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

pd.set_option('display.max_columns',10,'display.width',1000)
train = pd.read_csv('/Users/aaryanbabuta/Documents/Prodigy DS
Internship June 2024/Task 2 data Prodigy (titanic)/train.csv')
test = pd.read_csv('/Users/aaryanbabuta/Documents/Prodigy DS
Internship June 2024/Task 2 data Prodigy (titanic)/test.csv')
train.head()

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

[5 rows x 12 columns]

train.shape

(891, 12)

test.shape

(418, 11)

train.isnull().sum().any()

True

train.isnull().sum()

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
```

```
test.isnull().sum().any()
```

```
True
```

```
test.isnull().sum()
```

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

```
train.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
```

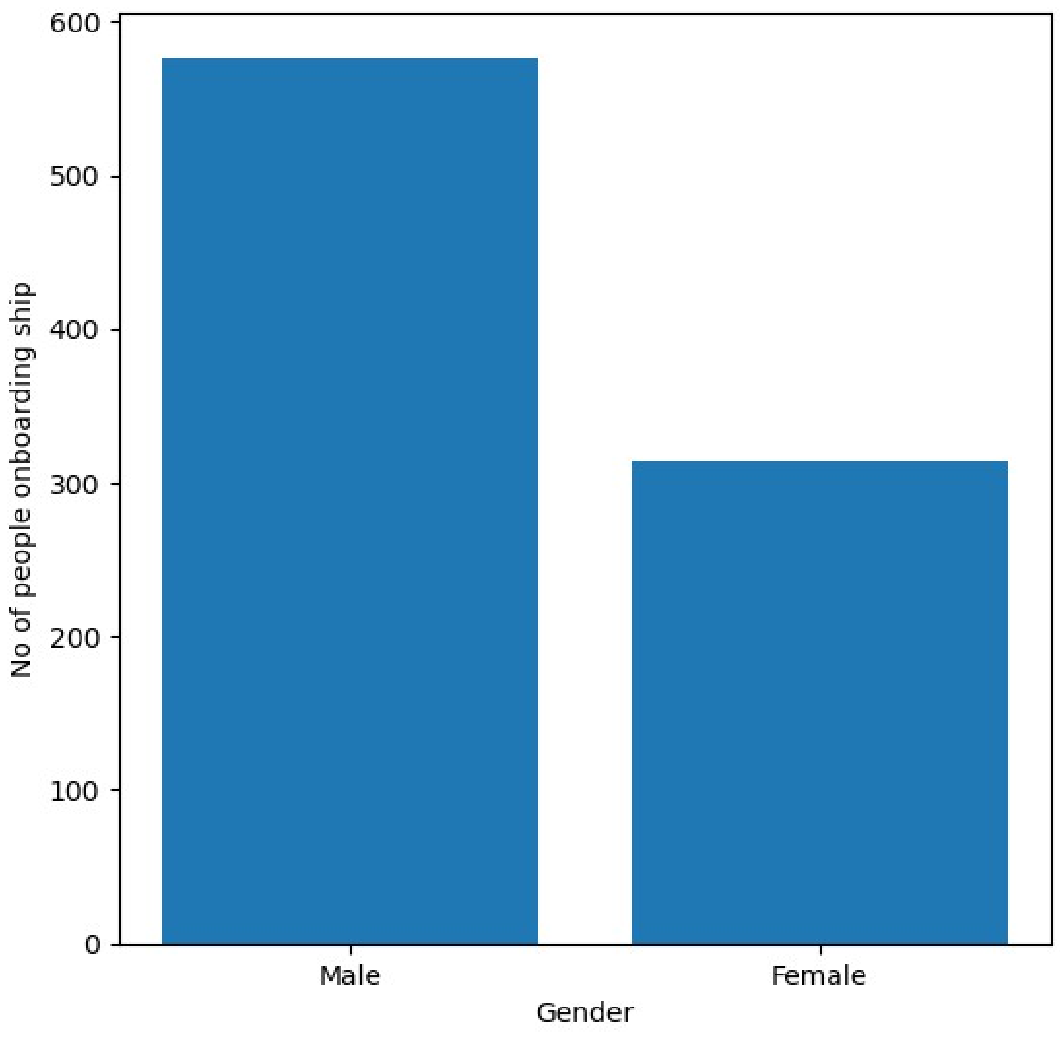
```
total_male = len(train[train['Sex'] == 'male'])
print("Total number of male in Titanic:",total_male)
```

