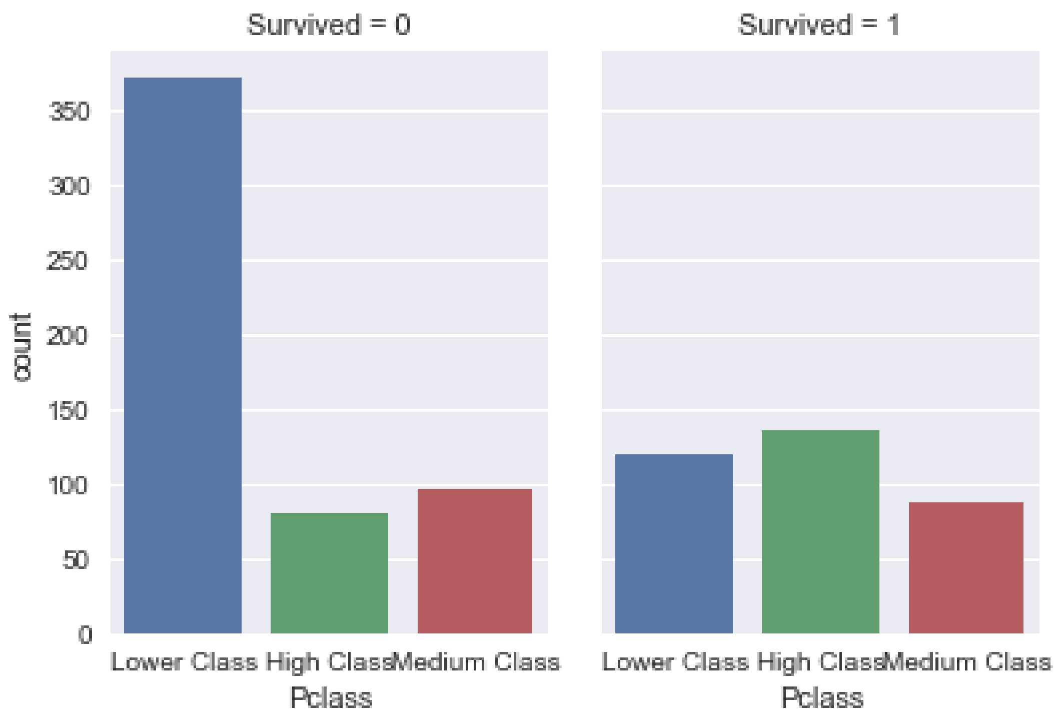

```
In [8]: sns.boxplot(fare)
```

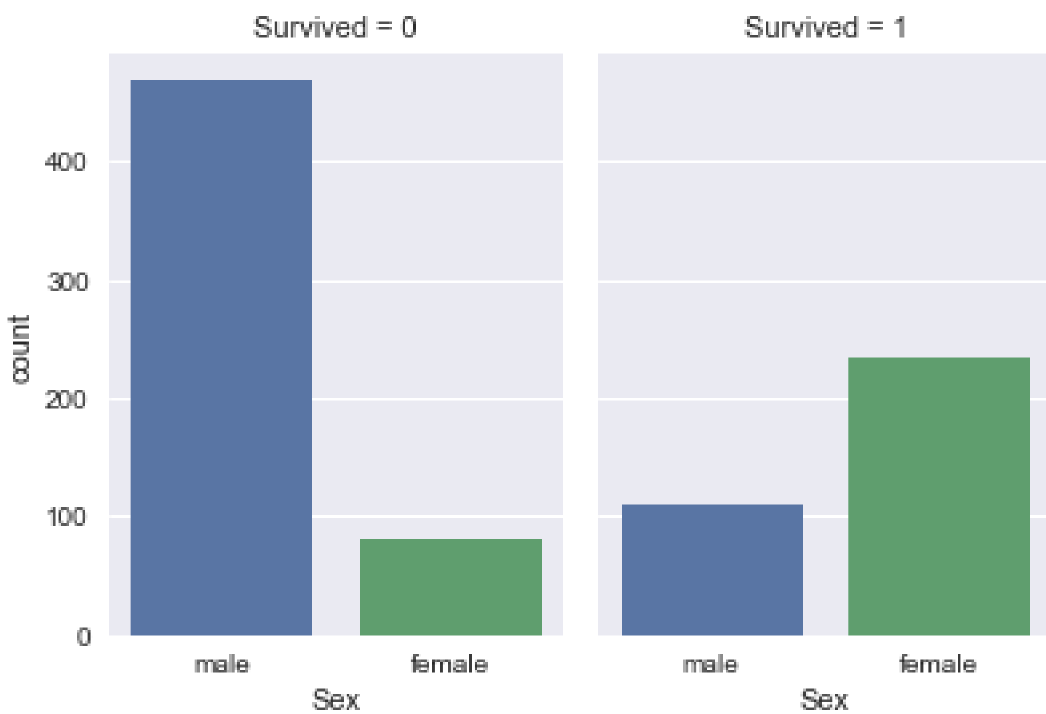
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x28e32695eb8>

