titanic-eda

April 20, 2024

[2]:
```python
#Importing All Required Libaries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from warnings import filterwarnings
filterwarnings(action='ignore')
```

[3]:
```python
#Loading Datasets
pd.set_option('display.max_columns',10,'display.width',1000)
train = pd.read_csv('C:\\Users\\EL HASSANI SAFAA\\Desktop\\Task2_Dataset\\train.
↪csv')
test = pd.read_csv('C:\\Users\\EL HASSANI SAFAA\\Desktop\\Task2_Dataset\\test.
↪csv')
train.head()
```

[3]:
```
   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]
```

[4]:
```python
#Display shape
train.shape
```

[4]: (891, 12)

[5]:
```python
test.shape
```

[5]: (418, 11)

[6]:
```
#Checking for Null values
train.isnull().sum()
```

[6]:
```
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
```

[7]:
```
test.isnull().sum()
```

[7]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

[8]:
```
#Description of dataset
train.describe(include="all")
```

[8]:
```
        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]
```

[9]:
```
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
```

[10]:
```
# Check unique values in the "Survived" column
print(train['Survived'].unique())

# Check for missing values in the "Survived" column
print(train['Survived'].isnull().sum())
```

```
[0 1]
0
```

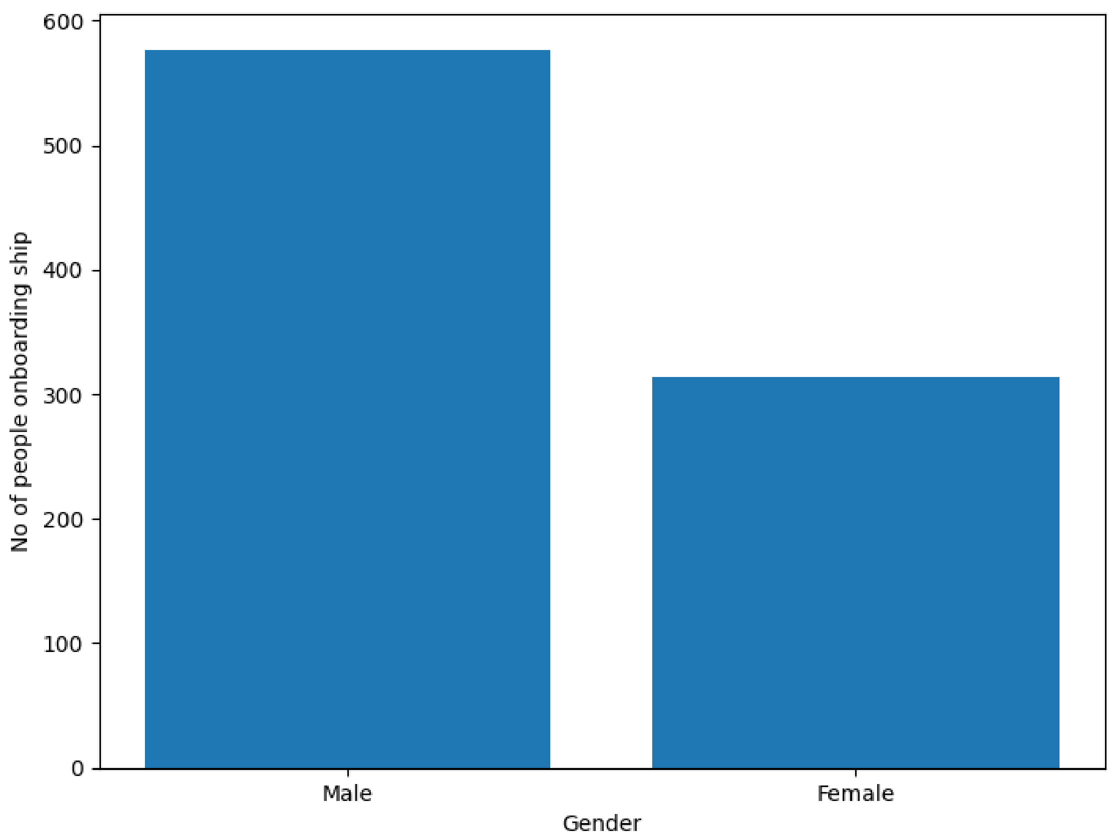
[11]:
```
male_ind = len(train[train['Sex'] == 'male'])
print("No of Males in Titanic:",male_ind)
```

