Assignment 12 - Mini Project

Title - Build a machine learning model that predicts the type of people who survived the Titanic shipwreck using passenger data (i.e. name, age, gender, socio-economic class, etc.).

```
# Name - Vedant Kulkarni
# Roll Number - 51
```

- Importing training data and verifying it

```
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('train.csv')
test = pd.read_csv('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()
```

```
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()
```

```
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.describe(include="all")
```

```
        PassengerId    Survived      Pclass                         Name
Sex  ...       Parch  Ticket        Fare    Cabin  Embarked
count    891.000000  891.000000  891.000000                          891
891  ...  891.000000     891  891.000000      204       889
unique          NaN         NaN         NaN                          891
2  ...         NaN     681         NaN      147         3
top             NaN         NaN         NaN  Braund, Mr. Owen Harris
male  ...         NaN  347082         NaN  B96 B98         S
freq            NaN         NaN         NaN                            1
577  ...         NaN       7         NaN        4       644
mean     446.000000    0.383838    2.308642                          NaN
NaN  ...    0.381594     NaN   32.204208      NaN       NaN
std      257.353842    0.486592    0.836071                          NaN
NaN  ...    0.806057     NaN   49.693429      NaN       NaN
min        1.000000    0.000000    1.000000                          NaN
NaN  ...    0.000000     NaN    0.000000      NaN       NaN
25%      223.500000    0.000000    2.000000                          NaN
```

```
NaN  ...    0.000000     NaN    7.910400      NaN        NaN
50%      446.000000    0.000000    3.000000                    NaN
NaN  ...    0.000000     NaN   14.454200      NaN        NaN
75%      668.500000    1.000000    3.000000                    NaN
NaN  ...    0.000000     NaN   31.000000      NaN        NaN
max      891.000000    1.000000    3.000000                    NaN
NaN  ...    6.000000     NaN  512.329200      NaN        NaN

[11 rows x 12 columns]
```