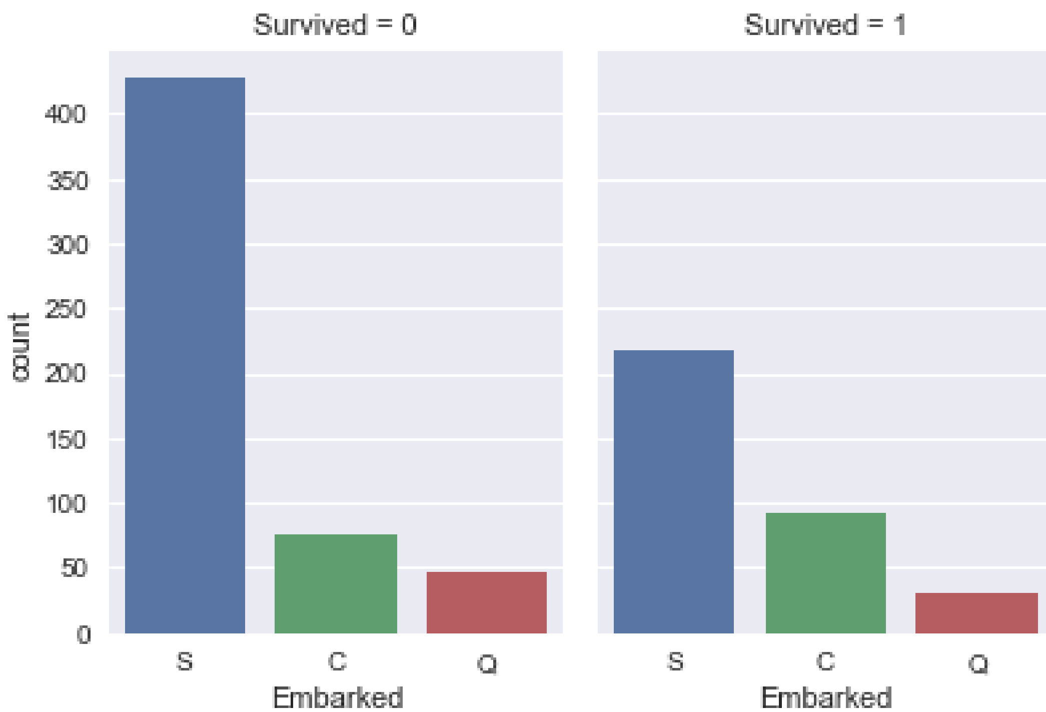


```
In [9]: sns.set(style="darkgrid")
        bar = sns.countplot(x="Pclass", data=train1)
        g = sns.factorplot(x="Pclass", col="Survived", data=train1, kind="count", size=4, aspec
        h = sns.factorplot(x="Sex", col="Survived", data=train1, kind="count", size=4, aspect=
        embarked = sns.factorplot(x="Embarked", col="Survived", data=train1, kind="count", size
```




Survived = 0
Survived = 1
count
350
300
250
200
150
100
50
0
Lower Class
High Class
Medium Class
Pclass


Survived = 0
Survived = 1
count
400
300
200
100
0
male
female
Sex



In [10]:
```
# importing the age through regression analysis

#target = train1.loc[:,'Survived']
#data_wo_survived = train1.iloc[:,1:]

agetest = train1[train1['Age'].isnull()]


train1['Age']  = train1['Age'].fillna(9999)

agetrain = train1[train1['Age'] != 9999]

agetrain.describe()


# running regression analysis


agetrain = pd.get_dummies(agetrain)

agetrain.columns
```

Out[10]:
```
Index(['Survived', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass_High Class',
       'Pclass_Lower Class', 'Pclass_Medium Class', 'Sex_female', 'Sex_male',
```

```
             'Embarked_C', 'Embarked_Q', 'Embarked_S'],
            dtype='object')
```

In [11]:
```
agetest.columns
```

Out[11]:
```
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
             'Embarked'],
            dtype='object')
```