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

[12]:
```
female_ind = len(train[train['Sex'] == 'female'])
print("No of Females in Titanic:",female_ind)
```

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

[13]:
```
#Plotting
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()
```



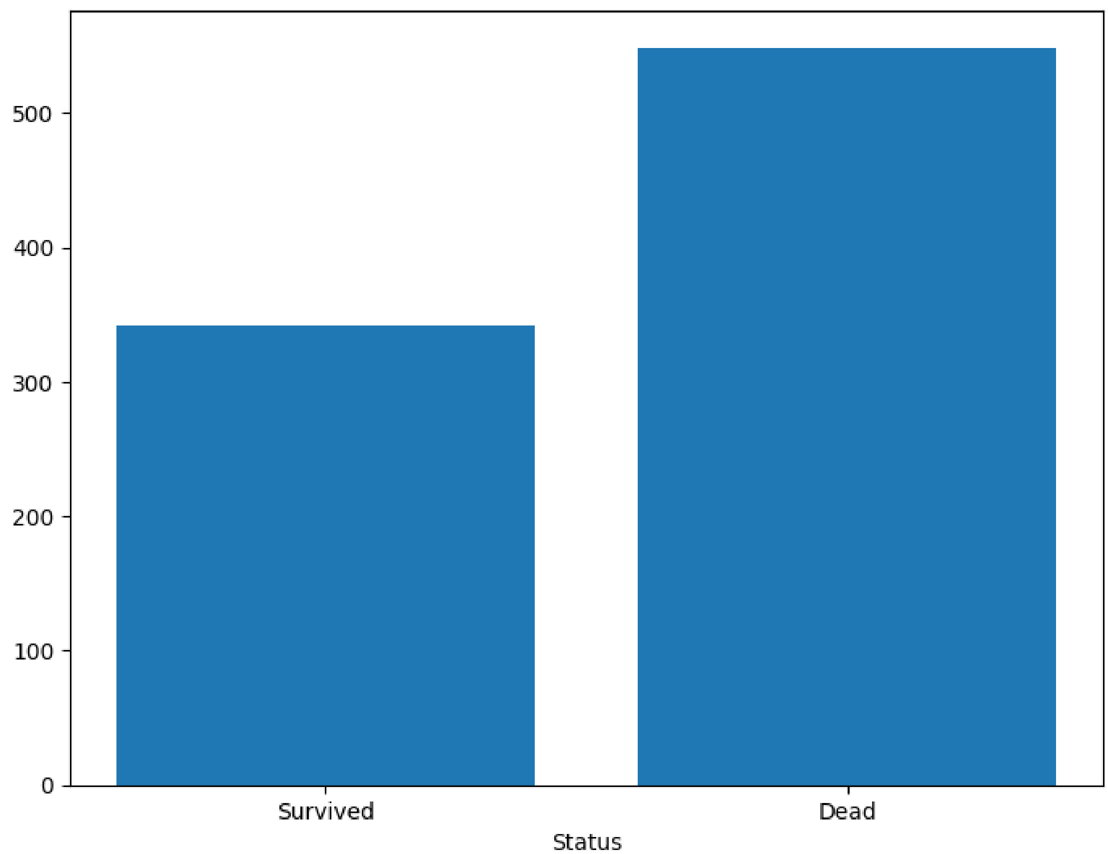
[14]:
```
alive = len(train[train['Survived'] == 1])
dead = len(train[train['Survived'] == 0])
```

[15]:
```
train.groupby('Sex')[['Survived']].mean()
```

[15]:

Sex	Survived
female	0.742038
male	0.188908

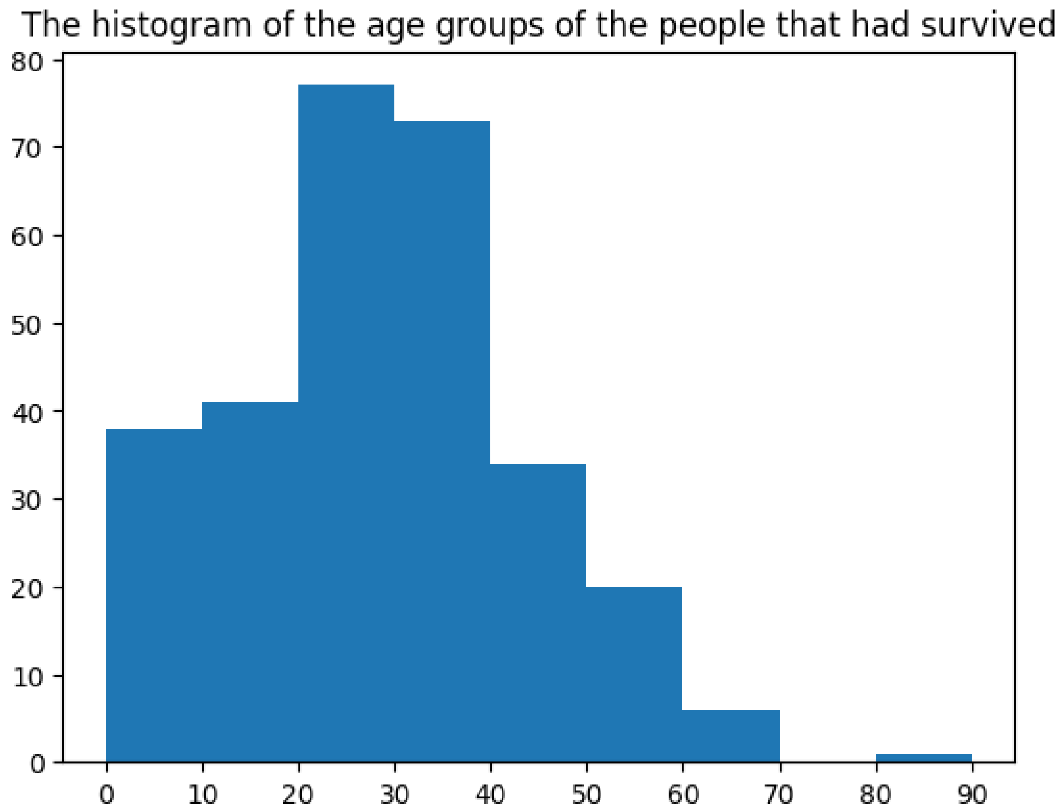
[16]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
status = ['Survived','Dead']
ind = [alive,dead]
ax.bar(status,ind)
plt.xlabel("Status")
plt.show()
```

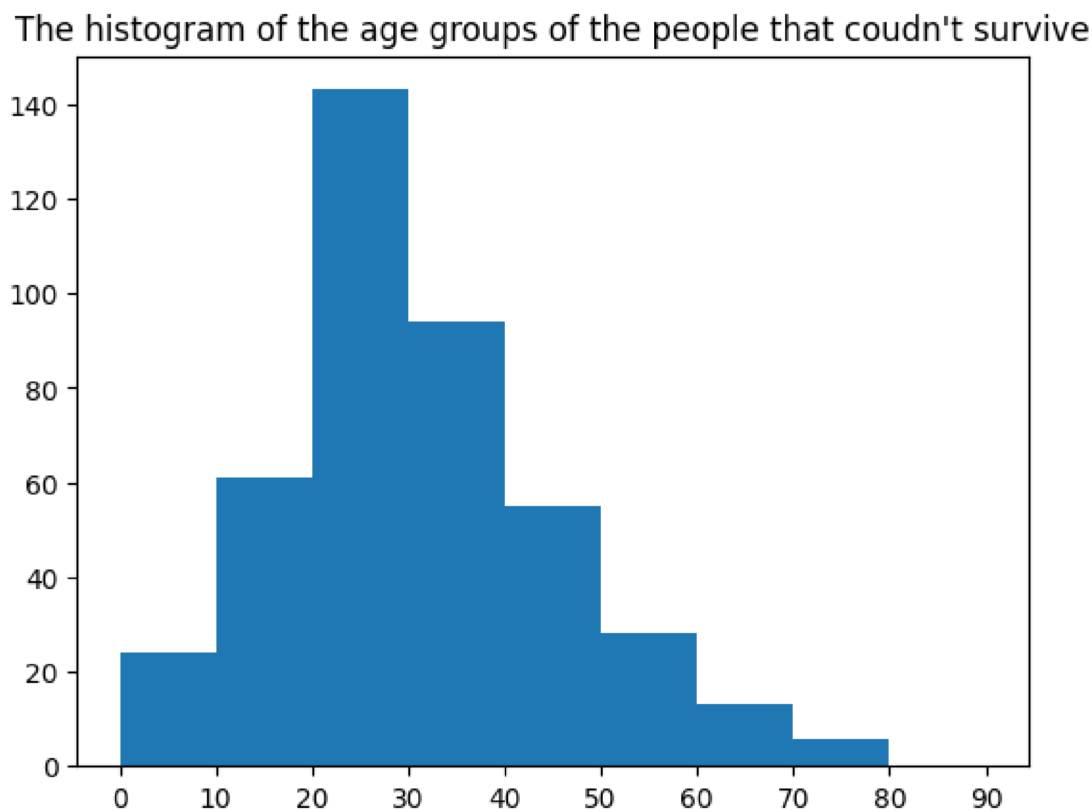


[17]:
```
plt.figure(1)
age  = train.loc[train.Survived == 1, 'Age']
plt.title('The histogram of the age groups of the people that had survived')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))