```
Total number of male in Titanic: 577
```

```
total_female = len(train[train['Sex'] == 'female'])
print("Total number of female in Titanic:",total_female)
```

```
Total number of female in Titanic: 314
```

```
fig = plt.figure()
ax = fig.add_axes([0,0,0.75,1])
gender = ['Male','Female']
index = [577,314]
ax.bar(gender,index)
plt.xlabel("Gender")
plt.ylabel("No of people onboarding ship")
plt.show()
```

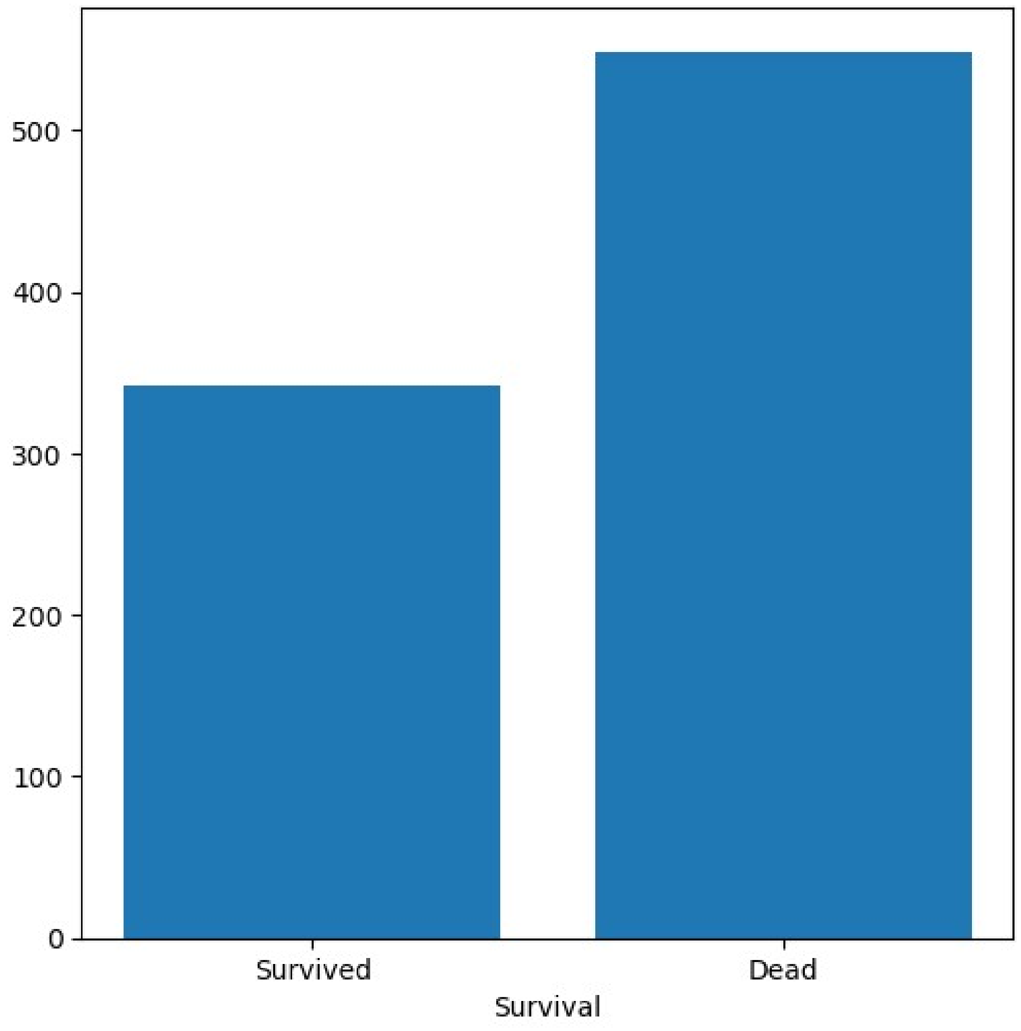