- Dropping Useless Columns

```
train = 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)

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

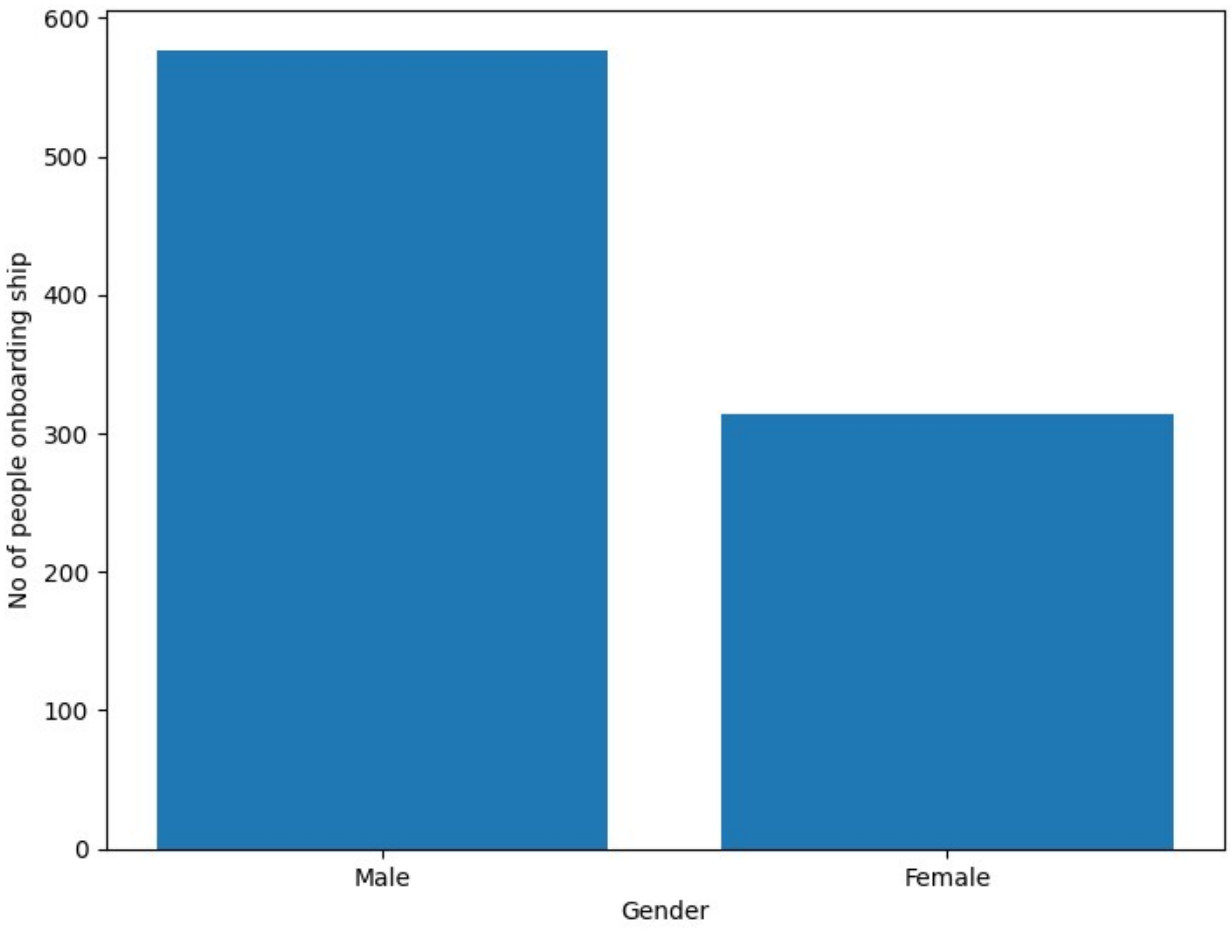
```
No of Males in Titanic: 577
```

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

