```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from warnings import filterwarnings
filterwarnings(action='ignore')
# In[3]:
pd.set_option('display.max_columns',10,'display.width',1000)
train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')
train.head()
# In[4]:
train.shape
# In[5]:
test.shape
# In[6]:
train.isnull().sum()
# In[8]:
test.isnull().sum()
# In[9]:
train.describe(include="all")
# In[17]:
male_ind=len(train[train['Sex']=='male'])
print("No of malemale_inds in titanic:",male_ind)
# In[19]:
male_ind=len(train[train['Sex']=='male'])
print("No of males in titanic:",male_ind)
# In[23]:
female_ind=len(train[train['Sex']=='female'])
print("No of females in titanic:",female_ind)
# In[27]:
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()
# In[34]:
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')
# In[35]:
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
coultdn\'survive')
plt.hist(age,np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))
# In[37]:
train[["SibSp","Survived"]].groupby(['SibSp'],as_index=False).mean().sort
_values(by='Survived',ascending=False)
# In[38]:
train[["Pclass","Survived"]].groupby(['Pclass'],as_index=False).mean().so
rt_values(by='Survived',ascending=False)
# In[39]:
train[["Age","Survived"]].groupby(['Age'],as_index=False).mean().sort_val
ues(by='Age',ascending=True)
# In[40]:
train[["Embarked","Survived"]].groupby(['Embarked'],as_index=False).mean(
).sort_values(by='Survived',ascending=True)
# In[41]:
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.axis('equal')
l=['C=Cherbourg','Q=Queenstown','S=Southampton']
s=[0.336957,0.389610,0.553571]
ax.pie(s,labels=l,autopct='%1.2f')
plt.show()
# In[42]:
test.describe(include="all")
# In[43]:
train=train.drop(['Ticket'],axis=1)
test=test.drop(['Ticket'],axis=1)
# In[44]:
train=train.drop(['Cabin'],axis=1)
test=test.drop(['Cabin'],axis=1)
# In[45]:
train=train.drop(['Name'],axis=1)
test=test.drop(['Name'],axis=1)
# In[46]:
column_train=['Age','Pclass','SibSp','Parch','Fare','Sex','Embarked']
X=train[column_train]
Y=train['Survived']
# In[49]:
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()
# In[57]:
X['Age']=X['Age'].fillna(X['Age'].median())
X['Age'].isnull().sum()
# In[58]:
X['Embarked']=train['Embarked'].fillna(method='pad')
X['Embarked'].isnull().sum()
# In[63]:
d={'male':0,'female':1}
```

```
X['Sex']=X['Sex'].apply(lambda x:d[x])
X['Sex'].head()
# In[65]:
e={'C':0,'Q':1,'S':2}
X['Embarked']=X['Embarked'].apply(lambda x:e[x])
X['Embarked'].head()
# In[66]:
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
tate=7)
# In[67]:
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))
# In[68]:
from sklearn.metrics import accuracy_score,confusion_matrix
confusion_mat=confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
# In[70]:
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))
# In[71]:
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))
# In[72]:
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))
# In[73]:
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))
# In[74]:
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))
# In[75]:
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
confusion_mat=confusion_matrix(A,y_pred3)
```

```
print(confusion_mat)
print(classification_report(Y_test,y_pred3))
# In[78]:
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))
# In[81]:
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))
# In[84]:
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)
# In[ ]:
```