```python
Alive = len(train[train['Survived'] == 1])
Dead = len(train[train['Survived'] == 0])
```

```python
train.groupby('Sex')[['Survived']].mean()
```

```
        Survived
Sex
female  0.742038
male    0.188908
```

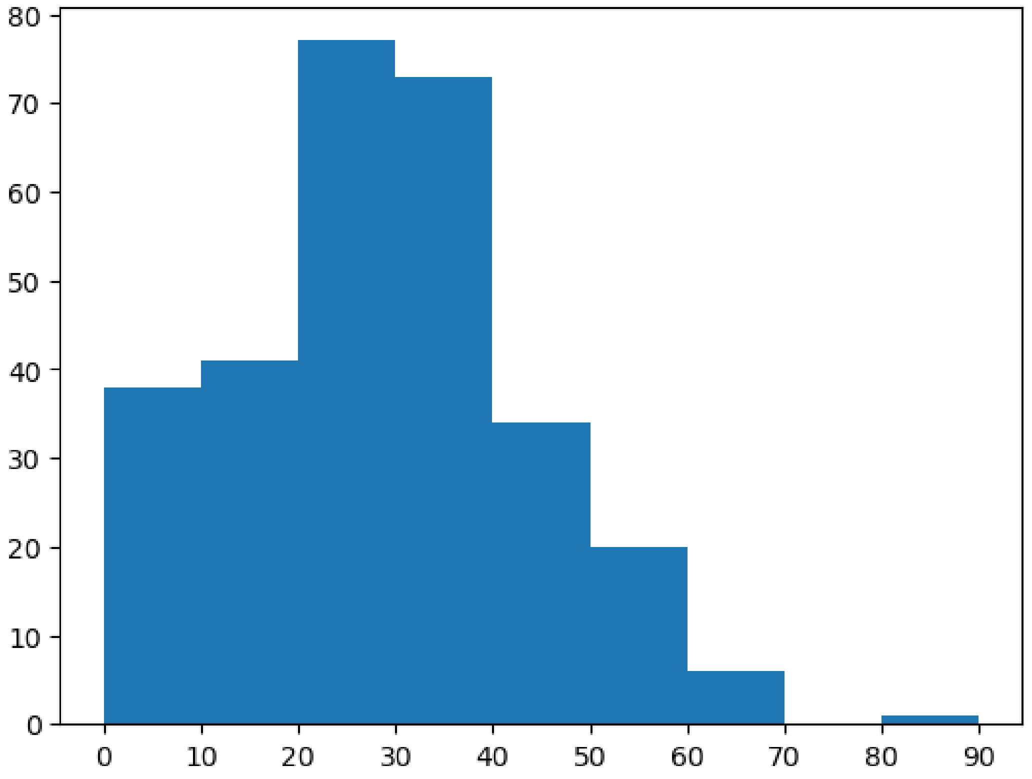
```python
fig = plt.figure()
ax = fig.add_axes([0,0,0.75,1])
survival = ['Survived','Dead'] #status -> survival
index = [Alive,Dead] #ind -> index
ax.bar(survival,index)
plt.xlabel("Survival")
plt.show()
```



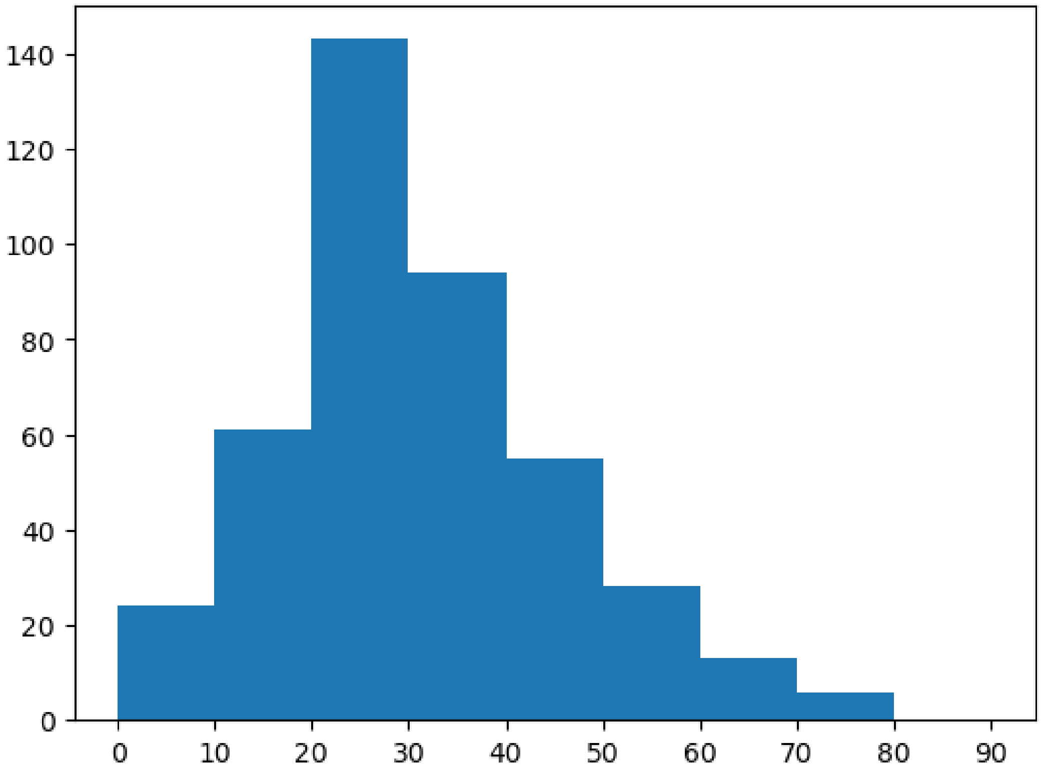
```
plt.figure(1)
age  = train.loc[train.Survived == 1, 'Age']
plt.title('The histogram of the age groups of the people that
survived')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))