In [12]:
```
agetest = agetest.iloc[:,[0,1,2,4,5,6,7]]
```

In [13]:
```
agetest.head()
agetest = pd.get_dummies(agetest)
```

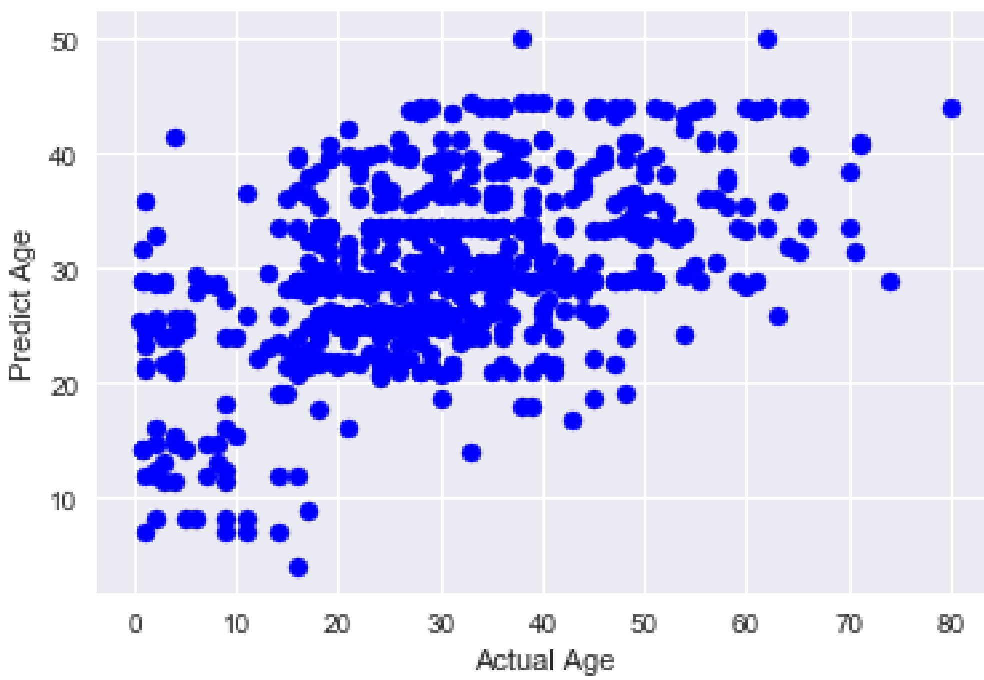
In [14]:
```
agetest.columns
```

Out[14]:
```
Index(['Survived', 'SibSp', 'Parch', 'Fare', 'Pclass_High Class',
             'Pclass_Lower Class', 'Pclass_Medium Class', 'Sex_female', 'Sex_male',
             'Embarked_C', 'Embarked_Q', 'Embarked_S'],
            dtype='object')
```

In [15]:
```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
survive_data_agetrain = agetrain.iloc[:,0]
survive_data_agetest = agetest.iloc[:,0]

agetest_wo_survive = agetest.iloc[:,1:]

agetrain_Y = agetrain.iloc[:,1]
agetrain_X = agetrain.iloc[:,2:]

regr = linear_model.LinearRegression()
regr.fit(agetrain_X, agetrain_Y)


# The coefficients
print('Coefficients: ', regr.coef_)
print('Intercept: ', regr.intercept_)


agetrain_Y_predict = regr.predict(agetrain_X)

# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(agetrain_Y, agetrain_Y_predict))

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(agetrain_Y, agetrain_Y_predict))
```

```
# Plot outputs

plt.scatter(agetrain_Y, agetrain_Y_predict, color='blue')
plt.xlabel('Actual Age')
plt.ylabel('Predict Age')

plt.show()
```

```
Coefficients:  [ -3.89188815  -0.7039223   -0.0221094    8.75649705  -6.7568533
  -1.99964375  -1.52891861   1.52891861 -12.99425641  -7.76204648
 -10.25152765]
Intercept:  44.5411738167
Variance score: 0.25
Mean squared error: 157.97
```




```
In [16]: # Using Statsmodels library to find the linear regression using OLS method.

         import statsmodels.api as sm
         import matplotlib.pyplot as plt


         model = sm.OLS(agetrain_Y, agetrain_X)
         results = model.fit()
         print(results.summary())
```