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))
```

[17]:
```
([<matplotlib.axis.XTick at 0x21b4b2c6900>,
  <matplotlib.axis.XTick at 0x21b4d905a60>,
  <matplotlib.axis.XTick at 0x21b4d8e5220>,
  <matplotlib.axis.XTick at 0x21b4d3da9f0>,
  <matplotlib.axis.XTick at 0x21b4d3db3e0>,
  <matplotlib.axis.XTick at 0x21b4d3dbd40>,
  <matplotlib.axis.XTick at 0x21b4d408680>,
  <matplotlib.axis.XTick at 0x21b4d3d86e0>,
  <matplotlib.axis.XTick at 0x21b4d408f80>,
  <matplotlib.axis.XTick at 0x21b4d4098e0>],
 [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
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90



[18]:
```
train[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().
↪sort_values(by='Survived', ascending=False)
```

[18]:

	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

[19]:
```
train[["Pclass", "Survived"]].groupby(['Pclass'], as_index=False).mean().
↪sort_values(by='Survived', ascending=False)
```

[19]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

[20]:
```
train[["Age", "Survived"]].groupby(['Age'], as_index=False).mean().
  ↪sort_values(by='Age', ascending=True)
```

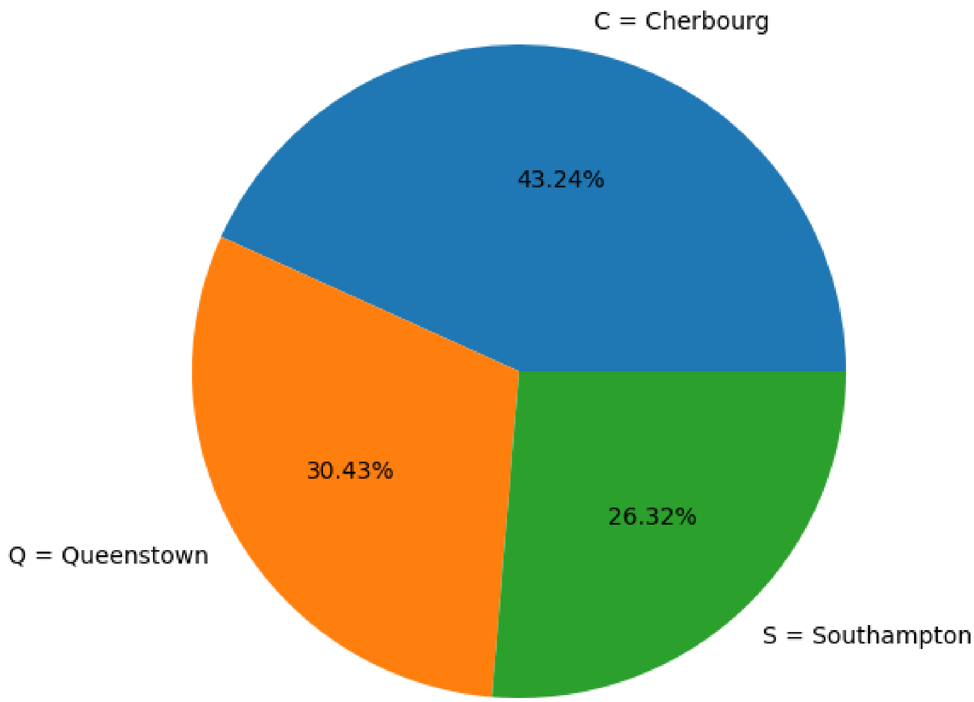
[20]:
```
      Age  Survived
0    0.42       1.0
1    0.67       1.0
2    0.75       1.0
3    0.83       1.0
4    0.92       1.0
..    …        …
83  70.00       0.0
84  70.50       0.0
85  71.00       0.0
86  74.00       0.0
87  80.00       1.0

[88 rows x 2 columns]
```