plt.figure(2)
age  = train.loc[train.Survived == 0, 'Age']
plt.title('The histogram of the age groups of the people that weren\'t
able to survive')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))
```

```
([<matplotlib.axis.XTick at 0x147481510>,
  <matplotlib.axis.XTick at 0x1474735d0>,
  <matplotlib.axis.XTick at 0x1474613d0>,
  <matplotlib.axis.XTick at 0x1474b6f90>,
  <matplotlib.axis.XTick at 0x1474c1290>,
  <matplotlib.axis.XTick at 0x1474c3250>,
  <matplotlib.axis.XTick at 0x1474c2010>,
  <matplotlib.axis.XTick at 0x1474ca210>,
  <matplotlib.axis.XTick at 0x1474cba50>,
  <matplotlib.axis.XTick at 0x1474d23d0>],
 [Text(0, 0, '0'),
  Text(10, 0, '10'),
  Text(20, 0, '20'),
  Text(30, 0, '30'),
  Text(40, 0, '40'),
  Text(50, 0, '50'),
  Text(60, 0, '60'),
  Text(70, 0, '70'),
  Text(80, 0, '80'),
  Text(90, 0, '90')])
```

The histogram of the age groups of the people that survived

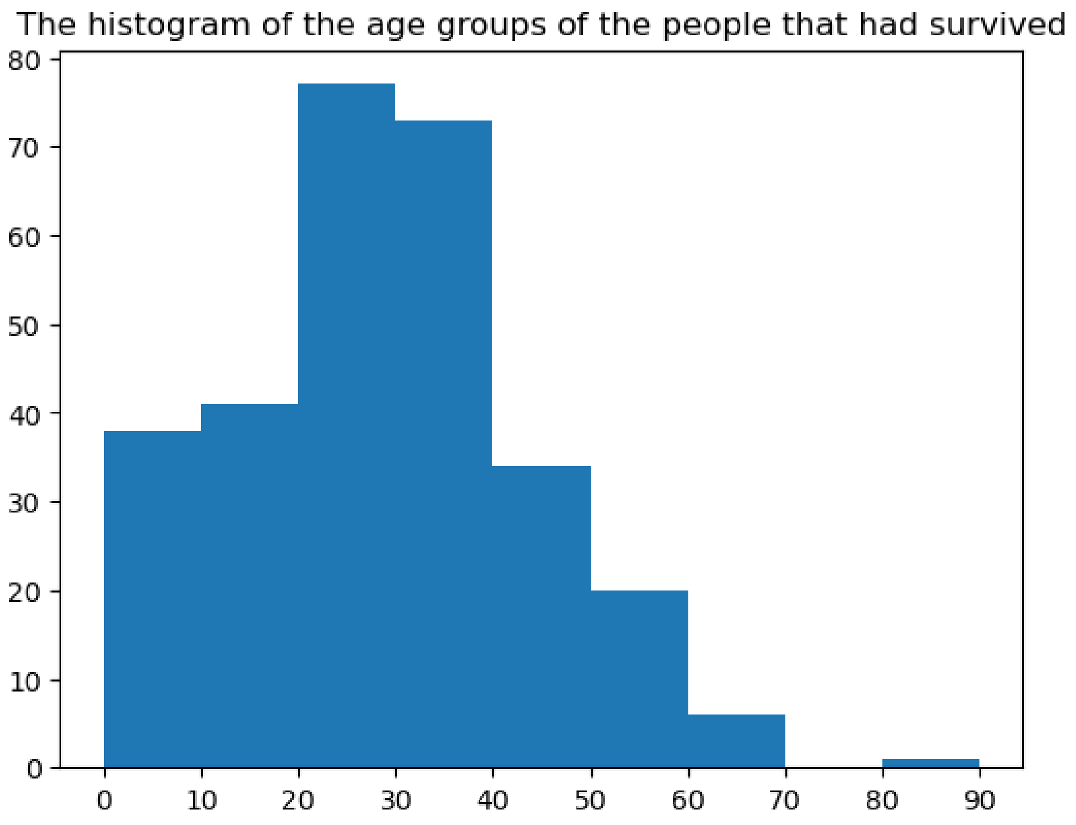


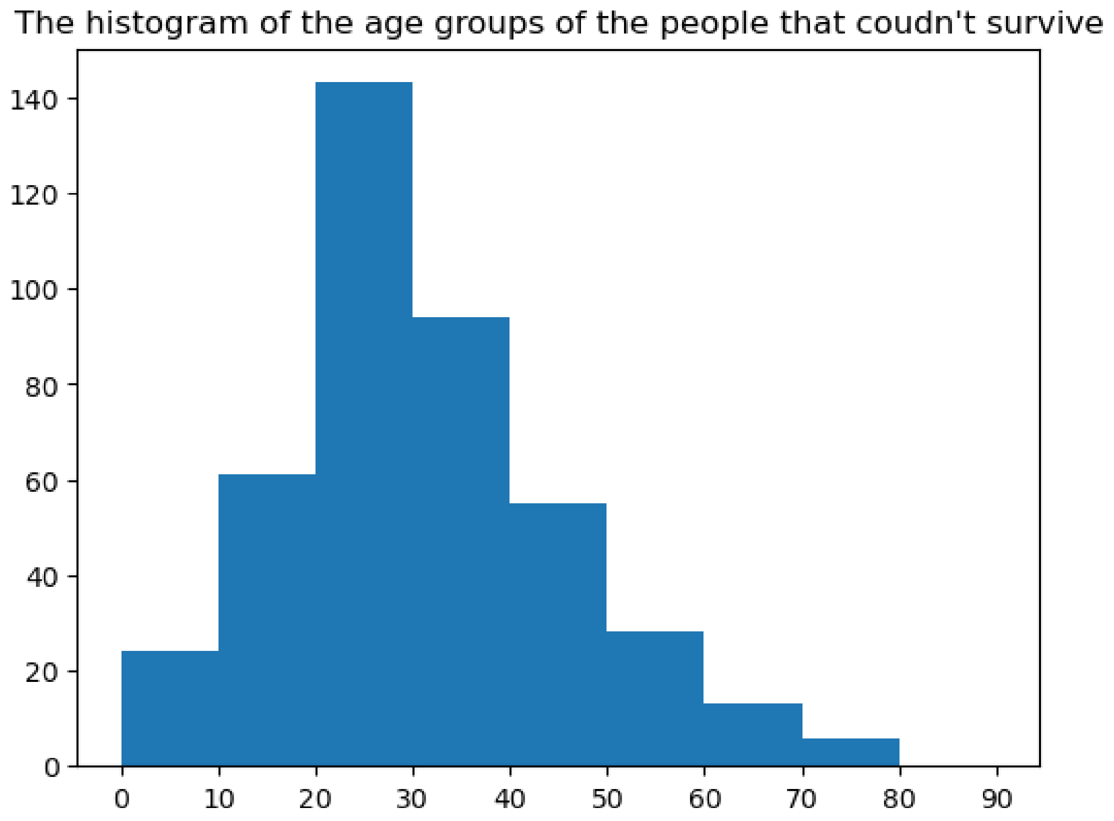
The histogram of the age groups of the people that weren't able to survive



```
plt.figure(1)
age  = train.loc[train.Survived == 1, 'Age']
plt.title('The histogram of the age groups of the people that had
survived')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))


plt.figure(2)
age  = train.loc[train.Survived == 0, 'Age']
plt.title('The histogram of the age groups of the people that coudn\'t
survive')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))
```

```
([<matplotlib.axis.XTick at 0x14757cd90>,
  <matplotlib.axis.XTick at 0x14759ed90>,
  <matplotlib.axis.XTick at 0x1473dd650>,
  <matplotlib.axis.XTick at 0x1475d58d0>,
  <matplotlib.axis.XTick at 0x1475d7990>,
  <matplotlib.axis.XTick at 0x1475d9bd0>,
  <matplotlib.axis.XTick at 0x1475d5f50>,
  <matplotlib.axis.XTick at 0x1475dccd0>,
  <matplotlib.axis.XTick at 0x1475dec10>,
  <matplotlib.axis.XTick at 0x1475e4dd0>],
 [Text(0, 0, '0'),
  Text(10, 0, '10'),
  Text(20, 0, '20'),
  Text(30, 0, '30'),
  Text(40, 0, '40'),
  Text(50, 0, '50'),
  Text(60, 0, '60'),
  Text(70, 0, '70'),
  Text(80, 0, '80'),
  Text(90, 0, '90')])
```

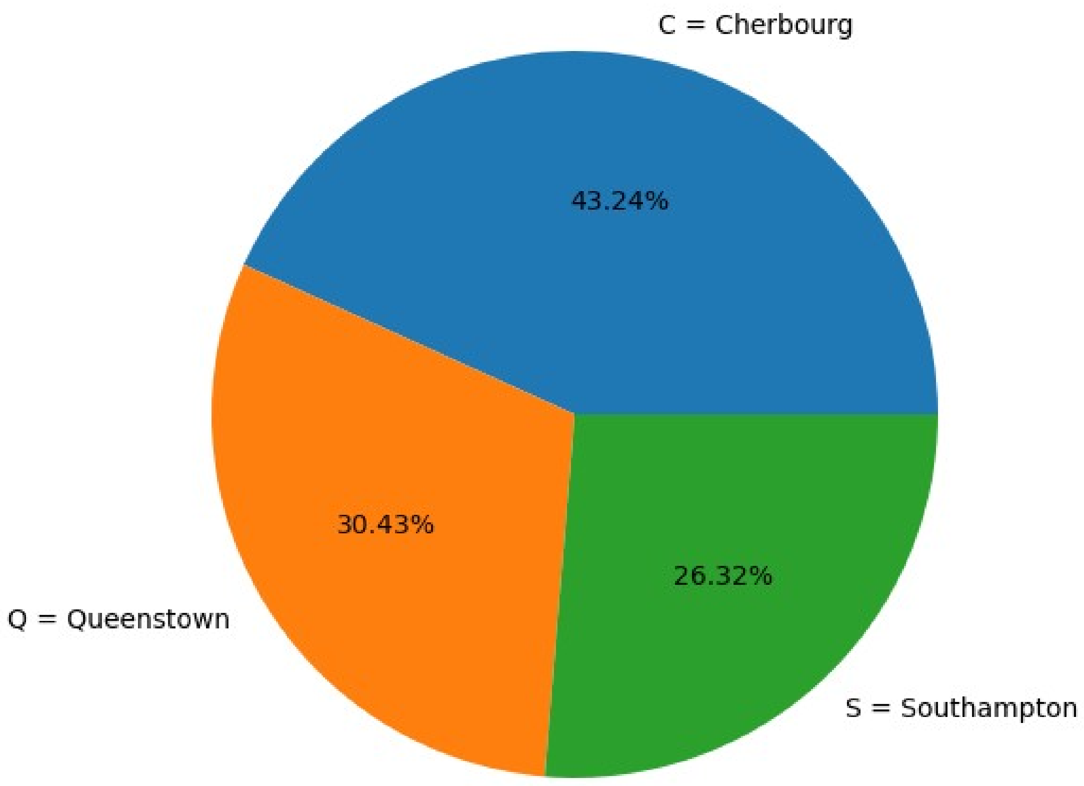

The histogram of the age groups of the people that had survived
0
10
20
30
40
50
60
70
80
90


The histogram of the age groups of the people that coudn't survive
0
20
40
60
80
100
120
140
10
30
50
70
90

```
train[["Embarked", "Survived"]].groupby(['Embarked'],
as_index=False).mean().sort_values(by='Survived', ascending=False)
```

```
  Embarked  Survived
0        C  0.553571
1        Q  0.389610
2        S  0.336957
```

```
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%%')
plt.show()
```



```
rain = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)

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

```
train = train.drop(['Name'], axis = 1)
test = test.drop(['Name'], axis = 1)

column_train=['Age','Pclass','SibSp','Parch','Fare','Sex','Embarked']

X=train[column_train]

Y=train['Survived']

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()

2

X['Age']=X['Age'].fillna(X['Age'].median())
X['Age'].isnull().sum()

0

X['Embarked'] = train['Embarked'].fillna(method ='pad')
X['Embarked'].isnull().sum()

0

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

0    0
1    1
2    1
3    1
4    0
Name: Sex, dtype: int64

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

0    2
1    0
2    2
3    2
4    2
Name: Embarked, dtype: int64
```

```
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_state=7)

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

from sklearn.metrics import accuracy_score,confusion_matrix
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)

[[130  26]
 [ 39  73]]

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

from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
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


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: 0.6567164179104478
```

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
confusion_mat = confusion_matrix(Y_test,y_pred2)
print(confusion_mat)
print(classification_report(Y_test,y_pred2))
```

```
[[126  30]
 [ 62  50]]
              precision    recall  f1-score   support

           0       0.67      0.81      0.73       156
           1       0.62      0.45      0.52       112

    accuracy                           0.66       268
   macro avg       0.65      0.63      0.63       268
weighted avg       0.65      0.66      0.64       268
```

```
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
```

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
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
```

```python
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
```