BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Electrical and Computer Engineering Department

MACHINE LEARNING AND DATA SCIENCE

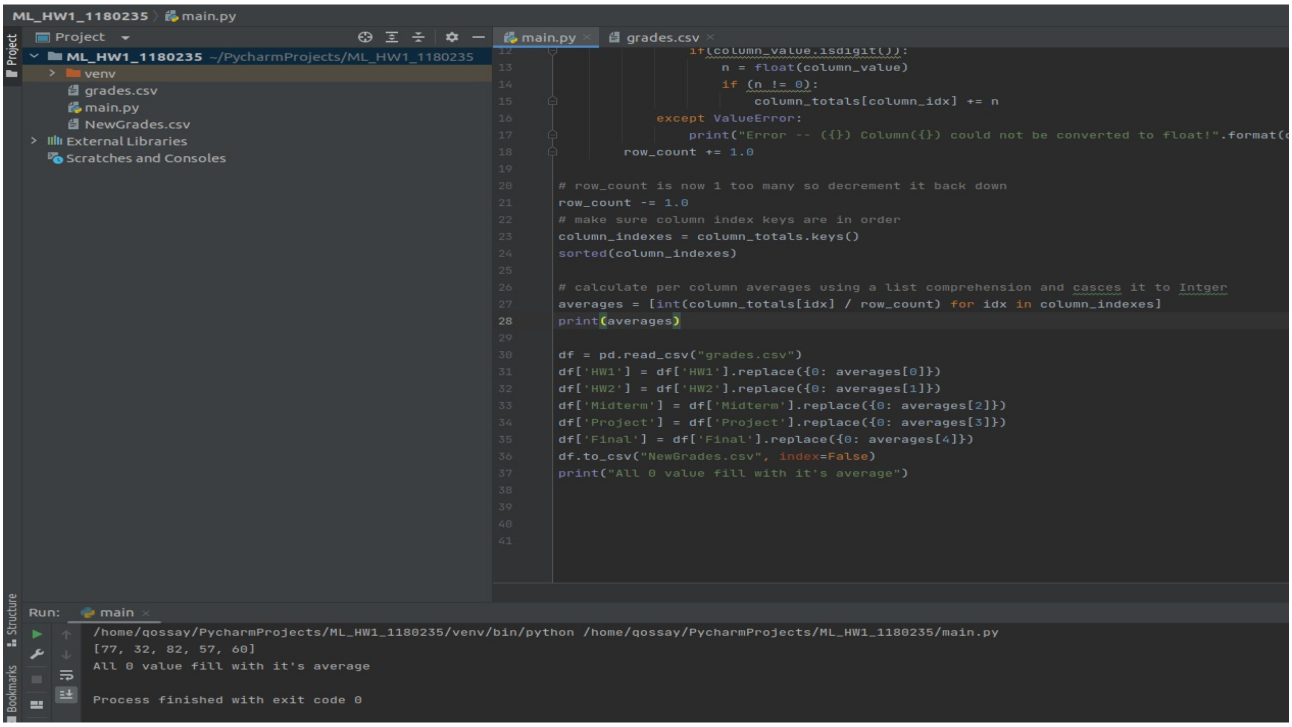
Report of
“ HomeWork 1”

- **Prepared by:** Qossay j.e. ZeinEddin 1180235
- **Instructor:** Dr . Yazan Abu Farha
- **Section No.**: 1
- **Date**: 10-12-2022

➢ Procedure and Discussion

1- Some values are missing (indicated by 0 value). Address all the missing values by using the average of the available values for the corresponding variable.

In this part I write a python code that calculate the average of all columns alone and fill all 0 values of it's column average values .[Appendex 1]



.

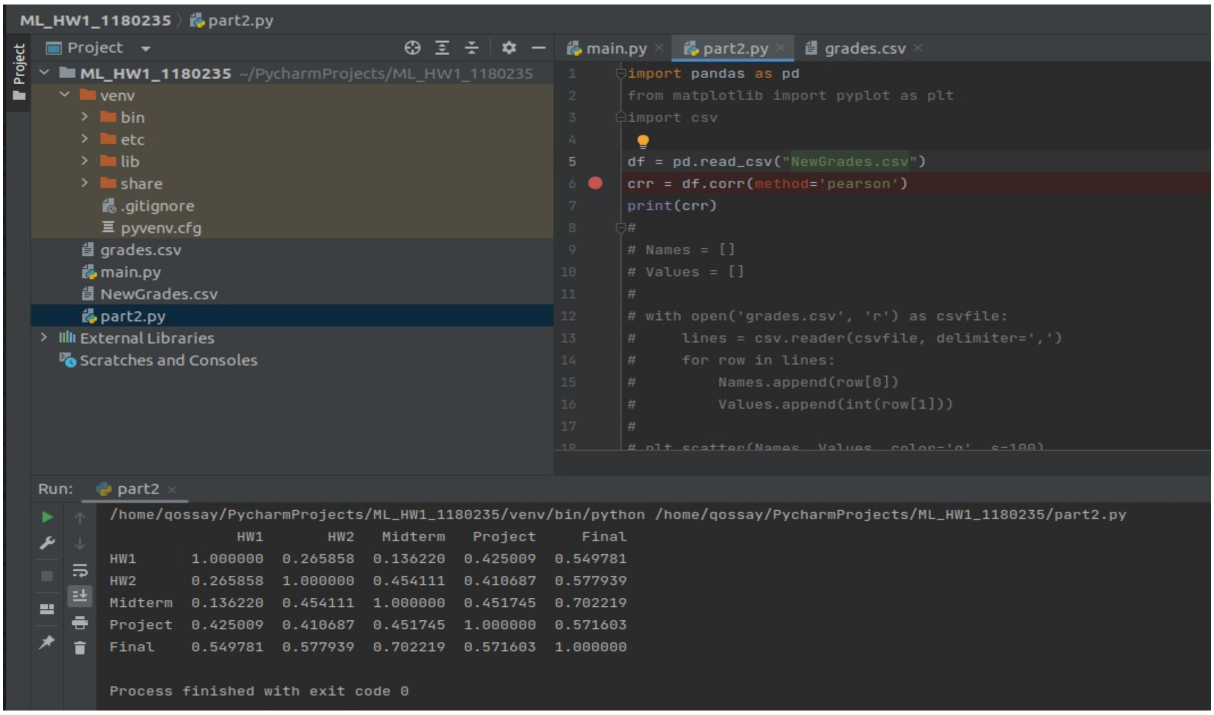
2- Using data science techniques that we discussed in the course, examine which of the input variables would be a good predictor for the final exam.
Method of correlation:

