import pandas as pd  
import numpy as np  
df=pd.read\_csv('C:\\Users\\rajkalyan\\Desktop\\py\\student\\Student\_data.csv')  
df.info()  
print(df.head())  
print(df.shape)  
print(df.columns)  
for col in df.columns:  
 if df[col].dtype == 'object':  
 print('\nColumn Name:', col,)  
 print(df[col].value\_counts())  
print(df['MC'].describe())  
import seaborn as sns  
import matplotlib.pyplot as plt  
#b = sns.countplot(df['MC'])  
b=sns.countplot(x='MC', data=df)  
b.axes.set\_title('Distribution of MC marks of students', fontsize= 25)  
b.set\_xlabel('MC marks of all the students', fontsize = 20)  
b.set\_ylabel('count', fontsize = 20)  
plt.show()  
  
  
print(df['IP'].describe())  
b=sns.countplot(x='IP', data=df)  
b.axes.set\_title('Distribution of IP marks of students', fontsize= 25)  
b.set\_xlabel('IP marks of all the students', fontsize = 20)  
b.set\_ylabel('count', fontsize = 20)  
plt.show()  
print(df['PCOL'].describe())  
b=sns.countplot(x='PCOL', data=df)  
b.axes.set\_title('Distribution of PCOL marks of students', fontsize= 25)  
b.set\_xlabel('PCOL marks of all the students', fontsize = 20)  
b.set\_ylabel('count', fontsize = 20)  
plt.show()  
print(df['PCOG'].describe())  
b=sns.countplot(x='PCOG', data=df)  
b.axes.set\_title('Distribution of PCOG marks of students', fontsize= 25)  
b.set\_xlabel('PCOG marks of all the students', fontsize = 20)  
b.set\_ylabel('count', fontsize = 20)  
plt.show()  
print(df['PJ'].describe())  
b=sns.countplot(x='PJ', data=df)  
b.axes.set\_title('Distribution of PJ marks of students', fontsize= 25)  
b.set\_xlabel('PJ marks of all the students', fontsize = 20)  
b.set\_ylabel('count', fontsize = 20)  
plt.show()  
  
print(df.isnull().any())  
print(df.isnull().sum())  
df['TOT'] = (df['MC']+df['IP']+df['PCOL']+df['PCOG']+df['PJ'])  
print(df['TOT'])  
print(df['TOT'].describe())  
b=sns.countplot(x='TOT', data=df)  
b.axes.set\_title('Distribution of Total marks of students', fontsize= 25)  
b.set\_xlabel('Total marks of all the students', fontsize = 20)  
b.set\_ylabel('count', fontsize = 20)  
plt.show()  
#scatter plots for subjects with tot  
plt.scatter(df.MC,df.TOT)  
plt.title('MC VS TOT')  
plt.xlabel('MC')  
plt.ylabel('TOT')  
plt.show()  
plt.scatter(df.IP,df.TOT)  
plt.title('IP VS TOT')  
plt.xlabel('IP')  
plt.ylabel('TOT')  
plt.show()  
plt.scatter(df.PCOG,df.TOT)  
plt.title('PCOG VS TOT')  
plt.xlabel('PCOG')  
plt.ylabel('TOT')  
plt.show()  
plt.scatter(df.PCOL,df.TOT)  
plt.title('PCOL VS TOT')  
plt.xlabel('PCOL')  
plt.ylabel('TOT')  
plt.show()  
  
plt.scatter(df.PJ,df.TOT)  
plt.title('PJ VS TOT')  
plt.xlabel('PJ')  
plt.ylabel('TOT')  
plt.show()  
  
df.head()  
# best fit line and calculation  
X = df.drop(['S.No','Roll num','Name','IP','PCOG', 'PCOL', 'PJ', 'TOT'], axis=1)  
print(X)  
#y = df.drop(['S.No','Roll num','Name','MC','IP','PCOG', 'PCOL', 'PJ'], axis=1)  
y = df.iloc[:, -1]  
print(y)  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  
  
#Fitting Linear Regression to the Training set  
from sklearn.linear\_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X\_train, y\_train)  
print("Intercept value: ", regressor.intercept\_)  
print("coefficient value: ", regressor.coef\_)  
# Predicting the Test set results  
y\_pred = regressor.predict(X\_test)  
data=[{'Actual': y\_test, 'predicted': y\_pred}]  
output\_df=pd.DataFrame(data)  
print(output\_df)  
# plot  
plt.scatter(X\_test, y\_test,color='black')  
plt.plot (X\_test, y\_pred,color='blue', linewidth=3)  
plt.xticks(())  
plt.yticks(())  
plt.title('MC versus TOT')  
plt.xlabel('MC')  
plt.ylabel('TOT')  
plt.show()  
from sklearn import metrics  
explained\_variance=metrics.explained\_variance\_score(y\_test, y\_pred)  
mean\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mse=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mean\_squared\_log\_error=metrics.mean\_squared\_log\_error(y\_test, y\_pred)  
median\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
r2=metrics.r2\_score(y\_test, y\_pred)  
print('Explained\_variance: ', round(explained\_variance,2))  
print('Mean\_Squared\_Log\_Error: ',  
round(mean\_squared\_log\_error,2))  
print('R-squared: ', round(r2,4))  
print('Mean Absolute Error(MAE): ',  
round(mean\_absolute\_error,2))  
print('Mean Squared Error (MSE): ', round(mse,2))  
print('Root Mean Squared Error (RMSE): ',  
round(np.sqrt(mse),2))  
  
# now between IP and TOT  
X= df.drop(['S.No','Roll num','Name','MC','PCOG', 'PCOL', 'PJ', 'TOT'], axis=1)  
y = df.iloc[:, -1]  
  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  
  
# Fitting Linear Regression to the Training set  
from sklearn.linear\_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X\_train, y\_train)  
print("Intercept value: ", regressor.intercept\_)  
print("coefficient value: ", regressor.coef\_)  
# Predicting the Test set results  
y\_pred = regressor.predict(X\_test)  
print("Intercept value: ", regressor.intercept\_)  
print("coefficient value: ", regressor.coef\_)  
# Predicting the Test set results  
y\_pred = regressor.predict(X\_test)  
data=[{'Actual': y\_test, 'predicted': y\_pred}]  
output\_df=pd.DataFrame(data)  
print(output\_df)  
# plot  
plt.scatter(X\_test, y\_test,color='black')  
plt.plot (X\_test, y\_pred,color='blue', linewidth=3)  
plt.xticks(())  
plt.yticks(())  
plt.title('IP versus TOT')  
plt.xlabel('IP')  
plt.ylabel('TOT')  
plt.show()  
from sklearn import metrics  
explained\_variance=metrics.explained\_variance\_score(y\_test, y\_pred)  
mean\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mse=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mean\_squared\_log\_error=metrics.mean\_squared\_log\_error(y\_test, y\_pred)  
median\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
r2=metrics.r2\_score(y\_test, y\_pred)  
print('Explained\_variance: ', round(explained\_variance,2))  
print('Mean\_Squared\_Log\_Error: ',  
round(mean\_squared\_log\_error,2))  
print('R-squared: ', round(r2,4))  
print('Mean Absolute Error(MAE): ',  
round(mean\_absolute\_error,2))  
print('Mean Squared Error (MSE): ', round(mse,2))  
print('Root Mean Squared Error (RMSE): ',  
round(np.sqrt(mse),2))  
from sklearn.metrics import r2\_score  
score=r2\_score(y\_test,y\_pred)  
print(score)  
  
# between PCOG and TOT  
X = df.drop(['S.No','Roll num','Name','MC','IP', 'PCOL', 'PJ', 'TOT'], axis=1)  
y = df.iloc[:, -1]  
  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  
  
# Fitting Linear Regression to the Training set  
from sklearn.linear\_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X\_train, y\_train)  
print("Intercept value: ", regressor.intercept\_)  
print("coefficient value: ", regressor.coef\_)  
# Predicting the Test set results  
y\_pred = regressor.predict(X\_test)  
data=[{'Actual': y\_test, 'predicted': y\_pred}]  
output\_df=pd.DataFrame(data)  
print(output\_df)  
# plot  
plt.scatter(X\_test, y\_test,color='black')  
plt.plot (X\_test, y\_pred,color='blue', linewidth=3)  
plt.xticks(())  
plt.yticks(())  
plt.title('PCOG versus TOT')  
plt.xlabel('PCOG')  
plt.ylabel('TOT')  
plt.show()  
from sklearn import metrics  
explained\_variance=metrics.explained\_variance\_score(y\_test, y\_pred)  
mean\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mse=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mean\_squared\_log\_error=metrics.mean\_squared\_log\_error(y\_test, y\_pred)  
median\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
r2=metrics.r2\_score(y\_test, y\_pred)  
print('Explained\_variance: ', round(explained\_variance,2))  
print('Mean\_Squared\_Log\_Error: ',  
round(mean\_squared\_log\_error,2))  
print('R-squared: ', round(r2,4))  
print('Mean Absolute Error(MAE): ',  
round(mean\_absolute\_error,2))  
print('Mean Squared Error (MSE): ', round(mse,2))  
print('Root Mean Squared Error (RMSE): ',  
round(np.sqrt(mse),2))  
from sklearn.metrics import r2\_score  
score=r2\_score(y\_test,y\_pred)  
print(score)  
# between PCOL and TOT  
X = df.drop(['S.No','Roll num','Name','MC','IP', 'PCOG', 'PJ', 'TOT'], axis=1)  
y = df.iloc[:, -1]  
  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  
  
# Fitting Linear Regression to the Training set  
from sklearn.linear\_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X\_train, y\_train)  
print("Intercept value: ", regressor.intercept\_)  
print("coefficient value: ", regressor.coef\_)  
# Predicting the Test set results  
y\_pred = regressor.predict(X\_test)  
data=[{'Actual': y\_test, 'predicted': y\_pred}]  
output\_df=pd.DataFrame(data)  
print(output\_df)  
# plot  
plt.scatter(X\_test, y\_test,color='black')  
plt.plot (X\_test, y\_pred,color='blue', linewidth=3)  
plt.xticks(())  
plt.yticks(())  
plt.title('PCOL versus TOT')  
plt.xlabel('PCOL')  
plt.ylabel('TOT')  
plt.show()  
from sklearn import metrics  
explained\_variance=metrics.explained\_variance\_score(y\_test, y\_pred)  
mean\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mse=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mean\_squared\_log\_error=metrics.mean\_squared\_log\_error(y\_test, y\_pred)  
median\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
r2=metrics.r2\_score(y\_test, y\_pred)  
print('Explained\_variance: ', round(explained\_variance,2))  
print('Mean\_Squared\_Log\_Error: ',  
round(mean\_squared\_log\_error,2))  
print('R-squared: ', round(r2,4))  
print('Mean Absolute Error(MAE): ',  
round(mean\_absolute\_error,2))  
print('Mean Squared Error (MSE): ', round(mse,2))  
print('Root Mean Squared Error (RMSE): ',  
round(np.sqrt(mse),2))  
from sklearn.metrics import r2\_score  
score=r2\_score(y\_test,y\_pred)  
print(score)  
  
  
# between PJ and TOT  
  
X = df.drop(['S.No','Roll num','Name','MC','IP', 'PCOL', 'PCOG', 'TOT'], axis=1)  
y = df.iloc[:, -1]  
  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  
  
# Fitting Linear Regression to the Training set  
from sklearn.linear\_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X\_train, y\_train)  
print("Intercept value: ", regressor.intercept\_)  
print("coefficient value: ", regressor.coef\_)  
# Predicting the Test set results  
y\_pred = regressor.predict(X\_test)  
data=[{'Actual': y\_test, 'predicted': y\_pred}]  
output\_df=pd.DataFrame(data)  
print(output\_df)  
# plot  
plt.scatter(X\_test, y\_test,color='black')  
plt.plot (X\_test, y\_pred,color='blue', linewidth=3)  
plt.xticks(())  
plt.yticks(())  
plt.title('PJ versus TOT')  
plt.xlabel('PJ')  
plt.ylabel('TOT')  
plt.show()  
from sklearn import metrics  
explained\_variance=metrics.explained\_variance\_score(y\_test, y\_pred)  
mean\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mse=metrics.mean\_absolute\_error(y\_test, y\_pred)  
mean\_squared\_log\_error=metrics.mean\_squared\_log\_error(y\_test, y\_pred)  
median\_absolute\_error=metrics.mean\_absolute\_error(y\_test, y\_pred)  
r2=metrics.r2\_score(y\_test, y\_pred)  
print('Explained\_variance: ', round(explained\_variance,2))  
print('Mean\_Squared\_Log\_Error: ',  
round(mean\_squared\_log\_error,2))  
print('R-squared: ', round(r2,4))  
print('Mean Absolute Error(MAE): ',  
round(mean\_absolute\_error,2))  
print('Mean Squared Error (MSE): ', round(mse,2))  
print('Root Mean Squared Error (RMSE): ',  
round(np.sqrt(mse),2))  
from sklearn.metrics import r2\_score  
score=r2\_score(y\_test,y\_pred)  
print(score)  
  
  
  
  
#plt.scatter(df.MC,df.IP,df.PCOG,df.PCOL,df.PJ,df.TOT)  
#23  
  
  
#X = df.drop(['S.No','Roll num','TOT'], axis=1)  
#y = df.iloc[:, -1]  
  
#from sklearn.model\_selection import train\_test\_split  
#X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  
  
# Fitting Linear Regression to the Training set  
#from sklearn.linear\_model import LinearRegression  
#regressor = LinearRegression()  
#regressor.fit(X\_train, y\_train)  
  
# Predicting the Test set results  
#y\_pred = regressor.predict(X\_test)  
  
#from sklearn.metrics import r2\_score  
#score=r2\_score(y\_test,y\_pred)  
#print(score)  
#from sklearn import preprocessing  
#creating labelEncoder  
#le = preprocessing.LabelEncoder()  
#for i in df.loc[:,df.dtypes==object].columns:  
 # df[i]=le.fit\_transform(df[i])  
#print(df\_new.head())  
#from sklearn.model\_selection import train\_test\_split  
#x=df\_new.drop("TOT",axis=1)  
#y=df\_new['TOT']  
#X\_train, X\_test, y\_train, y\_test= train\_test\_split(x,y, test\_size= 0.25, random\_state=42)  
#from sklearn.linear\_model import LinearRegression  
#L=LinearRegression()  
#L.fit(X\_train, y\_train)  
#LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)  
#y\_pred=L.predict(X\_test)  
#from sklearn.metrics import accuracy\_score,mean\_absolute\_error,mean\_squared\_error,r2\_score  
  
#print(accuracy\_score(y\_test,y\_pred))  
#print(mean\_absolute\_error(y\_test,y\_pred))  
#print(mean\_squared\_error(y\_test,y\_pred))  
#print(r2\_score(y\_test,y\_pred))  
#print(L.score(X\_test,y\_test))  
  
  
  
  
#x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=0)  
#model1=RandomForestRegressor()  
#model1.fit(x\_train,y\_train)  
#predictions=model1.predict(x\_test)  
#print("Accuracy for RandomForest:",r2\_score(predictions,y\_test))  
  
#model2=GaussianNB()  
#model2.fit(x\_train,y\_train)  
#predictions=model2.predict(x\_test)  
#print("Accuracy for Gaussian:",r2\_score(predictions,y\_test))  
  
  
#model2=SVC()  
#model2.fit(x\_train,y\_train)  
#predictions=model2.predict(x\_test)  
#print("Accuracy for SVC :",r2\_score(predictions,y\_test))

"C:\Users\rajkalyan\PycharmProjects\python project 1\venv\Scripts\python.exe" C:/Users/rajkalyan/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/performance.py

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 103 entries, 0 to 102

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 S.No 103 non-null int64

1 Roll num 103 non-null object

2 Name 103 non-null object

3 MC 103 non-null int64

4 IP 103 non-null int64

5 PCOL 103 non-null int64

6 PCOG 103 non-null int64

7 PJ 103 non-null int64

dtypes: int64(6), object(2)

memory usage: 6.6+ KB

S.No Roll num Name MC IP PCOL PCOG PJ

0 1 1702 18 881 001 AKANKSHA TRIPATHI 21 20 22 22 22

1 2 1702 18 881 002 KARDAS AKHILA 21 20 22 23 22

2 3 1702 18 881 003 RAPAKA AKHILA 22 20 22 21 21

3 4 1702 18 881 004 MERUGU AKHILA SHIVANI 18 17 19 22 20

4 5 1702 18 881 005 TANNERU AMRUTHA 13 9 13 12 13

(103, 8)

Index(['S.No', 'Roll num', 'Name', 'MC', 'IP', 'PCOL', 'PCOG', 'PJ'], dtype='object')

Column Name: Roll num

1702 18 881 001 1

1702 18 881 068 1

1702 18 881 080 1

1702 18 881 079 1

1702 18 881 076 1

..

1702 18 881 033 1

1702 18 881 032 1

1.70E+11 1

1702 18 881 030 1

1702 17 881 077 1

Name: Roll num, Length: 103, dtype: int64

Column Name: Name

AKANKSHA TRIPATHI 1

KATASANI BHARGAV REDDY 1

JEEDIPALLY SUCHITH BHARATH 1

SIVA SUBASH CHOWDARY NEKKANTI 1

KANNEGANTI REVANTH KUMAR 1

..

V.RACHANA PRAVALIKA 1

P PRANATHI 1

JALIGI NISHITHA 1

CH.NIKHITA 1

VATSAVAI NEEHARIKA 1

Name: Name, Length: 103, dtype: int64

count 103.000000

mean 19.048544

std 3.687229

min 7.000000

25% 17.000000

50% 19.000000

75% 22.000000

max 25.000000

Name: MC, dtype: float64

count 103.000000

mean 18.446602

std 3.342326

min 5.000000

25% 17.000000

50% 18.000000

75% 21.000000

max 24.000000

Name: IP, dtype: float64

count 103.000000

mean 20.543689

std 2.667071

min 13.000000

25% 19.000000

50% 21.000000

75% 22.500000

max 25.000000

Name: PCOL, dtype: float64

count 103.000000

mean 20.281553

std 3.062799

min 8.000000

25% 18.500000

50% 21.000000

75% 23.000000

max 24.000000

Name: PCOG, dtype: float64

count 103.000000

mean 20.106796

std 2.845404

min 12.000000

25% 18.000000

50% 20.000000

75% 22.000000

max 25.000000

Name: PJ, dtype: float64

S.No False

Roll num False

Name False

MC False

IP False

PCOL False

PCOG False

PJ False

dtype: bool

S.No 0

Roll num 0

Name 0

MC 0

IP 0

PCOL 0

PCOG 0

PJ 0

dtype: int64

0 107

1 108

2 106

3 96

4 60

...

98 81

99 80

100 74

101 62

102 120

Name: TOT, Length: 103, dtype: int64

count 103.000000

mean 98.427184

std 14.708009

min 45.000000

25% 90.000000

50% 100.000000

75% 109.000000

max 123.000000

Name: TOT, dtype: float64

MC

0 21

1 21

2 22

3 18

4 13

.. ..

98 15

99 14

100 12

101 12

102 23

[103 rows x 1 columns]

0 107

1 108

2 106

3 96

4 60

...

98 81

99 80

100 74

101 62

102 120

Name: TOT, Length: 103, dtype: int64

Intercept value: 24.370791849368075

coefficient value: [3.87385092]

Actual predicted

0 26 98

60 100

2 106

51 118

71 ... [97.97395924684034, 82.4785555842146, 109.5955...

Explained\_variance: 0.8

Mean\_Squared\_Log\_Error: 0.0

R-squared: 0.7905

Mean Absolute Error(MAE): 3.48

Mean Squared Error (MSE): 3.48

Root Mean Squared Error (RMSE): 1.87

Intercept value: 24.020097944142762

coefficient value: [4.03571082]

Intercept value: 24.020097944142762

coefficient value: [4.03571082]

Actual predicted

0 26 98

60 100

2 106

51 118

71 ... [96.66289274631497, 108.77002521334367, 104.73...

Explained\_variance: 0.82

Mean\_Squared\_Log\_Error: 0.0

R-squared: 0.8242

Mean Absolute Error(MAE): 3.52

Mean Squared Error (MSE): 3.52

Root Mean Squared Error (RMSE): 1.88

0.8241908569767485

Intercept value: 5.248764175632431

coefficient value: [4.58703111]

Actual predicted

0 26 98

60 100

2 106

51 118

71 ... [96.98938644954929, 106.16344867694097, 101.57...

Explained\_variance: 0.91

Mean\_Squared\_Log\_Error: 0.0

R-squared: 0.9065

Mean Absolute Error(MAE): 2.82

Mean Squared Error (MSE): 2.82

Root Mean Squared Error (RMSE): 1.68

0.906509458015069

Intercept value: -6.6504472679619795

coefficient value: [5.12557689]

Actual predicted

0 26 98

60 100

2 106

51 118

71 ... [100.98666742635747, 100.98666742635747, 106.1...

Explained\_variance: 0.92

Mean\_Squared\_Log\_Error: 0.0

R-squared: 0.9069

Mean Absolute Error(MAE): 2.8

Mean Squared Error (MSE): 2.8

Root Mean Squared Error (RMSE): 1.67

0.9069098218271774

Intercept value: -0.9575158296469937

coefficient value: [4.92831989]

Actual predicted

0 26 98

60 100

2 106

51 118

71 ... [97.60888194986556, 102.53720183884118, 102.53...

Explained\_variance: 0.84

Mean\_Squared\_Log\_Error: 0.0

R-squared: 0.8226

Mean Absolute Error(MAE): 3.35

Mean Squared Error (MSE): 3.35

Root Mean Squared Error (RMSE): 1.83

0.82258383456067

Process finished with exit code 0