In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')
```

In [3]:
```
train_data
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

In [4]:
```
test_data
```

Out[4]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

In [5]:
```
col = train_data.columns
print(col)
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [6]:
```
print('Train Data Info.:\n')
train_data.info()
```

```
Train Data Info.:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [7]:
```
print('Test Data Info:\n')
test_data.info()
```

```
Test Data Info:

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

In [8]:
```
print('Train Data:\n')
train_data.sum()
```

```
Train Data:
```

Out[8]:
```
PassengerId                                                   397386
Survived                                                         342
Pclass                                                          2057
Name           Braund, Mr. Owen HarrisCumings, Mrs. John Brad...
Sex            malefemalefemalefemalemalemalemalemalefemalefe...
Age                                                          21205.2
SibSp                                                            466
Parch                                                            340
Ticket         A/5 21171PC 17599STON/O2. 31012821138033734503...
Fare                                                         28693.9
dtype: object
```

In [9]:
```
print('Test Data:\n')
test_data.sum()
```

```
Test Data:
```

Out[9]:
```
PassengerId                                                   460009
Pclass                                                           947
Name           Kelly, Mr. JamesWilkes, Mrs. James (Ellen Need...
Sex            malefemalemalemalefemalemalefemalemalefemalema...
Age                                                          10050.5
SibSp                                                            187
Parch                                                            164
Ticket         33091136327224027631515431012987538330972248 73...
Fare                                                         14856.5
Embarked       QSQSSSQSCSSSSSSCQCSCCSSCCSCCSCSSSSCCSSSSCSSSSS...
dtype: object
```

In [10]:
```
print('Train Data Shape:\n')
train_data.shape
```

```
Train Data Shape:
```

Out[10]:
```
(891, 12)
```

In [11]:
```
print('Train Data Axes:\n')
train_data.axes
```

```
Train Data Axes:
```

Out[11]:
```
[RangeIndex(start=0, stop=891, step=1),
 Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
        'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
       dtype='object')]
```

In [12]:
```
print('Test Data Shape and axes:\n')
test_data.shape
```

```
Test Data Shape and axes:
```

Out[12]:
```
(418, 11)
```

In [13]:
```
print('Test Data Axes:\n')
test_data.axes
```

```
Test Data Axes:
```

Out[13]:
```
[RangeIndex(start=0, stop=418, step=1),
 Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
        'Ticket', 'Fare', 'Cabin', 'Embarked'],
       dtype='object')]
```

In [14]:
```
print('To get Description of Train Data :\n')
train_data.describe()
```

```
To get Description of Train Data :
```

Out[14]:

	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 [15]:
```
print('To get Description of Test Data :\n')
test_data.describe()
```

```
To get Description of Test Data :
```

Out[15]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [16]:
```
# To check which columns have nan value
train_data.isna().sum()
```

Out[16]:
```
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 [17]:
```
# Survival based on SEX of passenger.

print("Total Females  = ", len(train_data[(train_data['Sex'] == 'female')].index))
print("Total Males  = ", len(train_data[(train_data['Sex'] == 'male')].index))
```

```
Total Females  =  314
Total Males  =  577
```

In [18]:
```
survived_males = train_data[(train_data['Sex'] == 'male') & (train_data['Survived']
== 1)]
survived_females = train_data[(train_data['Sex'] == 'female') & (train_data['Surviv
ed'] == 1)]
print("Total Males Survived = ",len(survived_males.index))
print("Total Females Survived = ",len(survived_females.index))
```

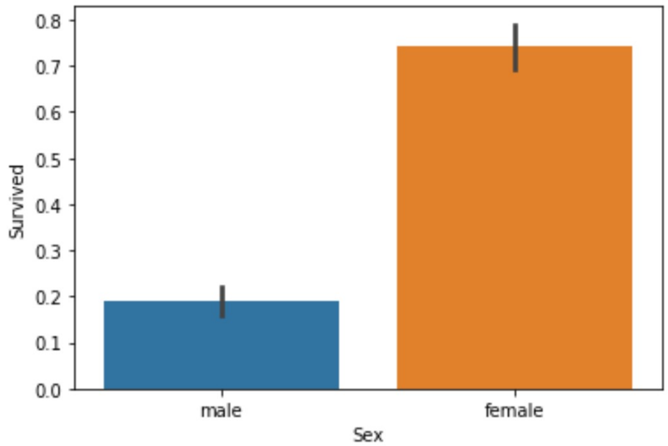
```
Total Males Survived =  109
Total Females Survived =  233
```

In [19]:
```
print([train_data.groupby('Sex')['Survived'].value_counts(normalize=True)])
```

```
[Sex     Survived
female  1           0.742038
        0           0.257962
male    0           0.811092
        1           0.188908
Name: Survived, dtype: float64]
```

In [20]:
```
sns.barplot(x='Sex', y='Survived', data=train_data)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce4e467508>



In [21]:
```
# Survival based on the PASSENGER CLASS.

print("Pclass 1 = ", len(train_data[(train_data['Pclass'] == 1)].index))
print("Pclass 2 = ", len(train_data[(train_data['Pclass'] == 2)].index))
print("Pclass 3 = ", len(train_data[(train_data['Pclass'] == 3)].index))
```

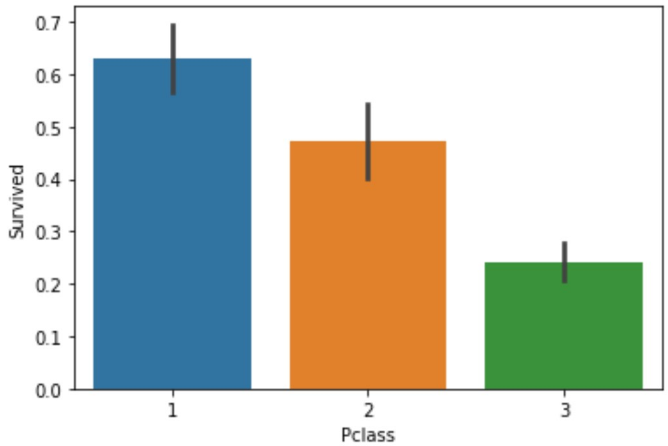
```
Pclass 1 =  216
Pclass 2 =  184
Pclass 3 =  491
```

In [22]:
```
survived_Pclass1 = train_data[(train_data['Pclass'] == 1) & (train_data['Survived']
== 1)]
survived_Pclass2 = train_data[(train_data['Pclass'] == 2) & (train_data['Survived']
== 1)]
survived_Pclass3 = train_data[(train_data['Pclass'] == 3) & (train_data['Survived']
== 1)]
print("Total Pclass-1 Survived  = ",len(survived_Pclass1.index))
print("Total Pclass-2 Survived  = ",len(survived_Pclass2.index))
print("Total Pclass-3 Survived  = ",len(survived_Pclass3.index))
```

```
Total Pclass-1 Survived  =  136
Total Pclass-2 Survived  =  87
Total Pclass-3 Survived  =  119
```

In [23]:
```
sns.barplot(x='Pclass', y='Survived', data=train_data)
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce4ec0bd88>



In [24]:
```
# Survival Based On PASSENGER_CLASS and SEX.
survived_Pclass1_males   = train_data[(train_data['Pclass'] == 1) & (train_data['Se
x'] == 'male')   & (train_data['Survived'] == 1)]
survived_Pclass1_females = train_data[(train_data['Pclass'] == 1) & (train_data['Se
x'] == 'female') & (train_data['Survived'] == 1)]
survived_Pclass2_males   = train_data[(train_data['Pclass'] == 2) & (train_data['Se
x'] == 'male')   & (train_data['Survived'] == 1)]
survived_Pclass2_females = train_data[(train_data['Pclass'] == 2) & (train_data['Se
x'] == 'female') & (train_data['Survived'] == 1)]
survived_Pclass3_males   = train_data[(train_data['Pclass'] == 3) & (train_data['Se
x'] == 'male') & (train_data['Survived'] == 1)]
survived_Pclass3_females = train_data[(train_data['Pclass'] == 3) & (train_data['Se
x'] == 'female') & (train_data['Survived'] == 1)]

print("Total Pclass-1-Males Survived    = ",len(survived_Pclass1_males.index))
print("Total Pclass-1-Females Survived  = ",len(survived_Pclass1_females.index))
print("Total Pclass-2-Males Survived    = ",len(survived_Pclass2_males.index))
print("Total Pclass-2-Females Survived  = ",len(survived_Pclass2_females.index))
print("Total Pclass-3-Males Survived    = ",len(survived_Pclass3_males.index))
print("Total Pclass-3-Females Survived  = ",len(survived_Pclass3_females.index))
```

```
Total Pclass-1-Males Survived    =  45
Total Pclass-1-Females Survived  =  91
Total Pclass-2-Males Survived    =  17
Total Pclass-2-Females Survived  =  70
Total Pclass-3-Males Survived    =  47
Total Pclass-3-Females Survived  =  72
```

In [25]:
```
print([train_data.groupby(['Pclass','Sex'])['Survived'].value_counts(normalize=True)])
```

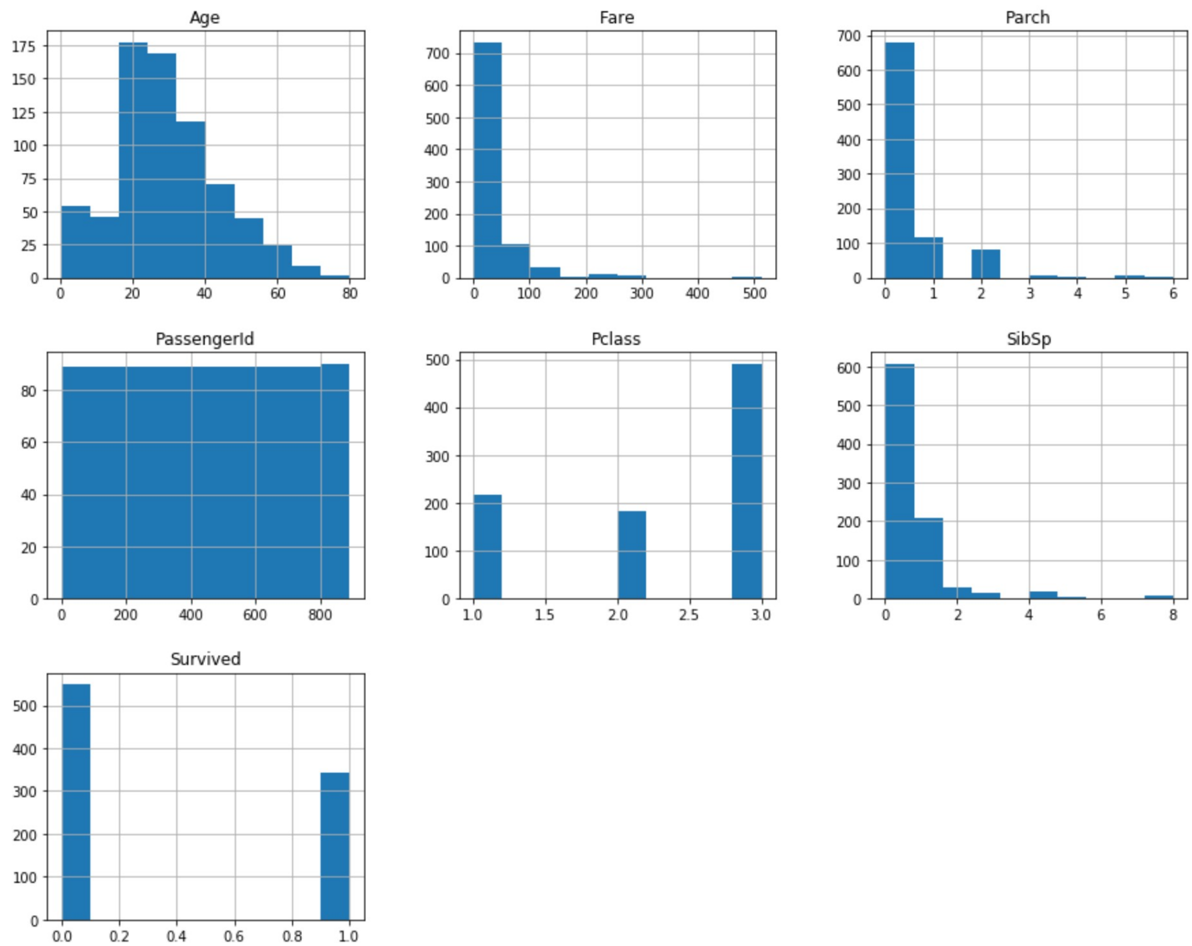
```
[Pclass  Sex     Survived
1       female  1           0.968085
                0           0.031915
        male    0           0.631148
                1           0.368852
2       female  1           0.921053
                0           0.078947
        male    0           0.842593
                1           0.157407
3       female  0           0.500000
                1           0.500000
        male    0           0.864553
                1           0.135447
Name: Survived, dtype: float64]
```

In [26]:

```
# Histogram of Training Data

train_data.hist(figsize=(15,12))
```

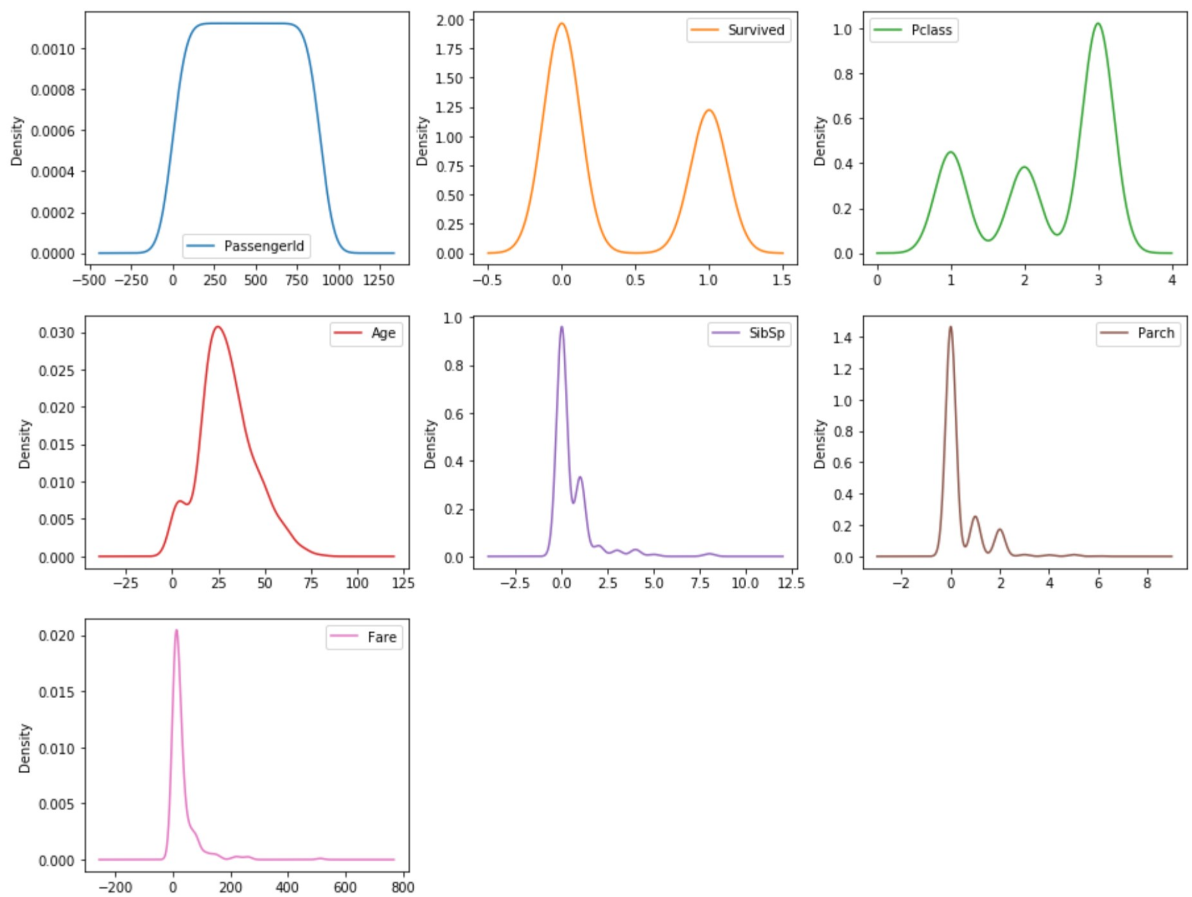
Out[26]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4EC70F08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4ECE7C48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4ED21908>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4ED59A08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4ED91B08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4EDCABC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4EE03CC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4EE3BDC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4EE459C8>]],
      dtype=object)
```



In [27]:
```
# Density Graph of Training Data

train_data.plot(kind='density',subplots=True, layout=(3, 3),sharex=False, figsize=
(15,12))
```

Out[27]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F4461C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F4B6688>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F07EE48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F0BA888>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F0F2288>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F128C48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F49C708>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F18E808>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F19A408>]],
      dtype=object)
```

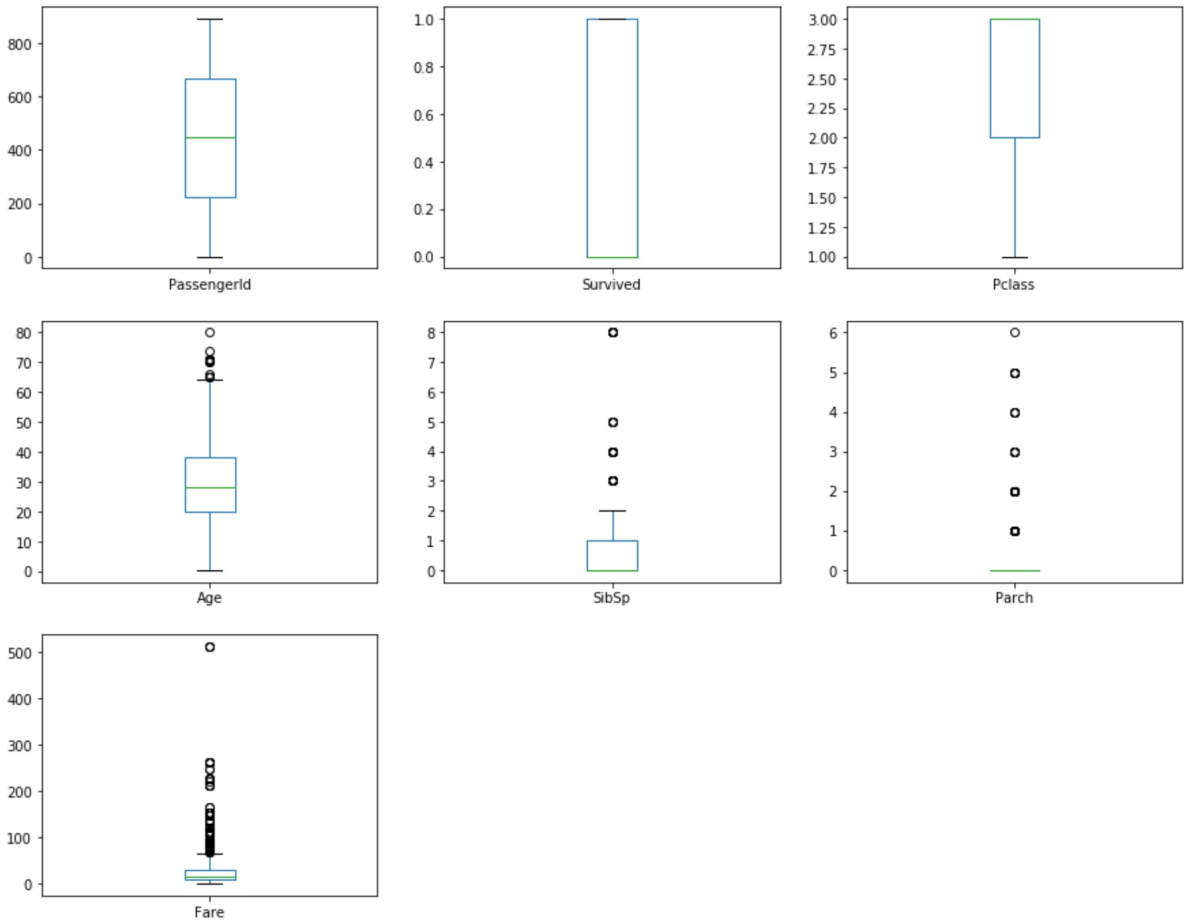


In [28]:

```
# Boxplot of Training Data to find out the outliers.

train_data.plot(kind='box',subplots=True, layout=(3, 3),sharex=False, figsize=(15,1
2))
```

Out[28]:

```
PassengerId       AxesSubplot(0.125,0.657941;0.227941x0.222059)
Survived       AxesSubplot(0.398529,0.657941;0.227941x0.222059)
Pclass         AxesSubplot(0.672059,0.657941;0.227941x0.222059)
Age               AxesSubplot(0.125,0.391471;0.227941x0.222059)
SibSp          AxesSubplot(0.398529,0.391471;0.227941x0.222059)
Parch          AxesSubplot(0.672059,0.391471;0.227941x0.222059)
Fare                  AxesSubplot(0.125,0.125;0.227941x0.222059)
dtype: object
```



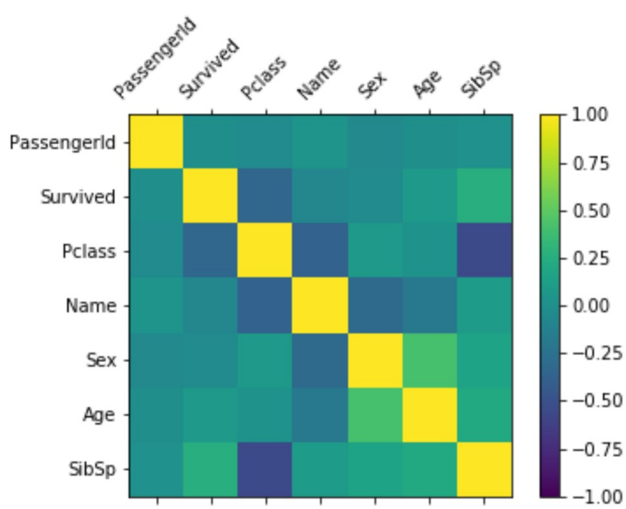
In [29]:
```
# Now, we will find out the co-relation between all the columns.

corr_mat = train_data.corr()
corr_mat
```

Out[29]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

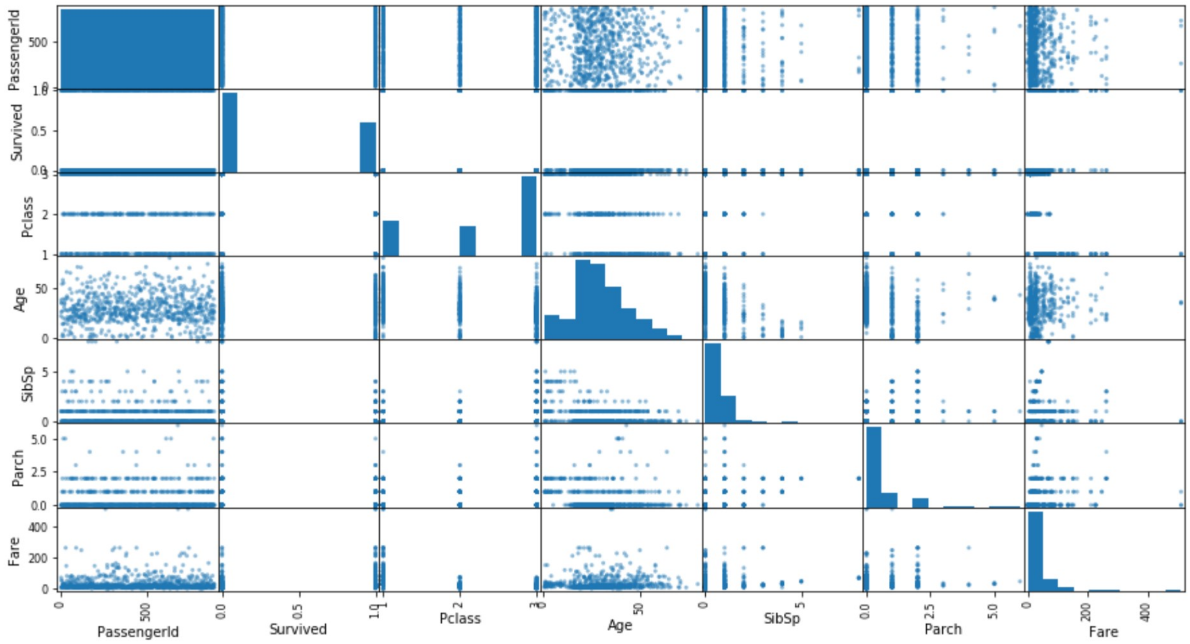
In [30]:
```
import numpy as np
# Plotting the co-relation between the column.

fig = plt.figure(4)
ax = fig.add_subplot(111)
cax = ax.matshow(corr_mat, vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(7)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(train_data.columns,  rotation=45)
ax.set_yticklabels(train_data.columns)
```

Out[30]:
```
[Text(0, 0, 'PassengerId'),
 Text(0, 0, 'Survived'),
 Text(0, 0, 'Pclass'),
 Text(0, 0, 'Name'),
 Text(0, 0, 'Sex'),
 Text(0, 0, 'Age'),
 Text(0, 0, 'SibSp')]
```



In [31]:
```
# Scatter_Matrix for training data.

from pandas.plotting import scatter_matrix
scatter_matrix(train_data,figsize=(15,8))
```

Out[31]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4F994E48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FDE3C88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FE20208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FE59108>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FE91248>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FEC9348>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FF01408>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FF39548>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FF47148>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FF7E308>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE4FFE5888>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5001C908>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50057A08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50090B48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50348C48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5037FD48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE503B8E08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE503D8988>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50412A88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5044BBC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50483CC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE504BCDC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE504F6EC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5052EFC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50569108>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE505A3248>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE505DD388>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50615408>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5064D508>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5068CE48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE506C0748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE506F8848>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50731908>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5076BA08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE507A3B08>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE507DAC48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50813D48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5084DE08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE50884F48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE508C2088>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE508FB188>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE509362C8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CE5096C3C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE509A4448>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE519AF588>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE519E9E88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE51A1E788>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE51A588C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CE51A91948>]],
      dtype=object)
```



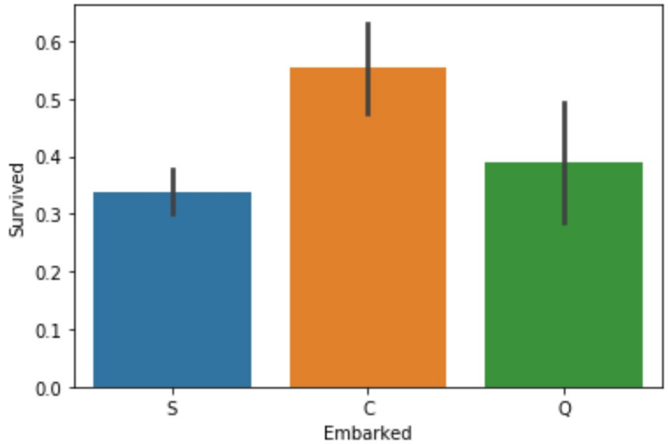
In [32]:
```
# Filling Missing Value.

# As there are many missing values in AGE column of Training Data.
# So, we fill all values by their mean.
train_data_new = train_data
train_data_new['Age'] = train_data_new['Age'].fillna(train_data['Age'].mean(), inpl
ace = False)

# The Embarked feature has only 2 missing values.
# We will just fill these with the most common one which is 'S'.
train_data_new.loc[train_data_new['Embarked'].isnull(),'Embarked'] = 'S'


# Cabin have non integer values so we can't fill them.
```

In [33]:
```
# Now again we will check the missing values in NEW TRAINING DATA.
train_data_new.isnull().sum()
```

Out[33]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```

In [34]:
```
# Plot on the basis of Embarked and Survived.
sns.barplot(x='Embarked', y='Survived', data=train_data_new)
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce5211bec8>



In [35]:
```
X_data = train_data_new.loc[:,['Age', 'Sex', 'Pclass', 'Embarked', 'SibSp', 'Fare
','Ticket','Parch']]
Y_data = train_data_new.loc[:,'Survived']
X_data
```

Out[35]:

	Age	Sex	Pclass	Embarked	SibSp	Fare	Ticket	Parch
0	22.000000	male	3	S	1	7.2500	A/5 21171	0
1	38.000000	female	1	C	1	71.2833	PC 17599	0
2	26.000000	female	3	S	0	7.9250	STON/O2. 3101282	0
3	35.000000	female	1	S	1	53.1000	113803	0
4	35.000000	male	3	S	0	8.0500	373450	0
...	...	...	...	...	...	...	...	...
886	27.000000	male	2	S	0	13.0000	211536	0
887	19.000000	female	1	S	0	30.0000	112053	0
888	29.699118	female	3	S	1	23.4500	W./C. 6607	2
889	26.000000	male	1	C	0	30.0000	111369	0
890	32.000000	male	3	Q	0	7.7500	370376	0

891 rows × 8 columns

In [36]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

le = LabelEncoder()      # Label Encoding
X_data.loc[:, 'Sex'] = le.fit_transform(X_data.loc[:,'Sex'])
X_data.loc[:, 'Embarked'] = le.fit_transform(X_data.loc[:, 'Embarked'])
X_data.loc[:, 'Fare'] = le.fit_transform(X_data.loc[:, 'Fare'])
X_data.loc[:, 'Ticket'] = le.fit_transform(X_data.loc[:, 'Ticket'])
X_data
```

Out[36]:

	Age	Sex	Pclass	Embarked	SibSp	Fare	Ticket	Parch
0	22.000000	1	3	2	1	18	523	0
1	38.000000	0	1	0	1	207	596	0
2	26.000000	0	3	2	0	41	669	0
3	35.000000	0	1	2	1	189	49	0
4	35.000000	1	3	2	0	43	472	0
...	...	...	...	...	...	...	...	...
886	27.000000	1	2	2	0	85	101	0
887	19.000000	0	1	2	0	153	14	0
888	29.699118	0	3	2	1	131	675	2
889	26.000000	1	1	0	0	153	8	0
890	32.000000	1	3	1	0	30	466	0

891 rows × 8 columns

In [37]:
```
# Showing Dummies variable
X_data_du = pd.get_dummies(X_data, columns=['Pclass','Sex','Embarked','SibSp'])
X_data_du.head(5)
```

Out[37]:

	Age	Fare	Ticket	Parch	Pclass_1	Pclass_2	Pclass_3	Sex_0	Sex_1	Embarked_0	Embarked_1	Embarke
0	22.0	18	523	0	0	0	1	0	1	0	0	
1	38.0	207	596	0	1	0	0	1	0	1	0	
2	26.0	41	669	0	0	0	1	1	0	0	0	
3	35.0	189	49	0	1	0	0	1	0	0	0	
4	35.0	43	472	0	0	0	1	0	1	0	0	

In [38]:
```
# Split data to training data and of test to check the accuracy of our model
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.3,
random_state=0)
```

START MODELING

In [39]:

```
# Start Modeling
# 1. Logistic Regrssion.
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

# TO perform features scaling to convert all data in a range.

scale = StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.fit_transform(X_test)

# Creating the model.
model_1 = LogisticRegression(random_state=0)
model_1.fit(X_train, Y_train)

# Predicting the model
pred_y = model_1.predict(X_test)
model_1_accu = accuracy_score(Y_test,pred_y)

print("Accuracy of First Model is {}".format(model_1_accu))

model_1_cfm = confusion_matrix(Y_test, pred_y)

print("Confusion Matrix of First Model is:-\n",model_1_cfm)
```

```
Accuracy of First Model is 0.8134328358208955
Confusion Matrix of First Model is:-
 [[144  24]
 [ 26  74]]
```

In [40]:
```
# 2. Univariate Feature Scaling By Chi2 method.

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

test = SelectKBest(chi2, k=4)  # how many best column you want to select
fit = test.fit(X_data, Y_data)   # Fitting the data.
selected_x = fit.transform(X_data)


training_x, testing_x, training_y, testing_y = train_test_split(selected_x, Y_data,
test_size=0.3, random_state=0)

scale = StandardScaler()
training_x = scale.fit_transform(training_x)
testing_x = scale.fit_transform(testing_x)

model_2 = LogisticRegression()
model_2.fit(training_x, training_y)
pred_y_2 = model_2.predict(testing_x)



model_2_accu = accuracy_score(testing_y, pred_y_2)

print("Accuracy of Second Model is {}".format(model_2_accu))

model_2_cfm = confusion_matrix(testing_y, pred_y)

print("Confusion Matrix of Second Model is:-\n",model_2_cfm)
```

```
Accuracy of Second Model is 0.7798507462686567
Confusion Matrix of Second Model is:-
 [[144  24]
 [ 26  74]]
```

In [42]:
```
from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [43]:
```
# 3. Recursive Feature Elimination

model_lr = LogisticRegression()
from sklearn.feature_selection import RFE

rfe = RFE(model_lr, 4)
fit = rfe.fit(X_data, Y_data)
recur_x = fit.transform(X_data)
#print(fit.ranking_)
training_x, testing_x, training_y, testing_y = train_test_split(recur_x, Y_data, te
st_size=0.3, random_state=0)

model_3 = LogisticRegression()
model_3.fit(training_x, training_y)
pred_y_3 = model_3.predict(testing_x)

model_3_accu = accuracy_score(testing_y, pred_y_3)

print("Accuracy of Third Model is {}".format(model_3_accu))

model_3_cfm = confusion_matrix(testing_y, pred_y_3)

print("Confusion Matrix of Third Model is:-\n",model_3_cfm)
```

```
Accuracy of Third Model is 0.7947761194029851
Confusion Matrix of Third Model is:-
 [[139  29]
 [ 26  74]]
```

In [44]:
```
# 4. Principle Component Analysis.

from sklearn.decomposition import PCA

pca = PCA(n_components=3)
fit_pca = pca.fit(X_data)

pca_x = fit.transform(X_data)

training_x, testing_x, training_y, testing_y = train_test_split(pca_x, Y_data, test
_size=0.3, random_state=7)

scale = StandardScaler()
training_x = scale.fit_transform(training_x)
testing_x = scale.fit_transform(testing_x)


model_4 = LogisticRegression()
model_4.fit(training_x, training_y)

pred_y_4 = model_4.predict(testing_x)

model_4_accu = accuracy_score(testing_y, pred_y_4)

print("Accuracy of Third Model is {}".format(model_4_accu))

model_4_cfm = confusion_matrix(testing_y, pred_y_4)

print("Confusion Matrix of Third Model is:-\n",model_4_cfm)
```

```
Accuracy of Third Model is 0.753731343283582
Confusion Matrix of Third Model is:-
 [[133  23]
 [ 43  69]]
```

In [45]:
```
# 5. K-Fold Cross Validation
#K-Fold Cross Validation randomly splits the training data into K subsets called fo
lds.


from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score


model_5 = LogisticRegression()

kfold = KFold(n_splits=10, random_state=0)
result_KFCV = cross_val_score(model_5,X_data,Y_data,cv=kfold)

print("Scores:", result_KFCV)
print("Mean:", result_KFCV.mean())
print("Standard Deviation:", result_KFCV.std())
```

```
Scores: [0.77777778 0.80898876 0.76404494 0.86516854 0.7752809  0.79775281
 0.78651685 0.7752809  0.85393258 0.79775281]
Mean: 0.8002496878901374
Standard Deviation: 0.032305406221494866
```

In [46]:
```
# 6. Leave one out cross validation
from sklearn.model_selection import LeaveOneOut

model_6 = LogisticRegression()

leaveone = LeaveOneOut()

result_LOO = cross_val_score(model_6,X_data,Y_data,cv=leaveone)
print("Scores", result_LOO)
print("Mean:", result_LOO.mean())
print("Standard Deviation:", result_LOO.std())
```

```
Scores [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 0. 1. 0.
 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1.
 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 0. 1. 1. 1. 0. 0.
 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 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. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1.
 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1.
 0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 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. 0. 1. 1. 1.
 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1. 0. 0. 1. 0.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0.
 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 1.
 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1.
 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0. 1. 1. 1. 0. 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. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 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. 0. 1. 1.
 0. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0.
 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 0. 1.]
Mean: 0.7957351290684624
Standard Deviation: 0.4031634078569878
```

In [47]:

```
# 7. RepeatedRandom train test splits
model_7 = LogisticRegression()

from sklearn.model_selection import ShuffleSplit

shuff = ShuffleSplit(n_splits=10, test_size=0.3, random_state=0)


from sklearn.model_selection import cross_val_score

result_shuff = cross_val_score(model_7,X_data,Y_data,cv=shuff)

print("Scores:", result_shuff)
print("Mean:", result_shuff.mean())
print("Standard Deviation:", result_shuff.std())
```

```
Scores: [0.80970149 0.79477612 0.80597015 0.76865672 0.81716418 0.79477612
 0.82089552 0.76492537 0.79850746 0.81716418]
Mean: 0.7992537313432837
Standard Deviation: 0.018491808497558
```

In [48]:

```
# 8. Support Vector Machine

from sklearn.svm import SVC


model_8= SVC()
model_8.fit(X_train, Y_train)

# Predicting the model
pred_y_8 = model_8.predict(X_test)
model_8_accu = accuracy_score(Y_test,pred_y_8)

print("Accuracy of Eight Model is {}".format(model_8_accu))

model_8_cfm = confusion_matrix(Y_test, pred_y_8)

print("Confusion Matri of Eight Model is:-\n",model_8_cfm)
```

```
Accuracy of Eight Model is 0.8097014925373134
Confusion Matri of Eight Model is:-
 [[145  23]
 [ 28  72]]
```

In [49]:
```
#9. Gaussian Naive Bayes algorithm

from sklearn.naive_bayes import GaussianNB

model_9 = GaussianNB()
model_9.fit(X_train, Y_train)

pred_y_9 = model_9.predict(X_test)

model_9_accu = accuracy_score(Y_test,pred_y_9)

print("Accuracy of Ninth Model is {}".format(model_9_accu))

model_9_cfm = confusion_matrix(Y_test, pred_y_9)

print("Confusion Matrix of Ninth Model is:-\n",model_9_cfm)
```

```
Accuracy of Ninth Model is 0.7798507462686567
Confusion Matrix of Ninth Model is:-
 [[128  40]
 [ 19  81]]
```

In [50]:
```
# 10. KNN(K-Nearest Neighbours)

from sklearn.neighbors import KNeighborsClassifier

training_x, testing_x, training_y, testing_y = train_test_split(X_data, Y_data, tes
t_size=0.25, random_state=0)


scale = StandardScaler()
training_x = scale.fit_transform(training_x)
testing_x = scale.fit_transform(testing_x)

model_10 = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
model_10.fit(training_x, training_y)

pred_y_10 = model_10.predict(testing_x)

model_10_accu = accuracy_score(testing_y,pred_y_10)

print("Accuracy of Ninth Model is {}".format(model_10_accu))

model_10_cfm = confusion_matrix(testing_y, pred_y_10)

print("Confusion Matrix of Ninth Model is:-\n",model_10_cfm)
```

```
Accuracy of Ninth Model is 0.8026905829596412
Confusion Matrix of Ninth Model is:-
 [[118  21]
 [ 23  61]]
```

In [55]:
```
# 11. Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor

training_x, testing_x, training_y, testing_y = train_test_split(X_data, Y_data, tes
t_size=0.25, random_state=0)

scale = StandardScaler()
training_x = scale.fit_transform(training_x)
testing_x = scale.fit_transform(testing_x)

model_11 = DecisionTreeRegressor(random_state=0)        # Using Decision Tree
model_11.fit(training_x, training_y)

pred_y_11 = model_11.predict(testing_x)

model_11_accu = accuracy_score(testing_y,pred_y_11.round())

print("Accuracy of Ninth Model is {}".format(model_11_accu))

model_11_cfm = confusion_matrix(testing_y, pred_y_11.round())

print("Confusion Matrix of Ninth Model is:-\n",model_11_cfm)

from sklearn.metrics import classification_report
print(classification_report(testing_y,pred_y_11.round()))
```

```
Accuracy of Ninth Model is 0.7533632286995515
Confusion Matrix of Ninth Model is:-
 [[108  31]
 [ 24  60]]
              precision    recall  f1-score   support

           0       0.82      0.78      0.80       139
           1       0.66      0.71      0.69        84

    accuracy                           0.75       223
   macro avg       0.74      0.75      0.74       223
weighted avg       0.76      0.75      0.76       223
```

In [52]:
```
# 12. RandomForestClassifier with K fold cross validation.
from sklearn.ensemble import RandomForestClassifier

model_12 = RandomForestClassifier(n_estimators=100)


from sklearn.model_selection import KFold

kfold = KFold(n_splits=10, random_state=0)

from sklearn.model_selection import cross_val_score




result = cross_val_score(model_12,X_data,Y_data,cv=kfold)
print(result)
print(result.mean())
```

```
[0.82222222 0.78651685 0.78651685 0.84269663 0.87640449 0.85393258
 0.84269663 0.79775281 0.88764045 0.84269663]
0.8339076154806492
```

In [56]:

```
# 13. RandomForestClassifier with Leaveoneout method.
from sklearn.ensemble import RandomForestClassifier

model_13 = RandomForestClassifier(n_estimators=100)

from sklearn.model_selection import LeaveOneOut
leaveone = LeaveOneOut()

from sklearn.model_selection import cross_val_score

result_LOO = cross_val_score(model_13,X_data,Y_data,cv=leaveone)
print("Scores",result_LOO)
print("Mean:", result_LOO.mean())
print("Standard Deviation:", result_LOO.std())
```

```
Scores [1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 0. 1. 0.
 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 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. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1.
 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 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. 0. 0. 1.
 0. 1. 1. 0. 1. 1. 0. 0. 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. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 0. 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. 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. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 0. 1. 1. 0.
 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. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 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. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1.
 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 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. 0. 1. 1. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 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. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0.
 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. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 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.
 1. 1. 1.]
Mean: 0.835016835016835
Standard Deviation: 0.371165354330523
```

In [57]:
```
test_data.head(10)
```

Out[57]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C
9	901	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S

In [58]:
```
pid = test_data['PassengerId']
```

In [60]:
```
test_data.drop(['Cabin','Name','PassengerId'],axis=1,inplace=True)
```

In [64]:
```
# LAbel encoding and removal of NAN values.
test_data .loc[:, 'Sex'] = le.fit_transform(test_data .loc[:,'Sex'])
test_data .loc[:, 'Embarked'] = le.fit_transform(test_data .loc[:, 'Embarked'])
test_data .loc[:, 'Fare'] = le.fit_transform(test_data .loc[:, 'Fare'])
test_data .loc[:, 'Ticket'] = le.fit_transform(test_data .loc[:, 'Ticket'])
test_data ['Age'] = test_data ['Age'].fillna(test_data ['Age'].mean(), inplace = Fa
lse)

test_data
```

Out[64]:

	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	3	1	34.50000	0	0	152	24	1
1	3	0	47.00000	1	0	221	5	2
2	2	1	62.00000	0	0	73	41	1
3	3	1	27.00000	0	0	147	34	2
4	3	0	22.00000	1	1	138	46	2
...	...	...	...	...	...	...	...	...
413	3	1	30.27259	0	0	267	31	2
414	1	0	39.00000	0	0	324	154	0
415	3	1	38.50000	0	0	346	9	2
416	3	1	30.27259	0	0	220	31	2
417	3	1	30.27259	1	1	105	84	0

418 rows × 8 columns

Titanic Test Data Training

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
training_x, testing_x, training_y, testing_y = train_test_split(X_data, Y_data, tes
t_size=0.25, random_state=0)
# Create the model with 100 trees
final_model = RandomForestClassifier(n_estimators=100)
# Fit on training data
final_model.fit(training_x,training_y)

score_rfc = final_model.score(testing_x,testing_y)
```

In [66]:
```
rfc_preds = final_model.predict(test_data)
```

In [68]:
```
# Making Final Report
submission =  pd.DataFrame({'PassengerId':pid,'Survived':rfc_preds})
submission.to_csv('submission_project.csv',index=False)
```

In [70]:
```
model_10
```

Out[70]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [71]:
```
knn = model_10.predict(test_data)
```

In [72]:
```
submission_knn =  pd.DataFrame({'PassengerId':pid,'Survived':knn})
submission_knn.to_csv('submission_knn.csv',index=False)
```

In [73]:
```
model_9
```

Out[73]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [74]:
```
 GaussianNB = model_9.predict(test_data)
```

In [75]:
```
submission_GNB =  pd.DataFrame({'PassengerId':pid,'Survived':GaussianNB})
submission_GNB.to_csv('submission_GaussianNB.csv',index=False)
```

In [76]:
```
model_8
```

Out[76]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [77]:
```
SVC = model_8.predict(test_data)
```

In [78]:
```
submission_SVC =  pd.DataFrame({'PassengerId':pid,'Survived':SVC})
submission_SVC.to_csv('submission_SVC.csv',index=False)
```

In [79]:
```
model_11
```

Out[79]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', 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='deprecated',
                      random_state=0, splitter='best')
```

In [80]:
```
DTR = model_11.predict(test_data)
```

In [81]:
```
submission_DTR =  pd.DataFrame({'PassengerId':pid,'Survived':DTR})
submission_DTR.to_csv('submission_DTR.csv',index=False)
```

Submission_Output

In [82]:
```
submission
```

Out[82]:

	PassengerId	Survived
0	892	1
1	893	1
2	894	1
3	895	1
4	896	1
...	...	...
413	1305	1
414	1306	1
415	1307	1
416	1308	1
417	1309	1

418 rows × 2 columns

In [83]:
```
submission_knn
```

Out[83]:

	PassengerId	Survived
0	892	1
1	893	1
2	894	0
3	895	1
4	896	1
...	...	...
413	1305	1
414	1306	1
415	1307	1
416	1308	1
417	1309	1

418 rows × 2 columns

In [84]:
```
submission_GNB
```

Out[84]:

	PassengerId	Survived
0	892	1
1	893	1
2	894	1
3	895	1
4	896	1
...	...	...
413	1305	1
414	1306	1
415	1307	1
416	1308	1
417	1309	1

418 rows × 2 columns

In [85]:
```
submission_SVC
```

Out[85]:

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	0
...	...	...
413	1305	0
414	1306	0
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

In [86]:
```
submission_DTR
```

Out[86]:

	PassengerId	Survived
0	892	1.0
1	893	1.0
2	894	1.0
3	895	1.0
4	896	1.0
...	...	...
413	1305	1.0
414	1306	1.0
415	1307	1.0
416	1308	1.0
417	1309	1.0

418 rows × 2 columns

Conclusion

Our predicting score is almost in between 80%-83%, which means that we have correctly predicted our target, i.e. the survival rate, in 80%-83%, of cases. During the data preprocessing part, I computed missing values, converted features into numeric ones, modified features and in a last step I made a prediction with various models.

NAME:- SHIVAM SONI

DATASETS USED:- TITANIC DATASET