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

- Training data plots

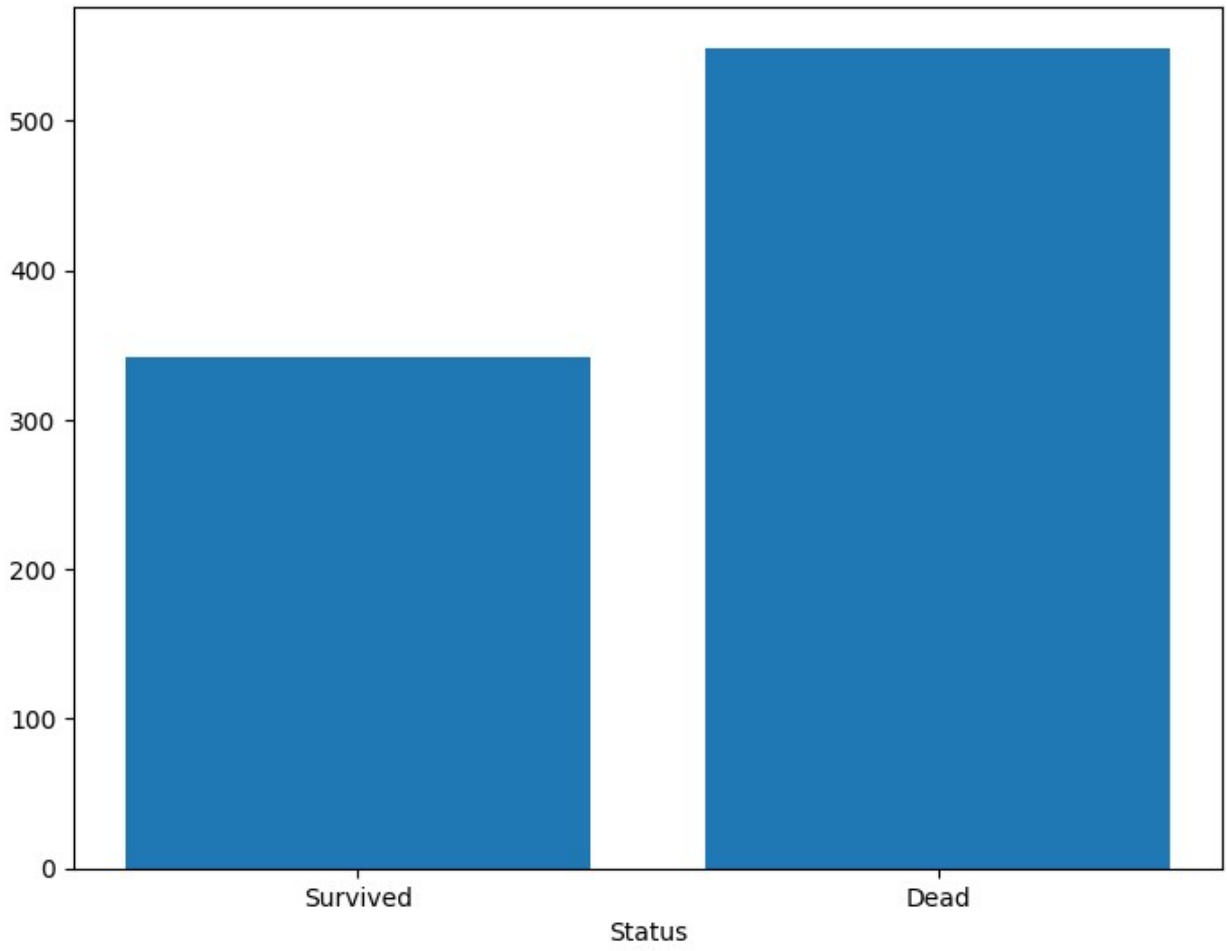
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
gender = ['Male','Female']
index = [577,314]
ax.bar(gender,index)
plt.xlabel("Gender")
plt.ylabel("No of people onboarding ship")
plt.show()
```




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

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

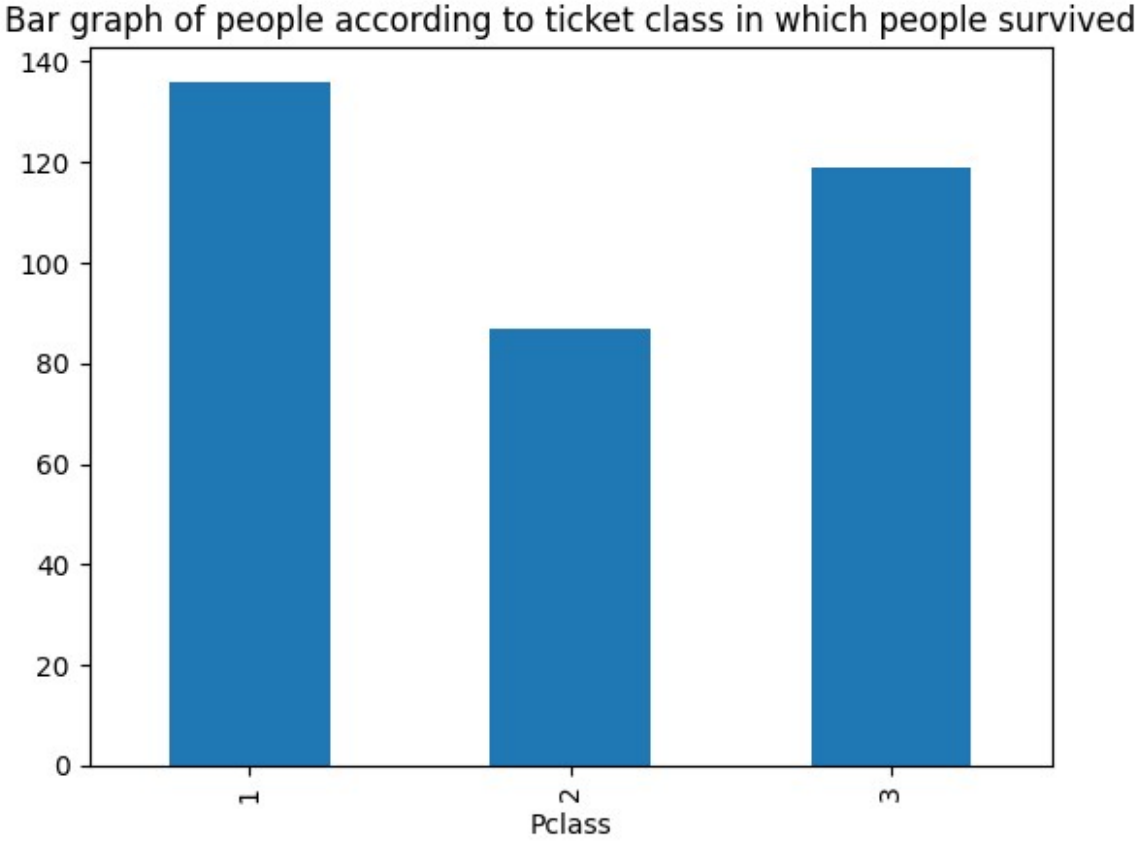
```
        Survived
Sex
female  0.742038
male    0.188908
```

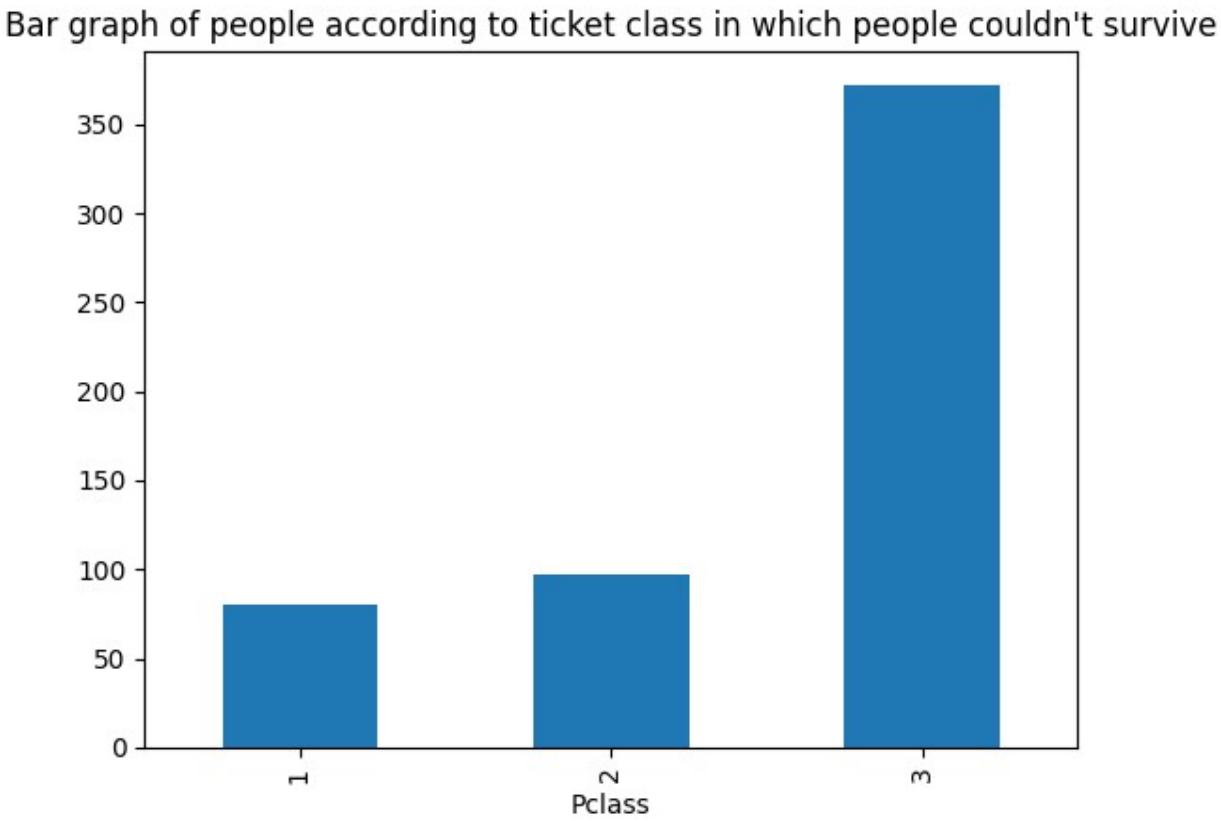
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
status = ['Survived','Dead']
ind = [alive,dead]
ax.bar(status,ind)
plt.xlabel("Status")
plt.show()
```




