GroupB

Assignment 1 sort merge

```
import numpy as mp
import pandas as pd
data=pd.read_csv(r"D:\DSDBA\dataset_Facebook.csv", sep=";")
data
data.describe()
data.shape
subset1=data[['Page total likes' , 'Category' , 'Post Month', 'Post Weekday']].loc[0:15]
subset1
subset2=data[['Page total likes' , 'Category' , 'Post Month', 'Post Weekday']].loc[16:30]
subset2
subset3=data[['Page total likes' , 'Category' , 'Post Month', 'Post Weekday']].loc[31:50]
subset3
merging=pd.concat([subset1,subset2,subset3])
merging
sort_values=data.sort_values('Page total likes', ascending=False)
sort_values
sort_values=data.sort_values('Page total likes')
sort_values
transposing=data.transpose()
transposing
data.shape
nrows=data.shape[0]
ncols=data.shape[1]
print(f"the data set consists of {nrows} rows and {ncols} column in dataset ")
pivot_table=pd.pivot_table(data,index=['Type', 'Category'], values='comment')
pivot_table
reshapping_arr=mp.array([1,2,3,4,5,6,7,8,9,10])
reshapping_arr.reshape(5,2)
```

Assignment 2. Heart and air quality

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
# In[3]:
data=pd.read_csv('heart.csv')
data.head()
# In[4]:
data.tail()
# In[6]:
data.shape
# In[7]:
data.columns
# In[8]:
data.info()
# In[9]:
data.describe()
# In[10]:
data.isnull().sum()
# In[11]:
duplicate_rows=data[data.duplicated()]
duplicate_sort=duplicate_rows.sort_values(by=['age'])
print(duplicate_sort)
# In[12]:
data.duplicated().sum()
# In[13]:
plt.figure(figsize=(10,3))
sns.boxplot(data['trestbps'])
# In[14]:
```

```
q1=data['trestbps'].quantile(0.25)
q3=data['trestbps'].quantile(0.75)
IQR=q3-q1
lower_limit=q1-1.5*IQR
upper_limit=q3+1.5*IQR
data=data[(data['trestbps']>lower_limit) & (data['trestbps']<upper_limit)]
plt.figure(figsize=(10,3))
sns.boxplot(data['trestbps'])
# In[15]:
plt.figure(figsize=(10,3))
sns.boxplot(data['chol'])
# In[16]:
q1=data['chol'].quantile(0.25)
q3=data['chol'].quantile(0.75)
IQR=q3-q1
lower_limit=q1-1.5*IQR
upper_limit=q3+1.5*IQR
data=data[(data['chol']>lower_limit) & (data['chol']<upper_limit)]
plt.figure(figsize=(10,3))
sns.boxplot(data['chol'])
# In[17]:
q1=data['oldpeak'].quantile(0.25)
q3=data['oldpeak'].quantile(0.75)
IQR=q3-q1
lower_limit=q1-1.5*IQR
upper_limit=q3+1.5*IQR
data=data[(data['oldpeak']>lower_limit) & (data['oldpeak']<upper_limit)]
plt.figure(figsize=(10,3))
sns.boxplot(data['oldpeak'])
# In[19]:
plt.figure(figsize=(15,6))
```

```
sns.boxplot(data=data,orient='h')

# In[21]:

data['target'].value_counts()

# In[22]:

f,ax=plt.subplots(figsize=(30,25))

corr=data.corr()

matt=data.corr()

mask=np.zeros_like(corr,dtype=np.bool)

cmap=sns.diverging_palette(230,20,as_cmap=True)

sns.heatmap(matt,mask=mask,cmap=cmap,vmax=1,center=0,annot=True,square=True,linewidths=7,
cbar_kws={"shrink": .5})

plt.xticks(rotation=90,fontsize=15)

plt.yticks(fontsize=15)

plt.show()

# In[23]:

sns.lmplot(data=data,x="slope",y="target")

# In[24]:

sns.lmplot(data=data,x="thalach",y="target")

# In[25]:

sns.pairplot(data=data,hue="target")

# In[26]:

x=data.drop('target',axis=1)

x

# In[27]:

y=data['target']

y

# In[29]:

from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import chi2

bestfeatures= SelectKBest(score_func=chi2)

