Import libraries, set directory and read the file

In [172…
```
import pandas as pd
import os
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [102…
```
#Lets check if we are in correct directory
os.getcwd()
```

Out[102]:
```
'C:\\Users\\tural\\OneDrive\\Desktop\\Study Materials\\Datasets'
```

In [103…
```
#As our file is in different directory lets change the directory to access the file
os.chdir("C:\\Users\\tural\\OneDrive\\Desktop\\Study Materials\\Datasets")
```

In [104…
```
#Now we can read our dataset to start our job
df = pd.read_csv("StudentsPerformance.csv")
```

Explore and Clean the Data

In [105…
```
df.head()
```

Out[105]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [106…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [107…
```
#Checking wether we have some NA values or not
df.isna().sum()
```

Out[107]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

In [108…
```
df.columns
```

Out[108]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [109…
```
#Renaming columns to be able to work easier
df.rename(columns = {'race/ethnicity':'race'}, inplace = True)
df.rename(columns = {'parental level of education':'parents_education'}, inplace = Tru
df.rename(columns = {'test preparation course':'prep_course'}, inplace = True)
df.columns = df.columns.str.replace(' ', '_')
```

In [110…
```
#parents education seems to be a bit messy so lets check the unique values and see if
pd.unique(df['parents_education'])
```

Out[110]:
```
array(["bachelor's degree", 'some college', "master's degree",
       "associate's degree", 'high school', 'some high school'],
      dtype=object)
```

In [111…
```
#Renaming some strings inside of data frame to look more organized
df = df.replace('some ','', regex=True)
df = df.replace(' degree', '', regex=True)
```

Descriptive Statistics

In [ ]:
```
# Before starting the descriptive statistics lets create a column which will be the av
df['avg_score'] = np.mean(df, axis = 1)
df = df.round(1)
```

In [150…
```
df.head()
```

Out[150]:

	gender	race	parents_education	lunch	prep_course	math_score	reading_score	writing_s
0	female	group B	bachelor's	standard	none	72	72	
1	female	group C	college	standard	completed	69	90	
2	female	group B	master's	standard	none	90	95	
3	male	group A	associate's	free/reduced	none	47	57	
4	male	group C	college	standard	none	76	78	

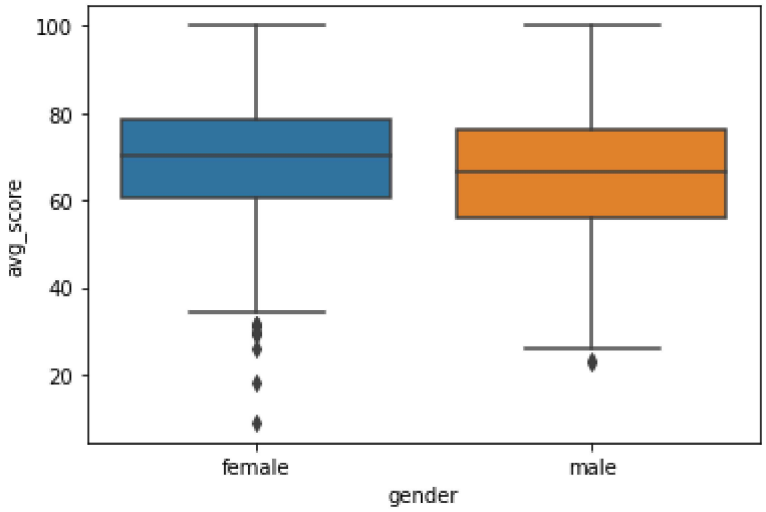
In [151…
```
df.describe()
```

Out[151]:

	math_score	reading_score	writing_score	avg_score
count	1000.00000	1000.000000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000	67.769800
std	15.16308	14.600192	15.195657	14.257197
min	0.00000	17.000000	10.000000	9.000000
25%	57.00000	59.000000	57.750000	58.300000
50%	66.00000	70.000000	69.000000	68.300000
75%	77.00000	79.000000	79.000000	77.700000
max	100.00000	100.000000	100.000000	100.000000

In [154…
```
# Lets create a box plot for avg score based on gender
sns.boxplot(x = 'gender', y = 'avg_score', data = df)
```

Out[154]:
```
<AxesSubplot:xlabel='gender', ylabel='avg_score'>
```



In [156…
```
#Lets go a bit deeper to see the differences in each exam
df.groupby('gender').describe()['math_score']
```

Out[156]:

gender	count	mean	std	min	25%	50%	75%	max
female	518.0	63.633205	15.491453	0.0	54.0	65.0	74.0	100.0
male	482.0	68.728216	14.356277	27.0	59.0	69.0	79.0	100.0

In [134…
```
df.groupby('gender').describe()['writing_score']
```

Out[134]:

gender	count	mean	std	min	25%	50%	75%	max
female	518.0	72.467181	14.844842	10.0	64.0	74.0	82.00	100.0
male	482.0	63.311203	14.113832	15.0	53.0	64.0	73.75	100.0

In [157…
```
df.groupby('gender').describe()['reading_score']
```

Out[157]:

gender	count	mean	std	min	25%	50%	75%	max
female	518.0	72.608108	14.378245	17.0	63.25	73.0	83.0	100.0
male	482.0	65.473029	13.931832	23.0	56.00	66.0	75.0	100.0

In [199…
```
df.groupby('race').describe()['avg_score']
```

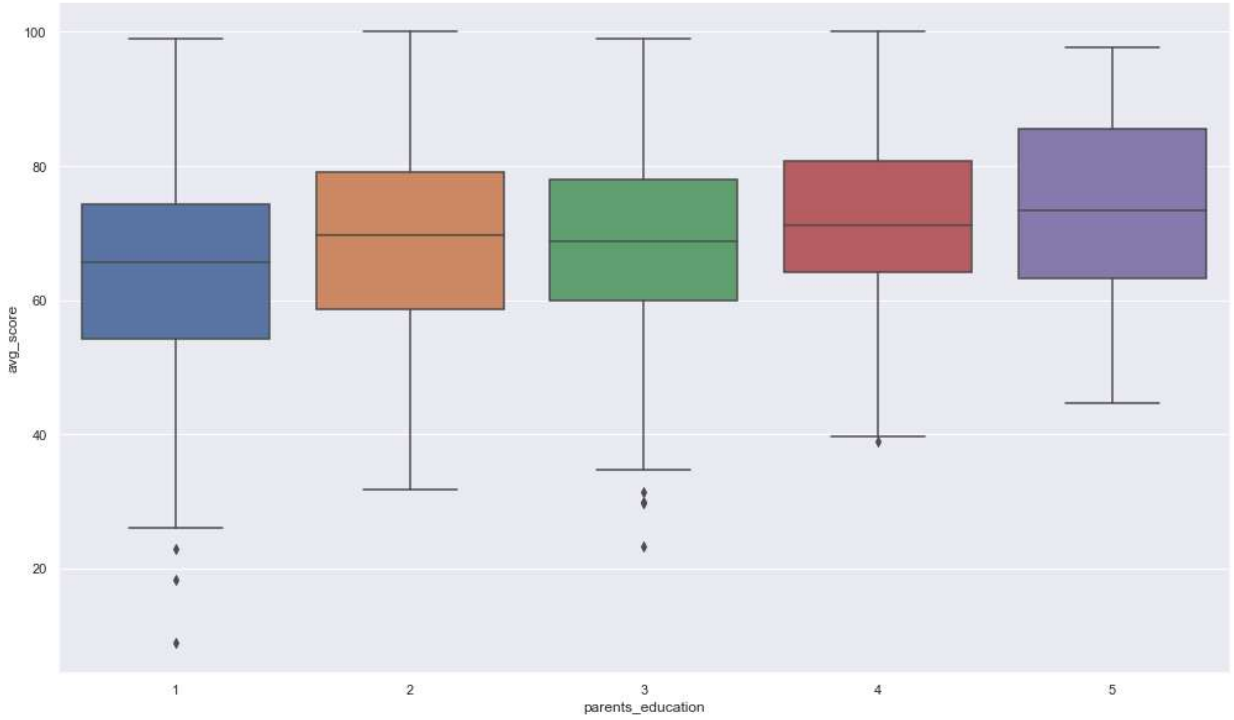
Out[199]:

race	count	mean	std	min	25%	50%	75%	max
group A	89.0	62.988764	14.448902	23.3	52.0	61.3	73.000	96.3
group B	190.0	65.470000	14.733527	18.3	56.7	65.0	76.825	96.7
group C	319.0	67.130721	13.871086	9.0	57.7	68.3	77.000	98.7
group D	262.0	69.179389	13.250497	31.0	60.3	70.0	78.600	99.0
group E	140.0	72.748571	14.566605	26.0	64.7	73.5	82.400	100.0

In [192…
```
#Lets turn our categorical parent's education level to numeric.
df['parents_education'].replace(['high school', "associate's", 'college', "bachelor's'
```

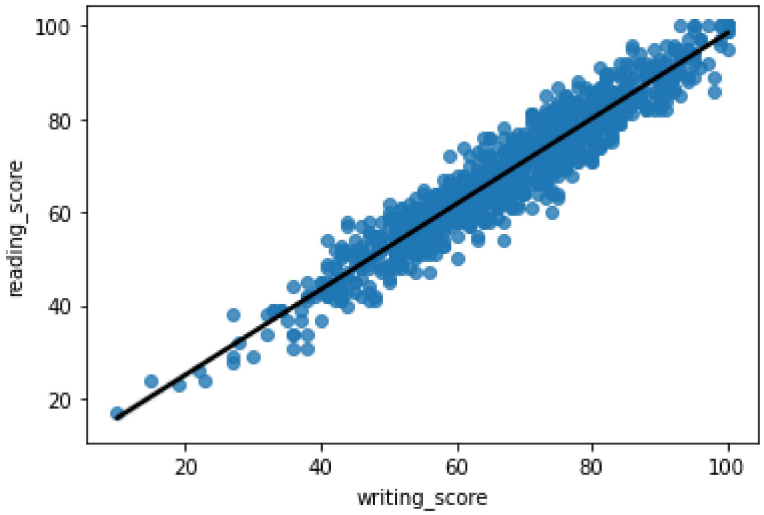
In [195…
```
sns.boxplot(x = df['parents_education'], y = df['avg_score'])
```

Out[195]:
```
<AxesSubplot:xlabel='parents_education', ylabel='avg_score'>
```



In [171…
```
#Lets create a scatter plot with regression line to get sense if writing and reading i
sns.regplot(x = df['writing_score'], y = df['reading_score'], data = df, line_kws={"co
```

Out[171]:
```
<AxesSubplot:xlabel='writing_score', ylabel='reading_score'>
```



In [184…
```
#Lets see the peaarson coefficient and p statistic value to see if the correlation is
pearson_coef, p_value = stats.pearsonr(df['writing_score'], df['reading_score'])
print(pearson_coef, p_value)
```

```
0.9545980771462478 0.0
```

Linear Model Development

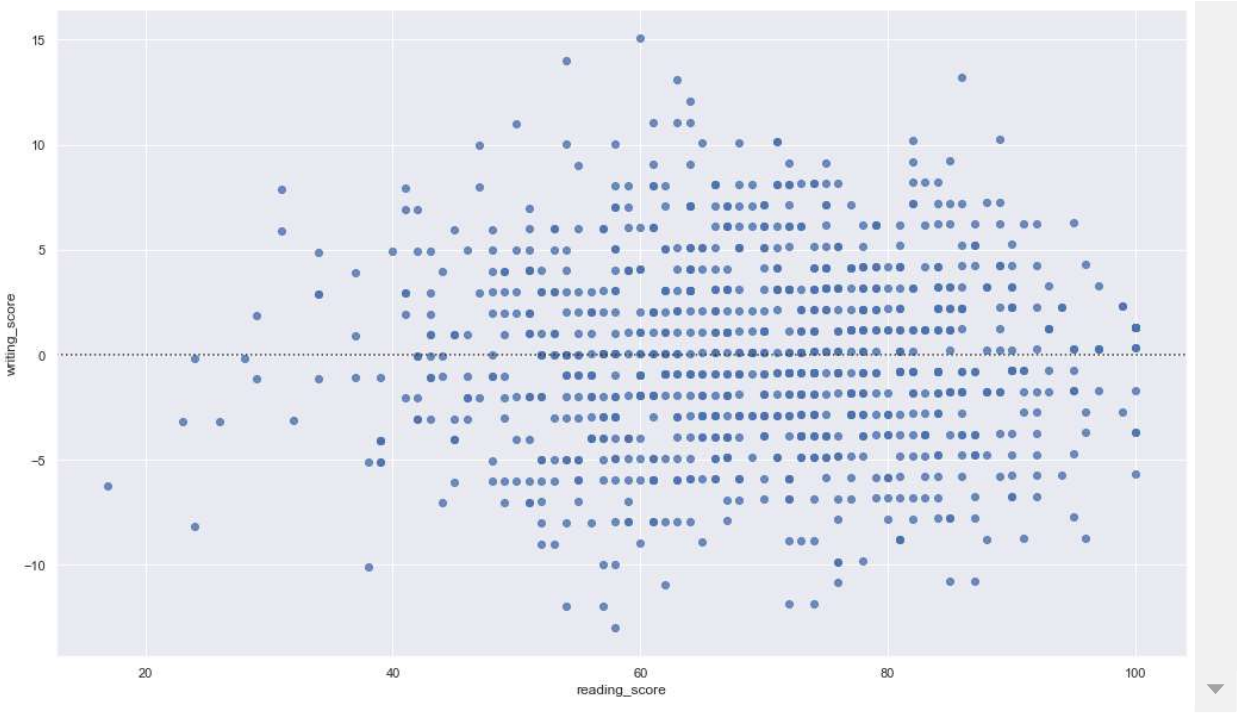
In [205…
```
#Lets develop our model and then show our equation for predicting the writing score
X = df[['reading_score']]
Y = df['writing_score']
lm.fit(X, Y)
Yhat = lm.predict(X)
print ("writing_score = ", "reading_score", "*", lm.coef_, "+", lm.intercept_)
```

```
writing_score =  reading_score * [0.99353111] + -0.6675536409329226
```

In [202…
```
#As the residuals are evenly distributed along the mean axes we can conclude that line
sns.residplot(df['reading_score'], df['writing_score'])
```

```
C:\Users\tural\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

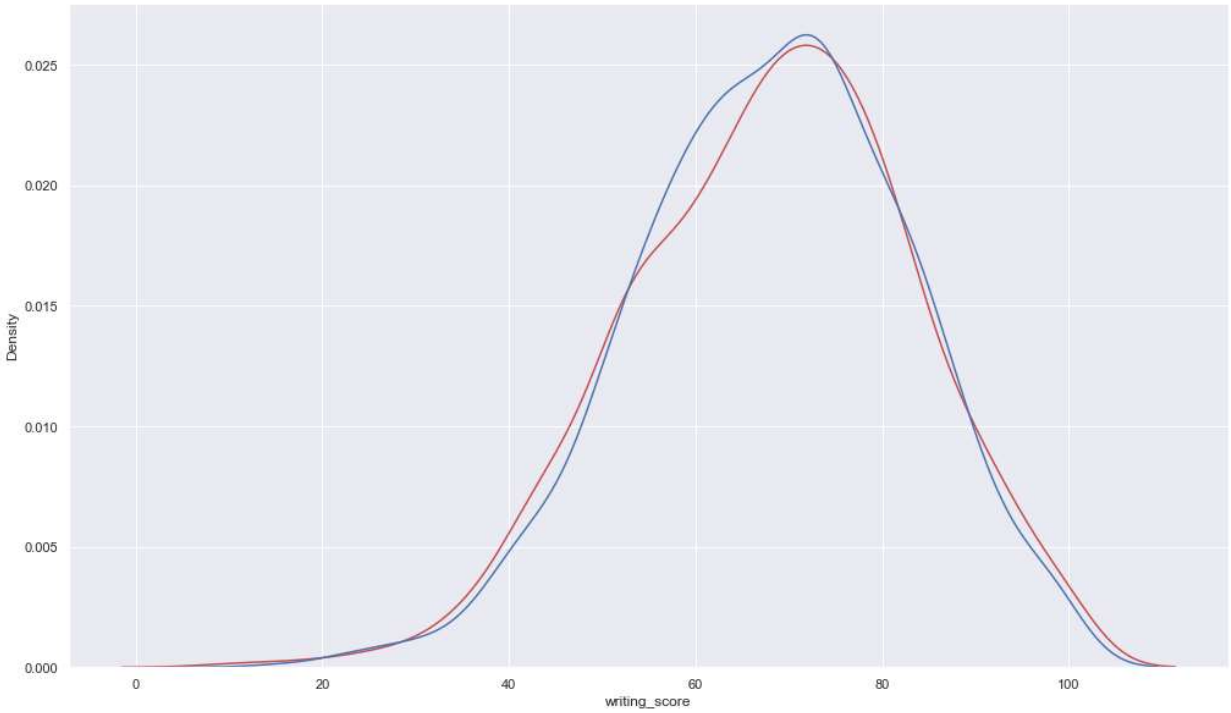
Out[202]:
```
<AxesSubplot:xlabel='reading_score', ylabel='writing_score'>
```



In [207…
```
# Finally lets draw a distribution plot in order to compare the actual writing score w
ax1 = sns.distplot(df['writing_score'], hist = False, color = 'r', label = "Actual Wri
sns.distplot(Yhat, hist= False, color = 'b', label = "Fitted Value", ax = ax1)
```

```
C:\Users\tural\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\tural\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[207]:
```
<AxesSubplot:xlabel='writing_score', ylabel='Density'>
```



As we can see our model is good enough to take it into consideration to predict the writing score