```
plt.figure(1)
train.loc[train['Survived'] == 1,
'Pclass'].value_counts().sort_index().plot.bar()
plt.title('Bar graph of people according to ticket class in which
people survived')


plt.figure(2)
train.loc[train['Survived'] == 0,
'Pclass'].value_counts().sort_index().plot.bar()
plt.title('Bar graph of people according to ticket class in which
people couldn\'t survive')
```

```
Text(0.5, 1.0, "Bar graph of people according to ticket class in which
people couldn't survive")
```

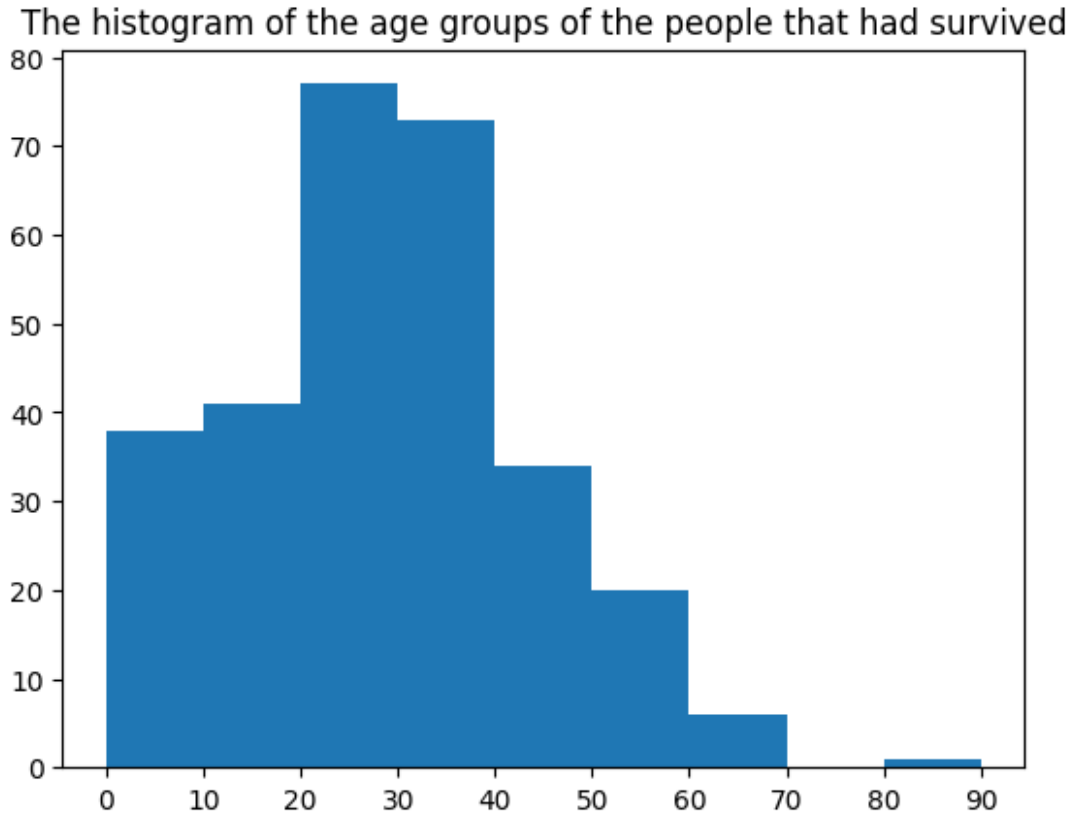

Bar graph of people according to ticket class in which people survived
140
120
100
80
60
40
20
0
1
2
3
Pclass

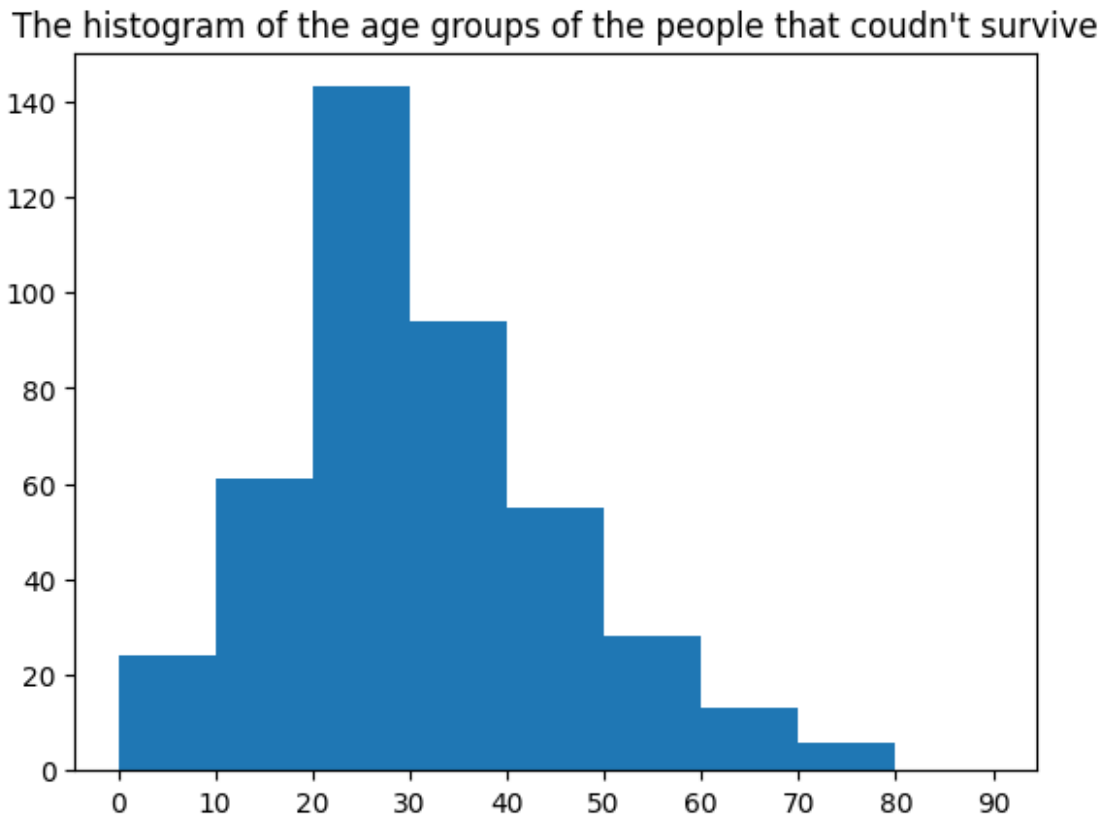



```
plt.figure(1)
age  = train.loc[train.Survived == 1, 'Age']
plt.title('The histogram of the age groups of the people that had
survived')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))


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 0x14f2b834e00>,
  <matplotlib.axis.XTick at 0x14f2df14440>,
  <matplotlib.axis.XTick at 0x14f2b9811f0>,
  <matplotlib.axis.XTick at 0x14f2b97a480>,
  <matplotlib.axis.XTick at 0x14f2df358e0>,
  <matplotlib.axis.XTick at 0x14f2df36240>,
  <matplotlib.axis.XTick at 0x14f2df34ce0>,
  <matplotlib.axis.XTick at 0x14f2df36cf0>,
  <matplotlib.axis.XTick at 0x14f2df37650>,
```

```
 <matplotlib.axis.XTick at 0x14f2df37ef0>],
[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')])
```






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

```
   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
```

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

```
   Pclass  Survived
0       1  0.629630
1       2  0.472826
2       3  0.242363
```

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

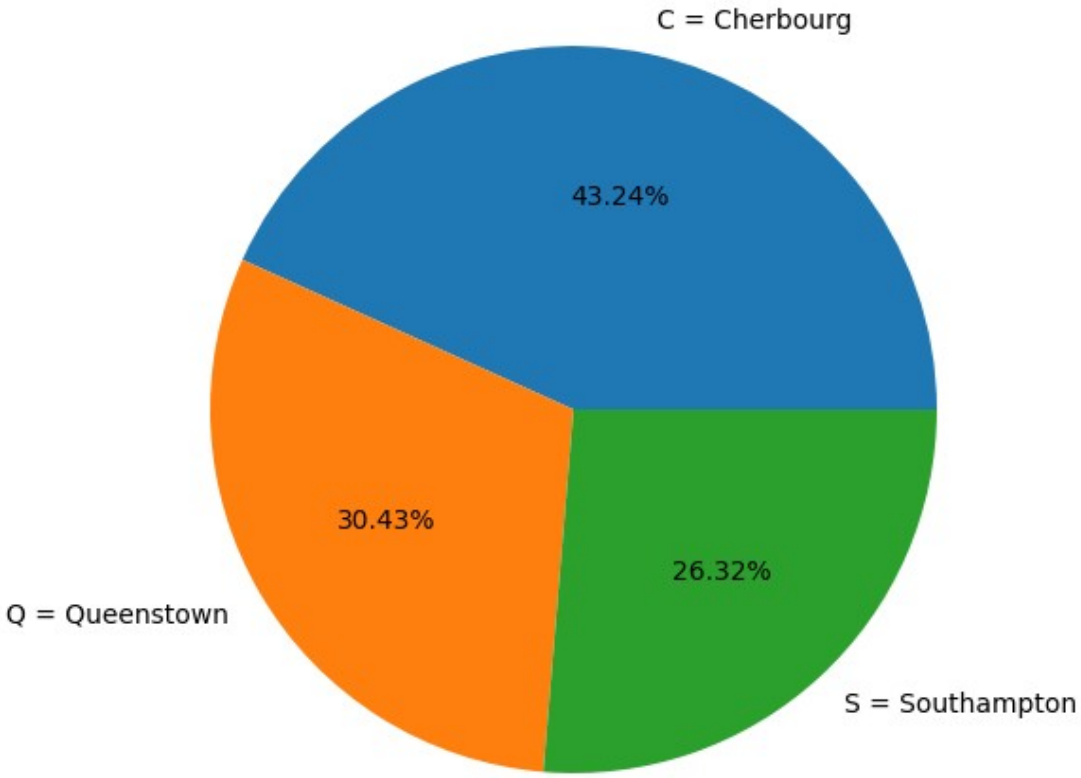
```
      Age  Survived
0    0.42       1.0
```

```
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 x 2 columns]
```

```
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()
```




```
test.describe(include="all")
```

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
count	418.000000	418.000000	418	332.000000	418.000000	418.000000	417.000000	418
unique	NaN	NaN	2	NaN	NaN	NaN	NaN	3
top	NaN	NaN	male	NaN	NaN	NaN	NaN	S
freq	NaN	NaN	266	NaN	NaN	NaN	NaN	270
mean	1100.500000	2.265550	NaN	30.272590	0.447368	0.392344	35.627188	NaN
std	120.810458	0.841838	NaN	14.181209	0.896760	0.981429	55.907576	NaN
min	892.000000	1.000000	NaN	0.170000	0.000000	0.000000	0.000000	NaN
25%	996.250000	1.000000	NaN	21.000000	0.000000	0.000000	7.895800	NaN
50%	1100.500000	3.000000	NaN	27.000000	0.000000			

```
0.000000   14.454200      NaN
75%     1204.750000    3.000000   NaN   39.000000    1.000000
0.000000   31.500000      NaN
max     1309.000000    3.000000   NaN   76.000000    8.000000
9.000000  512.329200      NaN
```

- Dealing with missing values and encoding categorical data in training dataset

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

0

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

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

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

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

0

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

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

- Encoded training data

```
train
```

```
     PassengerId  Survived  Pclass  Sex   Age  SibSp  Parch     Fare
Embarked
0              1         0       3    0  22.0      1      0   7.2500
2
1              2         1       1    1  38.0      1      0  71.2833
0
2              3         1       3    1  26.0      0      0   7.9250
2
3              4         1       1    1  35.0      1      0  53.1000
2
4              5         0       3    0  35.0      0      0   8.0500
2
..           ...       ...     ...  ...   ...    ...    ...      ...
...
886          887         0       2    0  27.0      0      0  13.0000
2
887          888         1       1    1  19.0      0      0  30.0000
2
888          889         0       3    1  28.0      1      2  23.4500
2
889          890         1       1    0  26.0      0      0  30.0000
0
890          891         0       3    0  32.0      0      0   7.7500
1

[891 rows x 9 columns]
```

- Training the model for prediction

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

X=train[column_train]

y=train["Survived"]
Y = pd.DataFrame(y)

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()

0

Y.groupby('Survived')

<pandas.core.groupby.generic.DataFrameGroupBy object at
0x0000014F2E5A3950>
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

train.corr()
```

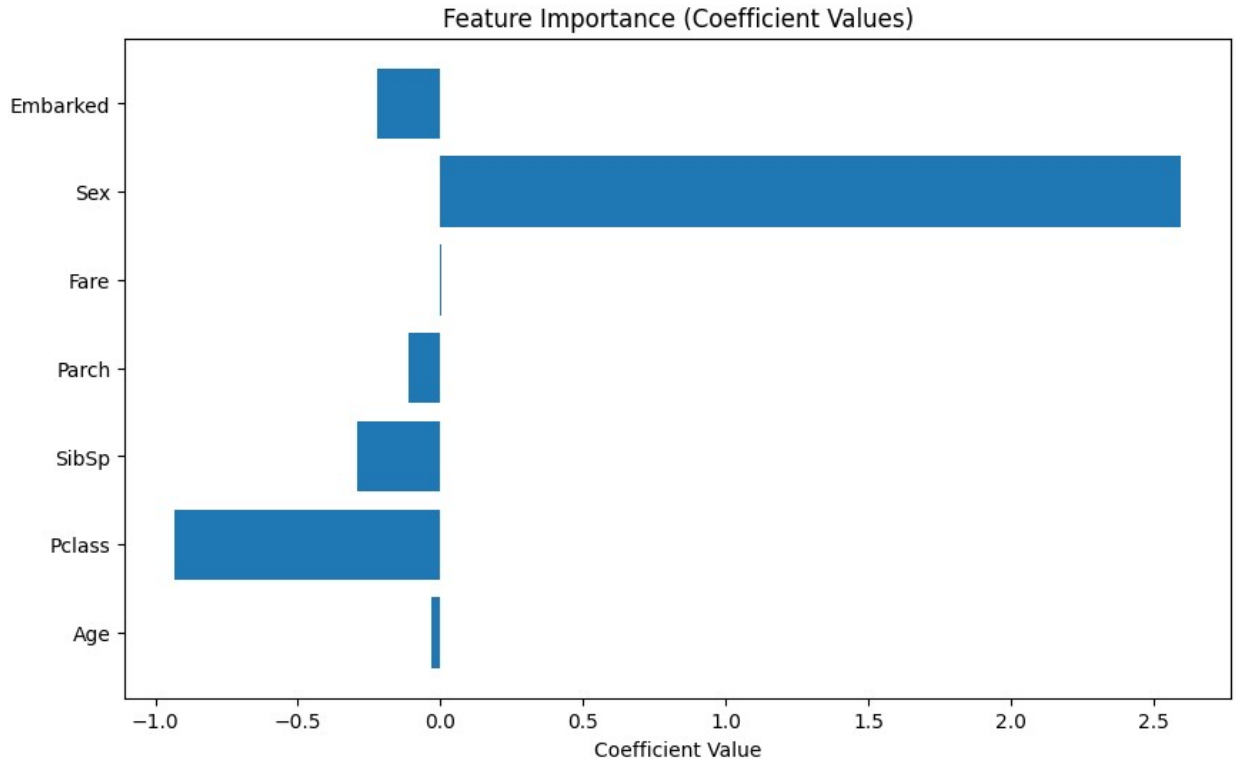
```
             PassengerId  Survived    Pclass       Sex       Age
SibSp     Parch      Fare  Embarked
PassengerId     1.000000 -0.005007 -0.035144 -0.042939  0.034212 -
0.057527 -0.001652  0.012658  0.015216
Survived       -0.005007  1.000000 -0.338481  0.543351 -0.064910 -
0.035322  0.081629  0.257307 -0.172726
Pclass         -0.035144 -0.338481  1.000000 -0.131900 -0.339898
0.083081  0.018443 -0.549500  0.168430
Sex            -0.042939  0.543351 -0.131900  1.000000 -0.081163
0.114631  0.245489  0.182333 -0.113807
Age             0.034212 -0.064910 -0.339898 -0.081163  1.000000 -
0.233296 -0.172482  0.096688 -0.024149
SibSp          -0.057527 -0.035322  0.083081  0.114631 -0.233296
1.000000  0.414838  0.159651  0.070111
Parch          -0.001652  0.081629  0.018443  0.245489 -0.172482
0.414838  1.000000  0.216225  0.041732
Fare            0.012658  0.257307 -0.549500  0.182333  0.096688
0.159651  0.216225  1.000000 -0.228364
Embarked        0.015216 -0.172726  0.168430 -0.113807 -0.024149
0.070111  0.041732 -0.228364  1.000000
```

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,rando
m_state=42)

model=LogisticRegression()

model.fit(x_train,y_train)

LogisticRegression()

prediction=model.predict(x_test)

prediction
```

```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1,
0,
```

```
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
0,
       0, 1, 1], dtype=int64)
```

```
coefficients = model.coef_.ravel()
features = X.columns

plt.figure(figsize=(10,6))
plt.barh(features, coefficients)
plt.xlabel('Coefficient Value')
plt.title('Feature Importance (Coefficient Values)')
plt.show()
```

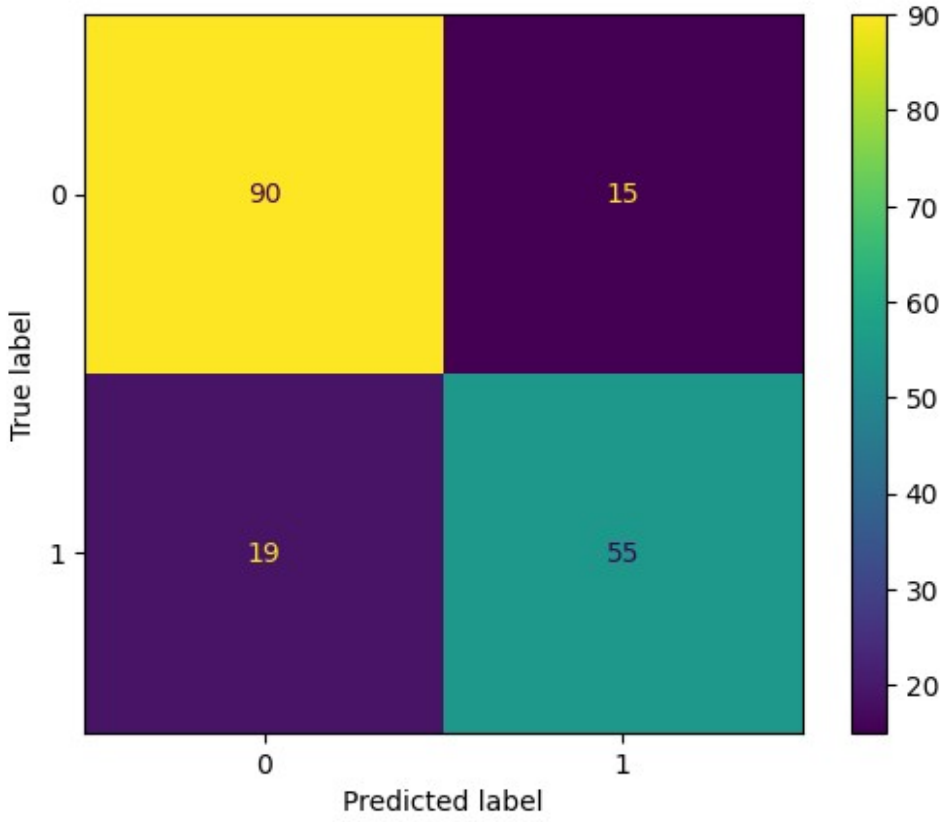



```
from sklearn.metrics import
accuracy_score,confusion_matrix,ConfusionMatrixDisplay

accuracy_score(y_test,prediction)
```

```
0.8100558659217877
```

```
cm = confusion_matrix(y_test,prediction)
display = ConfusionMatrixDisplay(cm)
display.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x14f3f2a2270>
```



- Importing testing dataset

```
test.head

<bound method NDFrame.head of      PassengerId  Pclass     Sex   Age
SibSp  Parch      Fare Embarked
0            892       3    male  34.5      0      0    7.8292
Q
1            893       3  female  47.0      1      0    7.0000
S
2            894       2    male  62.0      0      0    9.6875
Q
3            895       3    male  27.0      0      0    8.6625
S
4            896       3  female  22.0      1      1   12.2875
S
..           ...     ...     ...   ...    ...    ...       ...      ..
.
413         1305       3    male   NaN      0      0    8.0500
S
414         1306       1  female  39.0      0      0  108.9000
```

```
C
415          1307       3    male  38.5      0      0    7.2500
S
416          1308       3    male   NaN      0      0    8.0500
S
417          1309       3    male   NaN      1      1   22.3583
C

[418 rows x 8 columns]>
```

- Selecting the relevant columns

```
clean_test = test[['PassengerId', 'Age', 'Pclass', 'SibSp', 'Parch',
'Fare', 'Sex', 'Embarked']]
clean_test.head()
```

	PassengerId	Age	Pclass	SibSp	Parch	Fare	Sex	Embarked
0	892	34.5	3	0	0	7.8292	male	Q
1	893	47.0	3	1	0	7.0000	female	S
2	894	62.0	2	0	0	9.6875	male	Q
3	895	27.0	3	0	0	8.6625	male	S
4	896	22.0	3	1	1	12.2875	female	S

- Encoding values

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

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

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

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

clean_test.shape

(418, 8)
```

```
clean_test.drop(columns=['PassengerId'], axis=1, inplace=True)
clean_test.head()
```

	Age	Pclass	SibSp	Parch	Fare	Sex	Embarked
0	34.5	3	0	0	7.8292	0	1
1	47.0	3	1	0	7.0000	1	2
2	62.0	2	0	0	9.6875	0	1
3	27.0	3	0	0	8.6625	0	2
4	22.0	3	1	1	12.2875	1	2

- Dealing with missing values in testing data

```
clean_test.isnull().any()
```

```
Age           True
Pclass       False
SibSp        False
Parch        False
Fare          True
Sex          False
Embarked     False
dtype: bool
```

```
clean_test.Fare = clean_test.Fare.fillna(train['Fare'].mean())
```

```
clean_test.Age = clean_test.Age.fillna(train['Age'].mean())
```

```
clean_test.isnull().any()
```

```
Age          False
Pclass       False
SibSp        False
Parch        False
Fare         False
Sex          False
Embarked     False
dtype: bool
```

- Predicting output

```
final_prediction = model.predict(clean_test)
```

```
final_prediction
```

```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
1,
```

```
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
0,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
0],
      dtype=int64)
```