[21]:
```
train[["Embarked", "Survived"]].groupby(['Embarked'], as_index=False).mean().
  ↪sort_values(by='Survived', ascending=False)
```

[21]:
```
  Embarked  Survived
0        C  0.553571
1        Q  0.389610
2        S  0.336957
```

[22]:
```
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()
```



[23]:
```
test.describe(include="all")
```

[23]:
```
        PassengerId      Pclass                Name   Sex         Age  …
Parch    Ticket        Fare           Cabin Embarked
count    418.000000  418.000000              418   418  332.000000  …
418.000000       418  417.000000              91      418
unique          NaN         NaN              418     2         NaN  …
NaN       363        NaN                76        3
top             NaN         NaN  Kelly, Mr. James  male         NaN  …
NaN  PC 17608        NaN  B57 B59 B63 B66        S
freq            NaN         NaN                1   266         NaN  …
NaN         5        NaN                 3      270
mean    1100.500000    2.265550              NaN   NaN   30.272590  …
0.392344        NaN   35.627188             NaN      NaN
std      120.810458    0.841838              NaN   NaN   14.181209  …
0.981429        NaN   55.907576             NaN      NaN
min      892.000000    1.000000              NaN   NaN    0.170000  …
0.000000        NaN    0.000000             NaN      NaN
25%      996.250000    1.000000              NaN   NaN   21.000000  …
0.000000        NaN    7.895800             NaN      NaN
```

```
50%      1100.500000    3.000000              NaN    NaN    27.000000  …
0.000000        NaN    14.454200            NaN       NaN
75%      1204.750000    3.000000              NaN    NaN    39.000000  …
0.000000        NaN    31.500000            NaN       NaN
max      1309.000000    3.000000              NaN    NaN    76.000000  …
9.000000        NaN   512.329200            NaN       NaN

[11 rows x 11 columns]
```

[24]:
```
#Droping Useless Columns
train = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)
```

[25]:
```
train = train.drop(['Cabin'], axis = 1)
test = test.drop(['Cabin'], axis = 1)
```

[26]:
```
train = train.drop(['Name'], axis = 1)
test = test.drop(['Name'], axis = 1)
```

[27]:
```
#Feature Selection
column_train=['Age','Pclass','SibSp','Parch','Fare','Sex','Embarked']
#training values
X=train[column_train]
#target value
Y=train['Survived']
```

[28]:
```
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()
```

[28]: 2

[29]:
```
X['Age']=X['Age'].fillna(X['Age'].median())
X['Age'].isnull().sum()
```

[29]: 0

[30]:
```
X['Embarked'] = train['Embarked'].fillna(method ='pad')
X['Embarked'].isnull().sum()
```

[30]: 0

[31]:
```
d={'male':0, 'female':1}
X['Sex']=X['Sex'].apply(lambda x:d[x])
X['Sex'].head()
```

[31]:
```
0    0
1    1
2    1
3    1
4    0
Name: Sex, dtype: int64
```

[32]:
```
e={'C':0, 'Q':1 ,'S':2}
X['Embarked']=X['Embarked'].apply(lambda x:e[x])
X['Embarked'].head()
```

[32]:
```
0    2
1    0
2    2
3    2
4    2
Name: Embarked, dtype: int64
```

[33]:
```
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)
```

[34]:
```
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
```

[38]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

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

[39]:
```
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
```

[40]:
```
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
```

[41]:
```
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
```

[42]:
```
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))
```

```
[[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
```

[43]:
```
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
```

[44]:
```
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
```

[45]:
```
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
```

[46]:
```
from sklearn.metrics import␣
 ↪accuracy_score,confusion_matrix,classification_report
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
```

[47]:

```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive␣
↪Bayes','KNN' ,'Decision Tree'],
    '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)
```

[47]:

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

[ ]: