TASK 2

Perform data cleaning and exploratory data analysis (EDA) on a dataset of your choice, such as the Titanic dataset from Kaggle. Explore the relationships between variables and identify patterns and trends in the data.

In [116]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [117]:
```
pd.set_option('display.max_columns',10,'display.width',1000)
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
train.head()
```

Out[117]:

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

5 rows × 12 columns

In [118]:
```
train.shape
```

Out[118]: (891, 12)

In [119]:
```
test.shape
```

Out[119]:
```
(418, 11)
```

In [120]:
```
#Checking for Null values
train.isnull().sum()
```

Out[120]:
```
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 [121]:
```
test.isnull().sum()
```

Out[121]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [122]:
```
#Description of data set
train.describe(include="all")
```

Out[122]:

	PassengerId	Survived	Pclass	Name	Sex	...	Parch	Ticket	Fare
count	891.000000	891.000000	891.000000	891	891	...	891.000000	891	891.000000
unique	NaN	NaN	NaN	891	2	...	NaN	681	NaN
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	...	NaN	347082	NaN
freq	NaN	NaN	NaN	1	577	...	NaN	7	NaN
mean	446.000000	0.383838	2.308642	NaN	NaN	...	0.381594	NaN	32.204208
std	257.353842	0.486592	0.836071	NaN	NaN	...	0.806057	NaN	49.693429
min	1.000000	0.000000	1.000000	NaN	NaN	...	0.000000	NaN	0.000000
25%	223.500000	0.000000	2.000000	NaN	NaN	...	0.000000	NaN	7.910400
50%	446.000000	0.000000	3.000000	NaN	NaN	...	0.000000	NaN	14.454200
75%	668.500000	1.000000	3.000000	NaN	NaN	...	0.000000	NaN	31.000000
max	891.000000	1.000000	3.000000	NaN	NaN	...	6.000000	NaN	512.329200

11 rows × 12 columns

In [123]:
```
numeric_columns = train.select_dtypes(include=[np.number])
mean_values = numeric_columns.groupby(train['Survived']).mean()
print(mean_values)
```

```
          PassengerId  Survived    Pclass        Age     SibSp     Parch
Fare
Survived
0          447.016393       0.0  2.531876  30.626179  0.553734  0.329690  2
2.117887
1          444.368421       1.0  1.950292  28.343690  0.473684  0.464912  4
8.395408
```

In [124]:
```
correlation_matrix = train.corr()
print(correlation_matrix)
```

```
             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 [125]:
```
male_ind = len(train[train['Sex'] == 'male'])
print("No of Males in Titanic:",male_ind)
```

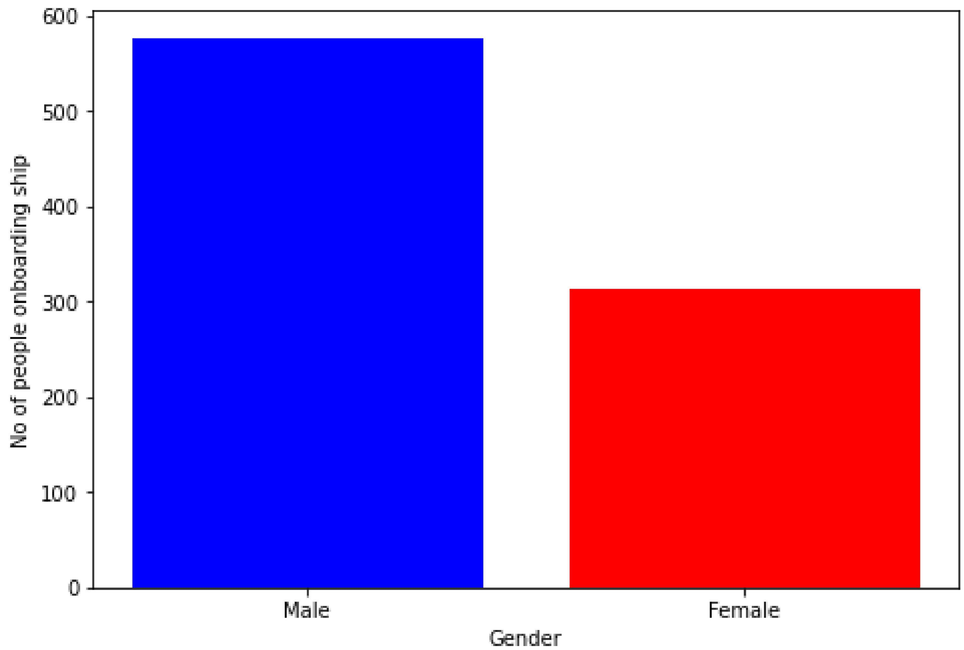
```
No of Males in Titanic: 577
```

In [126]:
```
female_ind = len(train[train['Sex'] == 'female'])
print("No of Females in Titanic:",female_ind)
```

```
No of Females in Titanic: 314
```

In [127]:
```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
gender = ['Male', 'Female']
index = [577, 314]
ax.bar(gender, index, color=['blue', 'red'])
plt.xlabel("Gender")
plt.ylabel("No of people onboarding ship")
plt.show()
```



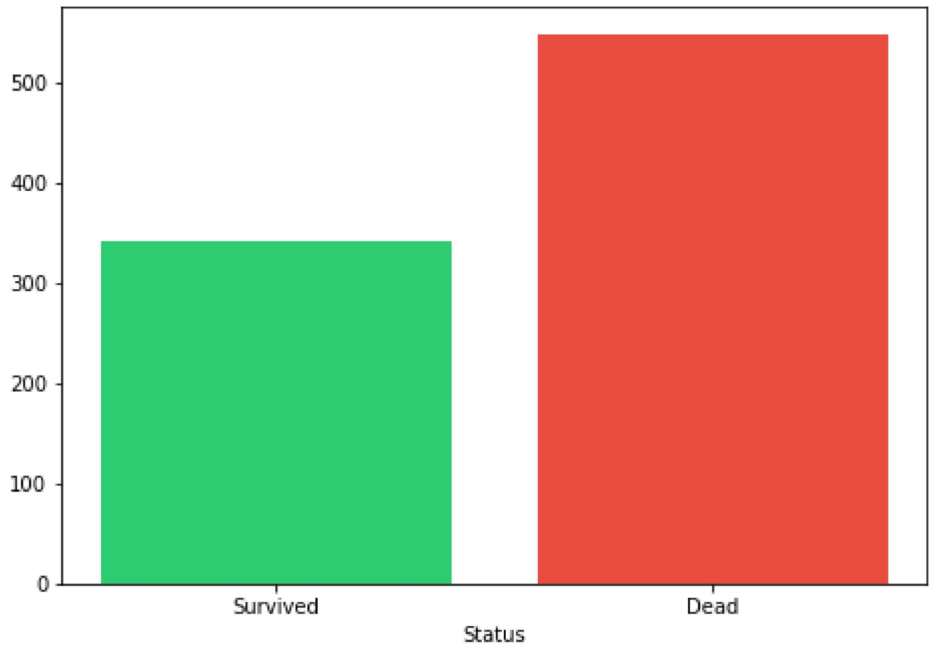
In [128]:
```
alive = len(train[train['Survived'] == 1])
dead = len(train[train['Survived'] == 0])
```

In [129]:
```
train.groupby('Sex')[['Survived']].mean()
```

Out[129]:

	Survived
Sex	
female	0.742038
male	0.188908

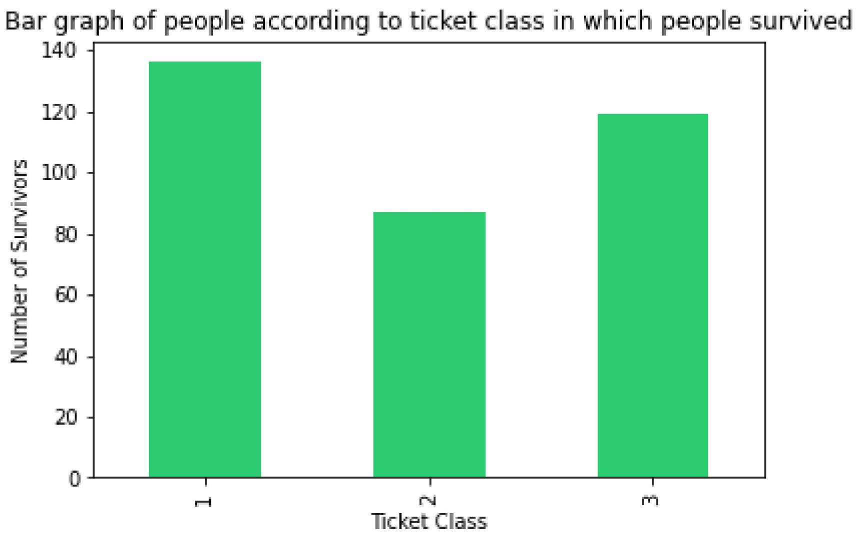
In [130]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
status = ['Survived','Dead']
ind = [alive,dead]
ax.bar(status,ind, color=['#2ecc71', '#e74c3c'])
plt.xlabel("Status")
plt.show()
```

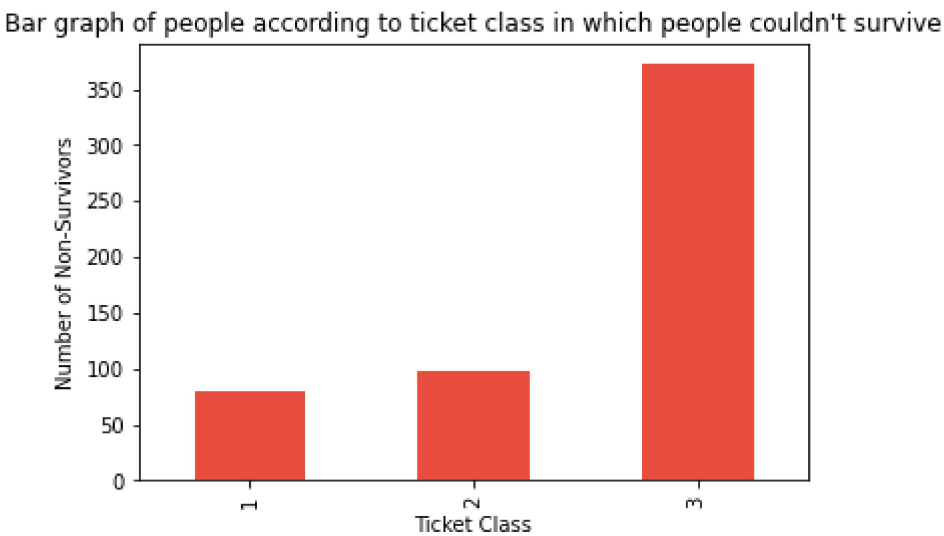


In [131]:
```
plt.figure(1)
train.loc[train['Survived'] == 1, 'Pclass'].value_counts().sort_index().plot.b
plt.title('Bar graph of people according to ticket class in which people survi
plt.xlabel('Ticket Class')
plt.ylabel('Number of Survivors')

plt.figure(2)
train.loc[train['Survived'] == 0, 'Pclass'].value_counts().sort_index().plot.b
plt.title('Bar graph of people according to ticket class in which people could
plt.xlabel('Ticket Class')
plt.ylabel('Number of Non-Survivors')

plt.show()
```

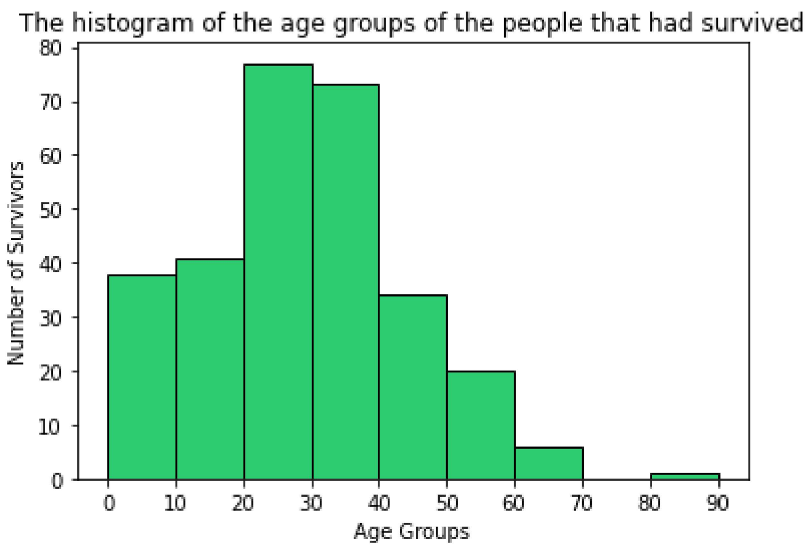


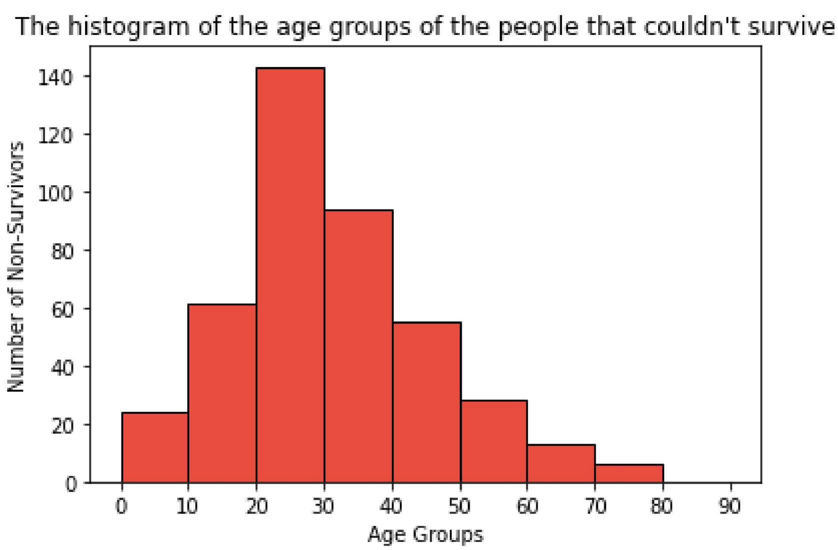


In [132]:
```
plt.figure(1)
age_survived = train.loc[train.Survived == 1, 'Age']
plt.title('The histogram of the age groups of the people that had survived')
plt.hist(age_survived, bins=np.arange(0, 100, 10), color='#2ecc71', edgecolor=
plt.xticks(np.arange(0, 100, 10))
plt.xlabel('Age Groups')
plt.ylabel('Number of Survivors')

plt.figure(2)
age_not_survived = train.loc[train.Survived == 0, 'Age']
plt.title('The histogram of the age groups of the people that couldn\'t surviv
plt.hist(age_not_survived, bins=np.arange(0, 100, 10), color='#e74c3c', edgecc
plt.xticks(np.arange(0, 100, 10))
plt.xlabel('Age Groups')
plt.ylabel('Number of Non-Survivors')

plt.show()
```





In [133]:
```
train[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_va
```

Out[133]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

In [134]:
```
train[["Pclass", "Survived"]].groupby(['Pclass'], as_index=False).mean().sort_
```

Out[134]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

In [135]:
```
train[["Age", "Survived"]].groupby(['Age'], as_index=False).mean().sort_values
```

Out[135]:

	Age	Survived
0	0.42	1.0
1	0.67	1.0
2	0.75	1.0
3	0.83	1.0
4	0.92	1.0
...	...	...
83	70.00	0.0
84	70.50	0.0
85	71.00	0.0
86	74.00	0.0
87	80.00	1.0

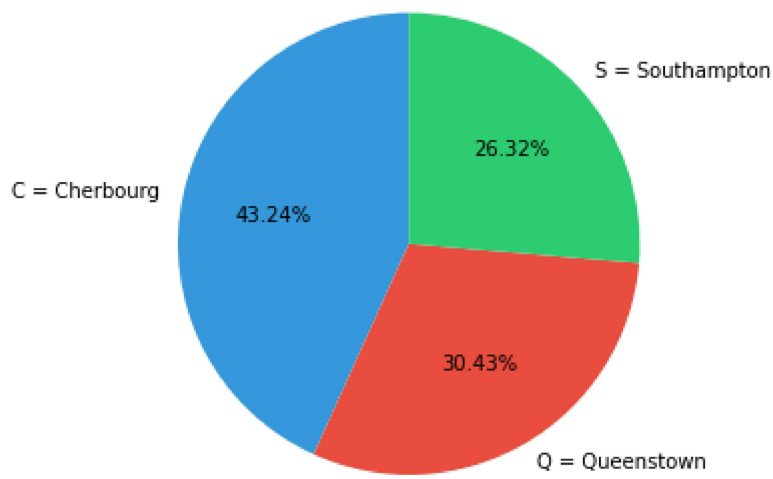
88 rows × 2 columns

In [136]:
```
train[["Embarked", "Survived"]].groupby(['Embarked'], as_index=False).mean().s
```

Out[136]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.336957

In [137]:
```
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
ax.axis('equal')
l = ['C = Cherbourg', 'Q = Queenstown', 'S = Southampton']
s = [0.553571, 0.389610, 0.336957]
ax.pie(s, labels=l, autopct='%1.2f%%', colors=['#3498db', '#e74c3c', '#2ecc71'
plt.show()
```



In [138]:
```
test.describe(include="all")
```

Out[138]:

	PassengerId	Pclass	Name	Sex	Age	...	Parch	Ticket	Fare
count	418.000000	418.000000	418	418	332.000000	...	418.000000	418	417.000000
unique	NaN	NaN	418	2	NaN	...	NaN	363	NaN
top	NaN	NaN	Kelly, Mr. James	male	NaN	...	NaN	PC 17608	NaN
freq	NaN	NaN	1	266	NaN	...	NaN	5	NaN
mean	1100.500000	2.265550	NaN	NaN	30.272590	...	0.392344	NaN	35.627188
std	120.810458	0.841838	NaN	NaN	14.181209	...	0.981429	NaN	55.907576
min	892.000000	1.000000	NaN	NaN	0.170000	...	0.000000	NaN	0.000000
25%	996.250000	1.000000	NaN	NaN	21.000000	...	0.000000	NaN	7.895800
50%	1100.500000	3.000000	NaN	NaN	27.000000	...	0.000000	NaN	14.454200
75%	1204.750000	3.000000	NaN	NaN	39.000000	...	0.000000	NaN	31.500000
max	1309.000000	3.000000	NaN	NaN	76.000000	...	9.000000	NaN	512.329200

11 rows × 11 columns

In [139]:
```
train = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)
```

In [140]:
```
train = train.drop(['Cabin'], axis = 1)
test = test.drop(['Cabin'], axis = 1)
```

In [141]:
```
if 'Name' in train.columns:
    train = train.drop(['Name'], axis=1)

if 'Name' in test.columns:
    test = test.drop(['Name'], axis=1)
```

In [142]:
```
#Feature Selection
column_train=['Age','Pclass','SibSp','Parch','Fare','Sex','Embarked']
#training values
X=train[column_train]
#target value
Y=train['Survived']
```

In [143]:
```
X['Age'].isnull().sum()
X['Pclass'].isnull().sum()
X['SibSp'].isnull().sum()
X['Parch'].isnull().sum()
X['Fare'].isnull().sum()
X['Sex'].isnull().sum()
X['Embarked'].isnull().sum()
```

Out[143]:
```
2
```

In [144]:
```
X['Age']=X['Age'].fillna(X['Age'].median())
X['Age'].isnull().sum()
```

Out[144]:
```
0
```

In [145]:
```
X['Embarked'] = train['Embarked'].fillna(method ='pad')
X['Embarked'].isnull().sum()
```

Out[145]:
```
0
```

In [146]:
```
d={'male':0, 'female':1}
X['Sex']=X['Sex'].apply(lambda x:d[x])
X['Sex'].head()
```

Out[146]:
```
0    0
1    1
2    1
3    1
4    0
Name: Sex, dtype: int64
```

In [147]:
```
e={'C':0, 'Q':1 ,'S':2}
X['Embarked']=X['Embarked'].apply(lambda x:e[x])
X['Embarked'].head()
```

Out[147]:
```
0    2
1    0
2    2
3    2
4    2
Name: Embarked, dtype: int64
```

In [148]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_s
```

In [149]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score: 0.7574626865671642
```

In [150]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

```
[[130  26]
 [ 39  73]]
```

In [151]:
```
from sklearn.svm import SVC
model1 = SVC()
model1.fit(X_train,Y_train)

pred_y = model1.predict(X_test)

from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(Y_test,pred_y))
```

```
Acc= 0.6604477611940298
```

In [152]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
confusion_mat = confusion_matrix(Y_test,pred_y)
print(confusion_mat)
print(classification_report(Y_test,pred_y))
```

```
[[149   7]
 [ 84  28]]
              precision    recall  f1-score   support

           0       0.64      0.96      0.77       156
           1       0.80      0.25      0.38       112

    accuracy                           0.66       268
   macro avg       0.72      0.60      0.57       268
weighted avg       0.71      0.66      0.61       268
```

In [153]:
```
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(X_train,Y_train)
y_pred2 = model2.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred2))
```

```
Accuracy Score: 0.6604477611940298
```

In [154]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
confusion_mat = confusion_matrix(Y_test,y_pred2)
print(confusion_mat)
print(classification_report(Y_test,y_pred2))
```

```
[[127  29]
 [ 62  50]]
              precision    recall  f1-score   support

           0       0.67      0.81      0.74       156
           1       0.63      0.45      0.52       112

    accuracy                           0.66       268
   macro avg       0.65      0.63      0.63       268
weighted avg       0.66      0.66      0.65       268
```

In [155]:
```
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred3 = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred3))
```

```
Accuracy Score: 0.7686567164179104
```

In [156]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
confusion_mat = confusion_matrix(Y_test,y_pred3)
print(confusion_mat)
print(classification_report(Y_test,y_pred3))
```

```
[[129  27]
 [ 35  77]]
              precision    recall  f1-score   support

           0       0.79      0.83      0.81       156
           1       0.74      0.69      0.71       112

    accuracy                           0.77       268
   macro avg       0.76      0.76      0.76       268
weighted avg       0.77      0.77      0.77       268
```

In [157]:
```
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(X_train,Y_train)
y_pred4 = model4.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred4))
```

```
Accuracy Score: 0.7425373134328358
```

In [158]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
confusion_mat = confusion_matrix(Y_test,y_pred4)
print(confusion_mat)
print(classification_report(Y_test,y_pred4))
```

```
[[132  24]
 [ 45  67]]
              precision    recall  f1-score   support

           0       0.75      0.85      0.79       156
           1       0.74      0.60      0.66       112

    accuracy                           0.74       268
   macro avg       0.74      0.72      0.73       268
weighted avg       0.74      0.74      0.74       268
```

In [159]:
```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive Bayes','
    'Score': [0.75,0.66,0.76,0.66,0.74]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Out[159]:

	Model
Score	
0.76	Naive Bayes
0.75	Logistic Regression
0.74	Decision Tree
0.66	Support Vector Machines
0.66	KNN