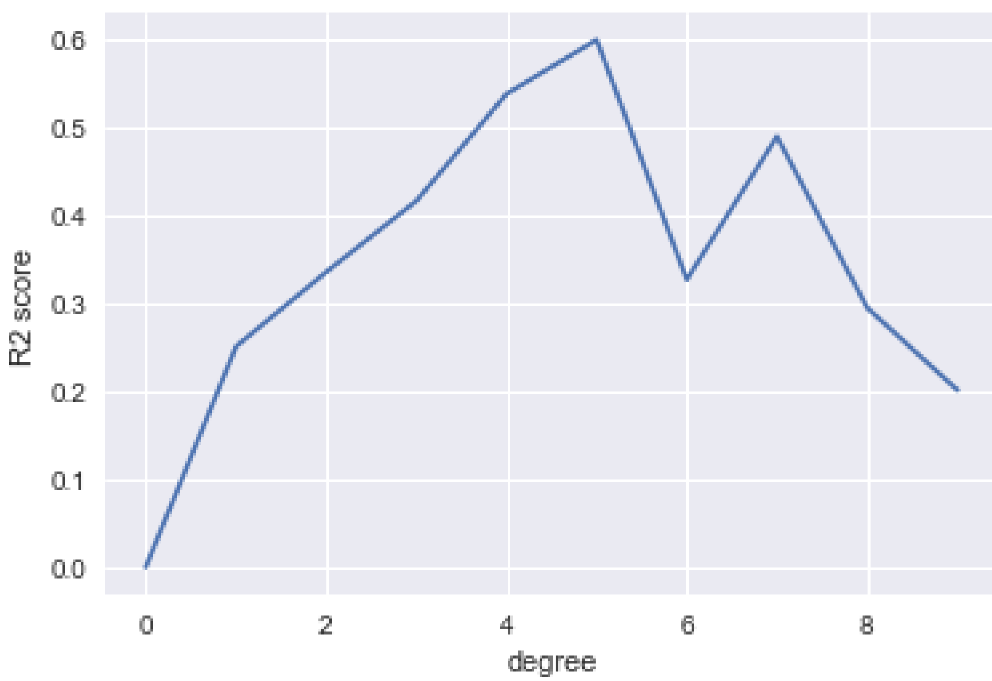
```
D:\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core
  from pandas.core import datetools
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Age   R-squared:                       0.250
Model:                            OLS   Adj. R-squared:                  0.241
Method:                 Least Squares   F-statistic:                     26.12
Date:                Mon, 13 Nov 2017   Prob (F-statistic):           5.18e-39
Time:                        02:48:53   Log-Likelihood:                -2820.4
No. Observations:                 714   AIC:                             5661.
Df Residuals:                     704   BIC:                             5707.
Df Model:                           9
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
SibSp                    -3.8919      0.559     -6.960      0.000      -4.990      -2.794
Parch                    -0.7039      0.631     -1.116      0.265      -1.943       0.535
Fare                     -0.0221      0.012     -1.861      0.063      -0.045       0.001
Pclass_High Class        26.5730      3.682      7.218      0.000      19.345      33.801
Pclass_Lower Class       11.0596      3.680      3.005      0.003       3.834      18.286
Pclass_Medium Class      15.8168      3.725      4.246      0.000       8.504      23.130
Sex_female               25.1958      5.402      4.664      0.000      14.589      35.803
Sex_male                 28.2536      5.454      5.180      0.000      17.546      38.961
Embarked_C              -12.9943      9.056     -1.435      0.152     -30.774       4.785
Embarked_Q               -7.7620      9.352     -0.830      0.407     -26.122      10.598
Embarked_S              -10.2515      9.045     -1.133      0.257     -28.010       7.507
==============================================================================
Omnibus:                       17.357   Durbin-Watson:                   1.898
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               18.019
Skew:                           0.365   Prob(JB):                     0.000122
Kurtosis:                       3.269   Cond. No.                     4.81e+17
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.23e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

1.0.2 Since the R2 is pretty low for the model, hence we will try for other regression methods.

In [17]:
```python
from sklearn.preprocessing import PolynomialFeatures

x = [0]
y = [0]
```

```
for i in np.arange(1,10,1):
    poly = PolynomialFeatures(degree=i,include_bias = False)
    X_poly = poly.fit_transform(agetrain_X)

    lin_reg = linear_model.LinearRegression()
    lin_reg.fit(X_poly,agetrain_Y)
    agetrain_Y_pred = lin_reg.predict(X_poly)
    x.append(i)
    a = r2_score(agetrain_Y, agetrain_Y_pred)
    y.append(a)

plt.plot(x,y)
plt.xlabel("degree")
plt.ylabel("R2 score")
plt.show()
```



1.0.3 Since at polynomial degree = 5 we are getting a suitable R2 score

1.0.4 ,therefore we will be using polynimial regression of degree 5, to calculate the missing age.

```
In [18]: poly = PolynomialFeatures(degree=5,include_bias = False)
         X_poly = poly.fit_transform(agetrain_X)
```

```
agetest_poly = poly.fit_transform(agetest_wo_survive)

lin_reg = linear_model.LinearRegression()
lin_reg.fit(X_poly,agetrain_Y)

agetest_Y_pred = lin_reg.predict(agetest_poly)

int_agetest_Y=[]
for i in agetest_Y_pred:
    int_agetest_Y.append(int(i))
```

In [19]:

```
age_pred = np.array(int_agetest_Y)
age_predict = pd.DataFrame(age_pred,columns=['Age'])
age_predict = age_predict.reset_index()
agetest_wo_survive = agetest_wo_survive.reset_index()
survive_data_agetest = survive_data_agetest.reset_index()

agetest1 = pd.concat([survive_data_agetest,agetest_wo_survive,age_predict],axis=1)
```

In [20]:

```
agetest1 = agetest1.drop(['index','level_0'],axis=1)
```

In [21]:

```
agetest1.head()
```

Out[21]:

	Survived	SibSp	Parch	Fare	Pclass_High Class	Pclass_Lower Class
0	0	0	0	8.4583	0	1
1	1	0	0	13.0000	0	0
2	1	0	0	7.2250	0	1
3	0	0	0	7.2250	0	1
4	1	0	0	7.8792	0	1

	Pclass_Medium Class	Sex_female	Sex_male	Embarked_C	Embarked_Q
0	0	0	1	0	1
1	1	0	1	0	0
2	0	1	0	1	0
3	0	0	1	1	0
4	0	1	0	0	1

	Embarked_S	Age
0	0	95
1	1	33
2	0	14
3	0	29
4	0	19

In [22]:

```
# dropping the rows with negative age
agetest1 = agetest1.drop(agetest1[agetest1.Age < 0].index)

# dropping the rows with age > 100
agetest1 = agetest1.drop(agetest1[agetest1.Age > 100].index)
```

```
In [23]: # joining the dataframe with age predicted and dataframe with available age.
         train_final = pd.concat([agetest1,agetrain])
```

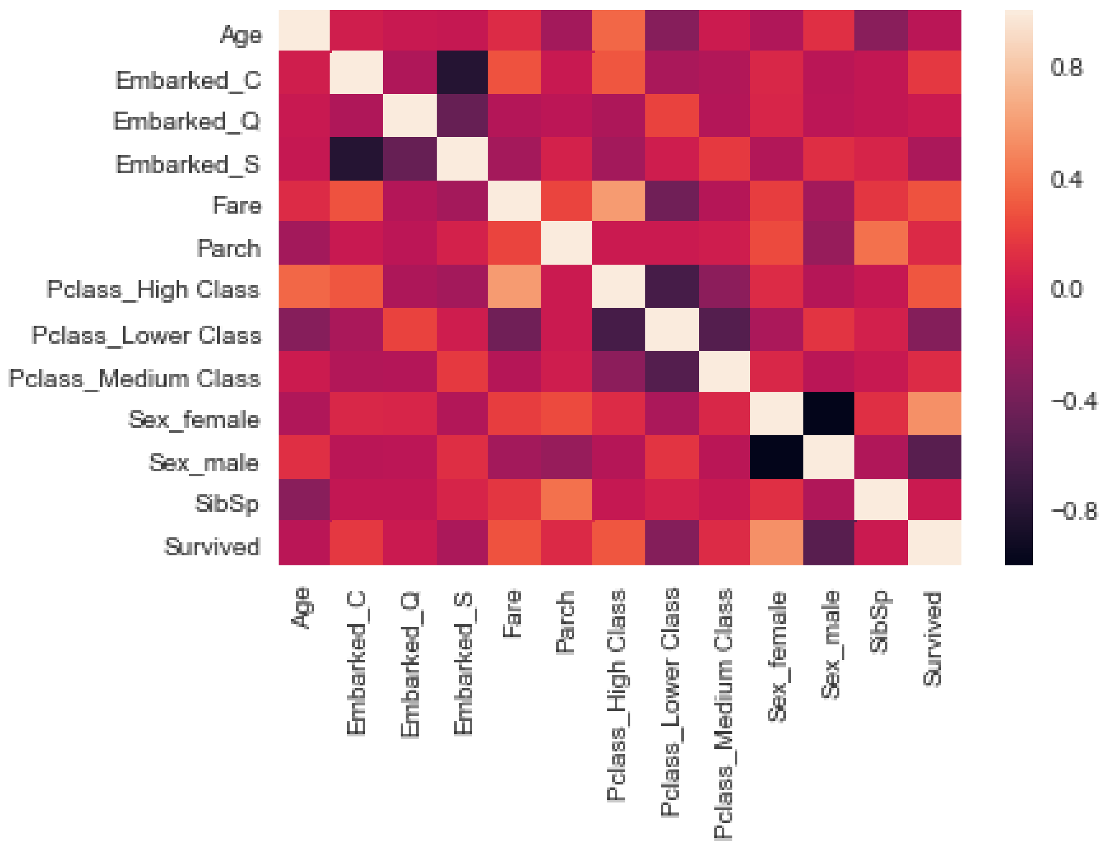
```
In [24]: train_final.head()
         #train_final.shape
```