# In[30]:
```

```
fit=bestfeatures.fit(x,y)
scores=pd.DataFrame(fit.scores_)
# In[31]:
dfcolumns=pd.DataFrame(x.columns)
# In[35]:
featureScores=pd.concat([dfcolumns,scores],axis=1)
featureScores.columns=['Label','Score']
featureScores.sort_values(by='Score',ascending=False)
# In[36]:
new_sex=pd.get_dummies(data=data['sex'],prefix='sex')
new_sex
# In[37]:
new_cp=pd.get_dummies(data=data['cp'],prefix='chestPain')
new_cp
# In[38]:
new_exang=pd.get_dummies(data=data['exang'],prefix='exang')
new_exang
# In[39]:
new_slope=pd.get_dummies(data=data['slope'],prefix='slope')
new_slope
# In[40]:
new_thal=pd.get_dummies(data=data['thal'],prefix='thal')
new_thal
# In[41]:
new_ca=pd.get_dummies(data=data['ca'],prefix='ca')
new_ca
# In[42]:
app=[data,new_sex,new_cp,new_ca,new_thal,new_exang,new_slope]
df1=pd.concat(app,axis=1)
df1.head()
# In[43]:
```

```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

# In[45]:

df1[['age','trestbps','chol','oldpeak','thalach']]=sc.fit_transform(df1[['age','trestbps','chol','oldpeak','thalach']])

# In[46]:

X=df1.drop('target',axis=1)

X

# In[47]:

Y=df1['target']

Y

# In[49]:

scaler=StandardScaler()

scaler.fit(X)

x_scaled=scaler.transform(X)

# In[51]:

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2)

print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)

# In[53]:

from sklearn.datasets import load_iris

from sklearn.model_selection import cross_val_score,KFold

from sklearn.linear_model import LogisticRegression

iris=load_iris()

X=x=X_train

Y=y_train

logreg=LogisticRegression()

kf=KFold(n_splits=5)

score=cross_val_score(logreg,X,Y,cv=kf)

print("cross validation scores are {}".format(score))

print("average cross validation score :{}".format(score.mean()))
```

```
# In[54]:
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold,cross_val_score
clf=DecisionTreeClassifier(random_state=42)
k_folds=KFold(n_splits=5)
scores=cross_val_score(clf,X_train,y_train,cv=k_folds)
print("cross validation scores: ",scores)
print("average cv score: ",scores.mean())
print("number of cv scores used in average: ",len(scores))
# In[55]:
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(X_train,y_train)
print("Logistic Regression train score: {:.4f}".format(logreg.score(X_train,y_train)))
print("Logistic Regression test score: {:.4f}".format(logreg.score(X_test,y_test)))
# In[58]:
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(max_depth=8,random_state=0)
tree.fit(X_train,y_train)
print("Desicion tree train score: {:.4f}".format(tree.score(X_train,y_train)))
print("Decision Tree test score: {:.4f}".format(tree.score(X_test,y_test)))
# In[59]:
print("Logistic Regression test score: {:.4f}".format(logreg.score(X_test,y_test)))
print("Decision Tree test score: {:.4f}".format(tree.score(X_test,y_test)))
# In[60]:
pred_logreg=logreg.predict(X_test)
pred_tree=tree.predict(X_test)
# In[61]:
from sklearn.metrics import confusion_matrix
cm_logreg=confusion_matrix(y_test,pred_logreg)
```

```
cm_tree=confusion_matrix(y_test,pred_tree)

# In[63]:

plt.subplot(1,5,2)

plt.title("Logistic Regression")

sns.heatmap(cm_logreg,annot=True,cmap="Reds",fmt="d",cbar=False,annot_kws={'size': 20,
'rotation': 45})

plt.subplot(1,5,5)

plt.title("Decision Tree")

sns.heatmap(cm_tree,annot=True,cmap="Reds",fmt="d",cbar=False,annot_kws={'size': 20, 'rotation':
45})

plt.show()

# In[66]:

from sklearn.metrics import classification_report

print('_____\n\nLogistic Regression')

print('_____')

print(classification_report(y_test,pred_logreg,target_names=["have disease","dont have disease"]))

print('_____\n\nDecision Tree')

print('_____')

print(classification_report(y_test,pred_tree,target_names=["have disease","dont have disease"]))

# In[ ]:
```

Assignment 4 visualization

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

dataset1=pd.read_csv('heart.csv')

dataset2=pd.read_csv('AirQuality.csv',sep=";")

# In[2]:

dataset1.head()

# In[3]:

dataset2.head()
```

```
# In[4]:

sns.pairplot(dataset2)

plt.show()

# In[6]:

f,ax=plt.subplots(figsize=(30,25))

corr=dataset1.corr()

matt=dataset1.corr()

mask=np.zeros_like(corr,dtype=np.bool)

cmap=sns.diverging_palette(230,20,as_cmap=True)

sns.heatmap(matt,mask=mask,cmap=cmap,vmax=1,center=0,annot=True,square=True,linewidths=7,
cbar_kws={"shrink": .5})

plt.xticks(rotation=90,fontsize=15)

plt.yticks(fontsize=15)

plt.title("correlation heatmap of heart disease")

plt.show()

# In[7]:

sns.boxplot(x='target',y='age',data=dataset1)

plt.title('Boxplot of Age by Heart Disease Diagnosis')

plt.xlabel('Heart Disease Diagnosis')

plt.ylabel('Age')

plt.show()

# In[8]:

sns.scatterplot(x='chol',y='trestbps',data=dataset1)

plt.title('Scatterplot of cholesterol and max heart rate')

plt.xlabel('Cholesterol')

plt.ylabel('Max Heart Rate')

plt.show()

# In[9]:

plt.hist(dataset1['age'],bins=10)

plt.title('Histogram of column age')

plt.xlabel('value')
```

```
plt.ylabel('Frequency')
plt.show()
# In[13]:
sns.countplot(x='cp',data=dataset1)
plt.title('Bar chart of column name')
plt.xlabel('value')
plt.ylabel('Frequency')
plt.show()
# In[14]:
plt.hist(dataset2['CO(GT)'].dropna(),bins=10)
plt.xlabel('CO Concentration')
plt.ylabel('Count')
plt.title('Histogram of CO concentration')
plt.xticks([0,20,40,60,80,100],['0','20','40','60','80','100'])
plt.show()
# In[15]:
sns.boxplot(y=dataset2['PT08.S1(CO)'].dropna().astype(float))
plt.ylabel('CO Concentration')
plt.title('Box plot of co concentration')
plt.show()
# In[17]:
sns.scatterplot(x='NO2(GT)',y='CO(GT)',data=dataset2.dropna(subset=['NO2(GT)','CO(GT)']))
plt.xlabel('CO Cnocentration')
plt.ylabel('NO2 Concentration')
plt.title('scatter plot of NO2 an CO concentration')
plt.yticks([0,20,40,60,80,100],['0','20','40','60','80','100'])
plt.show()
# In[18]:
sns.violinplot(y='NMHC(GT)',data=dataset2.dropna(subset=['NMHC(GT)']))
plt.ylabel('NMHC Concentration')
plt.show()
```

Assignment 5(word count python integrate)

```
cd Documents
touch word_count.txt
nano word_count.txt
cat word_count.txt
touch mapper.py
nano mapper.py
cat mapper.py
cat word_count.txt | python3 mapper.py
nano reducer.py
cat reducer.py
cat word_count.txt | python3 mapper.py | sort -k1,1 | python3 reducer.py
start-all.sh
hdfs dfs -mkdir /word_count_in_python
hdfs dfs -copyFromLocal /home/user/Documents/word_count.txt /word_count_in_python
hdfs dfs -ls /
hdfs dfs -ls /word_count_in_python
hdfs dfs -ls /
hdfs dfs -ls /word_count_in_python
chmod 777 mapper.py reducer.py
hadoop jar /home/user/Documents/hadoop-streaming-3.3.4.jar -input
/word_count_in_python/word_count.txt -output /word_count_in_python/output -mapper
/home/user/Documents/mapper.py -reducer/home/user/Documents/reducer.py
```

Assignment group c(1)

```
import requests
import bs4
# In[2]:
request1=requests.get('https://www.flipkart.com/poco-c31-royal-blue-64-gb/p/itm19effae969b86')
request1
# In[3]
```

```
request1.content
# In[5]:
soup=bs4.BeautifulSoup(request1.text)
soup
# In[6]:
reviews=soup.find_all('div',{'class':'t-ZTKy'});
for review in reviews:
    print(review.get_text()+"\n\n")
# In[7]:
rating=soup.find('div',{'class':'_2d4LTz'}).get_text();
print(rating)
# In[8]:
individual_rating=soup.findAll('div',{'class':'_3LWZlK _1BLPMq'});
for indi_rating in individual_rating:
    print(indi_rating.get_text()+"\n")
# In[9]:
tags=soup.find('span',{'class':'yhB1nd GXgmTe'}).get_text();
tags
# In[10]:
customer_name=soup.findAll('p',{'class':'_2sc7ZR _2V5EHH'});
for cust_name in customer_name:
    print(cust_name.get_text()+"\n")
# In[17]:
price=soup.find('div',{'class':'_30jeq3 _16Jk6d'}).get_text();
price
# In[19]:
quetions=soup.findAll('div',{'class':'_1xR0kG _3cziW5'});
for que in quetions:
    print(que.get_text())
    answer=que.findNext('div',{'class':'_2yeNfb'}).get_text();
    print(answer+"\n\n")
```