```
Out[24]:    Age  Embarked_C  Embarked_Q  Embarked_S     Fare  Parch  \
         0  95.0           0           1           0   8.4583      0
         1  33.0           0           0           1  13.0000      0
         2  14.0           1           0           0   7.2250      0
         3  29.0           1           0           0   7.2250      0
         4  19.0           0           1           0   7.8792      0

            Pclass_High Class  Pclass_Lower Class  Pclass_Medium Class  Sex_female  \
         0                  0                   1                    0           0
         1                  0                   0                    1           0
         2                  0                   1                    0           1
         3                  0                   1                    0           0
         4                  0                   1                    0           1

            Sex_male  SibSp  Survived
         0         1      0         0
         1         1      0         1
         2         0      0         1
         3         1      0         0
         4         0      0         1
```

1.0.5 Checking for collinearity

```
In [25]: import seaborn as sb
         sb.heatmap(train_final.corr())
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x28e35fdb630>
```




```
In [26]: train_final.corr()
```

Out[26]:

	Age	Embarked_C	Embarked_Q	Embarked_S	Fare \
Age	1.000000	0.029511	-0.012771	-0.026256	0.102511
Embarked_C	0.029511	1.000000	-0.138387	-0.803142	0.273551
Embarked_Q	-0.012771	-0.138387	1.000000	-0.468976	-0.111601
Embarked_S	-0.026256	-0.803142	-0.468976	1.000000	-0.181780
Fare	0.102511	0.273551	-0.111601	-0.181780	1.000000
Parch	-0.189084	-0.010164	-0.070661	0.053395	0.219426
Pclass_High Class	0.365630	0.296553	-0.143063	-0.187558	0.590834
Pclass_Lower Class	-0.321185	-0.156641	0.215463	0.016988	-0.424403
Pclass_Medium Class	0.004650	-0.123952	-0.112235	0.179331	-0.108830
Sex_female	-0.128774	0.084137	0.070473	-0.123688	0.191660
Sex_male	0.128774	-0.084137	-0.070473	0.123688	-0.191660
SibSp	-0.305680	-0.043817	-0.041573	0.066288	0.158250
Survived	-0.081162	0.166877	-0.007497	-0.150330	0.274081

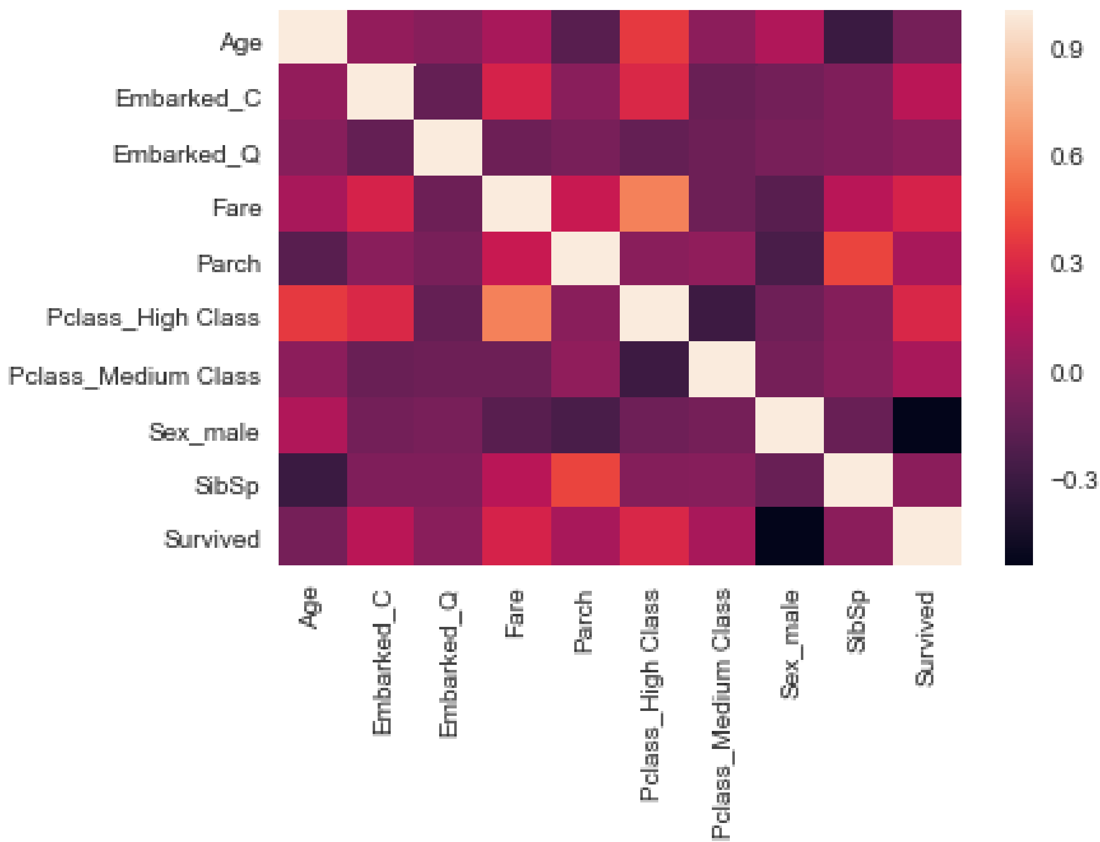
	Parch	Pclass_High Class	Pclass_Lower Class \
Age	-0.189084	0.365630	-0.321185
Embarked_C	-0.010164	0.296553	-0.156641
Embarked_Q	-0.070661	-0.143063	0.215463

```
Embarked_S           0.053395          -0.187558              0.016988
Fare                 0.219426           0.590834             -0.424403
Parch                1.000000          -0.006362             -0.007558
Pclass_High Class   -0.006362           1.000000             -0.629137
Pclass_Lower Class  -0.007558          -0.629137              1.000000
Pclass_Medium Class  0.016086          -0.293857             -0.558100
Sex_female           0.245051           0.103927             -0.153972
Sex_male            -0.245051          -0.103927              0.153972
SibSp                0.401943          -0.026003              0.041219
Survived             0.099255           0.290519             -0.335726

                     Pclass_Medium Class  Sex_female  Sex_male     SibSp  \
Age                             0.004650   -0.128774  0.128774 -0.305680
Embarked_C                     -0.123952    0.084137 -0.084137 -0.043817
Embarked_Q                     -0.112235    0.070473 -0.070473 -0.041573
Embarked_S                      0.179331   -0.123688  0.123688  0.066288
Fare                           -0.108830    0.191660 -0.191660  0.158250
Parch                           0.016086    0.245051 -0.245051  0.401943
Pclass_High Class              -0.293857    0.103927 -0.103927 -0.026003
Pclass_Lower Class             -0.558100   -0.153972  0.153972  0.041219
Pclass_Medium Class             1.000000    0.078398 -0.078398 -0.022929
Sex_female                      0.078398    1.000000 -1.000000  0.131345
Sex_male                       -0.078398   -1.000000  1.000000 -0.131345
SibSp                          -0.022929    0.131345 -0.131345  1.000000
Survived                        0.102714    0.540922 -0.540922 -0.000101

                     Survived
Age                 -0.081162
Embarked_C           0.166877
Embarked_Q          -0.007497
Embarked_S          -0.150330
Fare                 0.274081
Parch                0.099255
Pclass_High Class    0.290519
Pclass_Lower Class  -0.335726
Pclass_Medium Class  0.102714
Sex_female           0.540922
Sex_male            -0.540922
SibSp               -0.000101
Survived             1.000000
```

In [27]:
```
train_final = train_final.drop(['Embarked_S','Sex_female','Pclass_Lower Class'],axis=
```

In [28]:
```
sb.heatmap(train_final.corr())
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28e00167b38>
```



1.1 Logistic Regression

```
In [29]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score
         X_target = train_final.iloc[:,-1]
         X_var = train_final.iloc[:,:-1]

         logistic = LogisticRegression()
         logistic.fit(X_var,X_target)
         X_pred = logistic.predict(X_var)

         print("Accuracy : %.2f" % accuracy_score(X_target, X_pred))
```

```
Accuracy : 0.81
```

```
In [30]: import statsmodels.api as sm

         logit = sm.Logit(X_target,X_var)

         result = logit.fit()
```

```
print (result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.457173
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:               Survived   No. Observations:                  856
Model:                          Logit   Df Residuals:                      847
Method:                           MLE   Df Model:                            8
Date:                Mon, 13 Nov 2017   Pseudo R-squ.:                  0.3154
Time:                        02:52:01   Log-Likelihood:                -391.34
converged:                       True   LL-Null:                       -571.62
                                        LLR p-value:                 5.053e-73
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
Age                    -0.0139      0.005     -2.584      0.010      -0.024      -0.003
Embarked_C              0.6249      0.227      2.752      0.006       0.180       1.070
Embarked_Q              0.7134      0.328      2.173      0.030       0.070       1.357
Fare                    0.0039      0.003      1.384      0.166      -0.002       0.009
Parch                   0.0606      0.113      0.536      0.592      -0.161       0.282
Pclass_High Class       2.0512      0.307      6.681      0.000       1.449       2.653
Pclass_Medium Class     1.5197      0.229      6.638      0.000       1.071       1.968
Sex_male               -2.1921      0.183    -11.958      0.000      -2.551      -1.833
SibSp                  -0.1463      0.109     -1.345      0.179      -0.360       0.067
=======================================================================================
```

In [31]:
```
X_var = X_var.drop(['Fare','Parch','SibSp'],axis=1)
```

In [32]:
```
import statsmodels.api as sm

logit = sm.Logit(X_target,X_var)

result = logit.fit()

print (result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.459439
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:               Survived   No. Observations:                  856
Model:                          Logit   Df Residuals:                      850
Method:                           MLE   Df Model:                            5
Date:                Mon, 13 Nov 2017   Pseudo R-squ.:                  0.3120
```

```
Time:                        02:52:21   Log-Likelihood:               -393.28
converged:                       True   LL-Null:                      -571.62
                                        LLR p-value:                6.396e-75
=========================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Age                     -0.0137      0.005     -2.609      0.009      -0.024      -0.003
Embarked_C               0.6871      0.223      3.080      0.002       0.250       1.124
Embarked_Q               0.7290      0.326      2.236      0.025       0.090       1.368
Pclass_High Class        2.2868      0.253      9.024      0.000       1.790       2.783
Pclass_Medium Class      1.5722      0.225      6.983      0.000       1.131       2.013
Sex_male                -2.2241      0.181    -12.262      0.000      -2.580      -1.869
=========================================================================================
```

1.1.1 odds ratio

```
In [36]: import numpy as np
         print (np.exp(result.params))
```

```
Age                      0.986430
Embarked_C               1.987977
Embarked_Q               2.073054
Pclass_High Class        9.843000
Pclass_Medium Class      4.817069
Sex_male                 0.108160
dtype: float64
```

1.1.2 odds ratio and 95%CI

```
In [37]: # odds ratios and 95% CI
         params = result.params
         conf = result.conf_int()
         conf['OR'] = params
         conf.columns = ['2.5%', '97.5%', 'OR']
         print (np.exp(conf))
```

```
                         2.5%      97.5%        OR
Age                  0.976358   0.996606  0.986430
Embarked_C           1.283922   3.078107  1.987977
Embarked_Q           1.094328   3.927116  2.073054
Pclass_High Class    5.989871  16.174747  9.843000
Pclass_Medium Class  3.098452   7.488952  4.817069
Sex_male             0.075800   0.154336  0.108160
```

```
In [38]: logistic = LogisticRegression()
         logistic.fit(X_var,X_target)
```

```
X_pred = logistic.predict(X_var)

print("Accuracy : %.2f" % accuracy_score(X_target, X_pred))
```

```
Accuracy : 0.79
```

1.2 Confusion Matrix

```
In [39]: from sklearn.metrics import confusion_matrix

         confusion_matrix = confusion_matrix(X_target, X_pred)

         confusion_matrix

Out[39]: array([[447,  77],
                [103, 229]], dtype=int64)
```

1.3 Classification Report

```
In [40]: from sklearn import metrics
         from sklearn.metrics import classification_report

         print(classification_report(X_target, X_pred))
```

```
             precision    recall  f1-score   support

          0       0.81      0.85      0.83       524
          1       0.75      0.69      0.72       332

avg / total       0.79      0.79      0.79       856
```

```
In [43]: test.head()
         # dropping the irrelevant columns
         test1 = test.drop(['PassengerId','Name','Ticket','Cabin'], axis=1)

         # nice way to encode the categorical values
         cleanup_nums = {"Pclass":     {1: "High Class", 2:"Medium Class" , 3:"Lower Class"}}
         test1.replace(cleanup_nums, inplace=True)
         test1.head()

         test1 = pd.get_dummies(test1)

In [44]: test1.head()
         test1 = test1.drop(['Embarked_S','Sex_female','Pclass_Lower Class'],axis=1)

In [45]: test1 = test1.drop(['Fare','Parch','SibSp'],axis=1)
```

1.4 Predicting the values of test data

```
In [46]: test1 = test1.dropna()

         y_pred = logistic.predict(test1)
```

1.5 Predicted Values

```
In [47]: y_pred

Out[47]: array([0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
                1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1], dtype=int64)
```

```
In [48]: y = pd.DataFrame(y_pred,columns = ['Survived'])
         test_pred = pd.concat([y,test],axis=1)
```