Correlation Coefficient Formula

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

We may get the strength of the relationship between an independent variable (x) and a dependent variable (y) from the following equation, and we can use the result to determine how strong the relationship is ,which is r varies between 0 which indicates no relationship to 1 which is identical strong relationship.

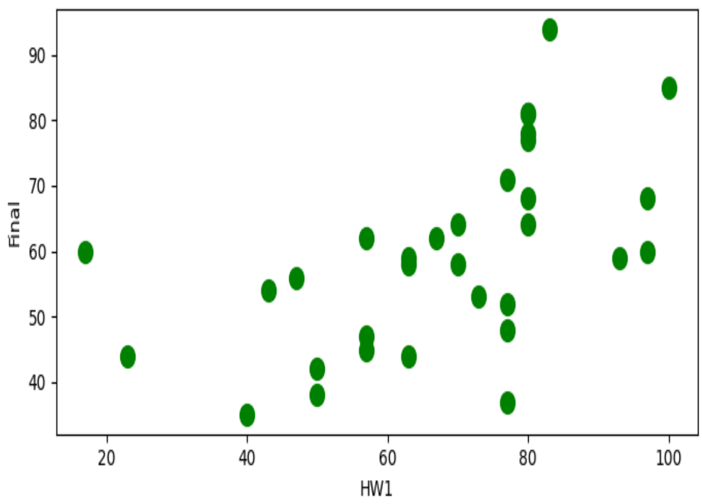
by using a pythone lib “pandas” and Method of correlation: “pearson” : standard correlation coefficient we get :



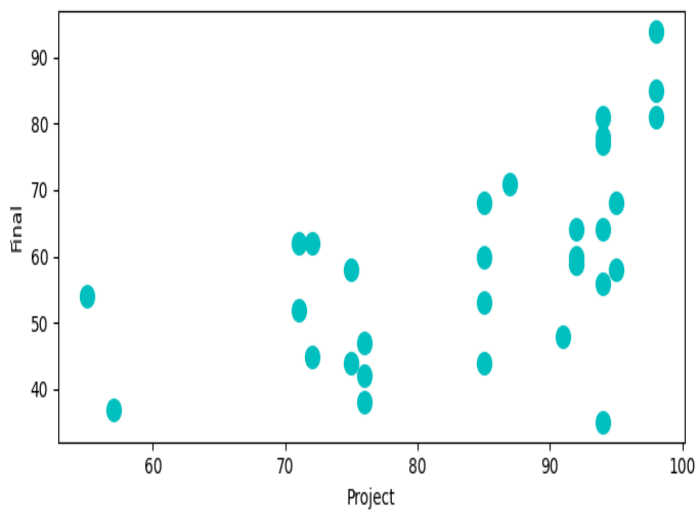
we see from the table (final & mid) have the highest value

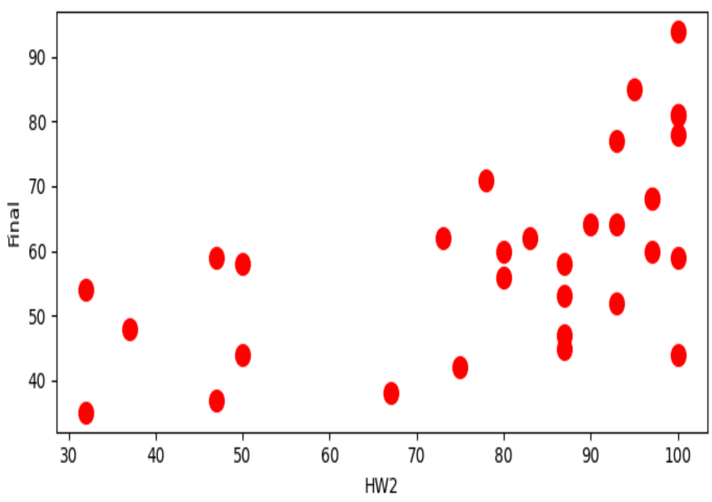
	HW1	HW2	Midterm	Project	Final
HW1	1.000000	0.265858	0.136220	0.425009	0.549781
HW2	0.265858	1.000000	0.454111	0.410687	0.577939
Midterm	0.136220	0.454111	1.000000	0.451745	0.702219
Project	0.425009	0.410687	0.451745	1.000000	0.571603
Final	0.549781	0.577939	0.702219	0.571603	1.000000

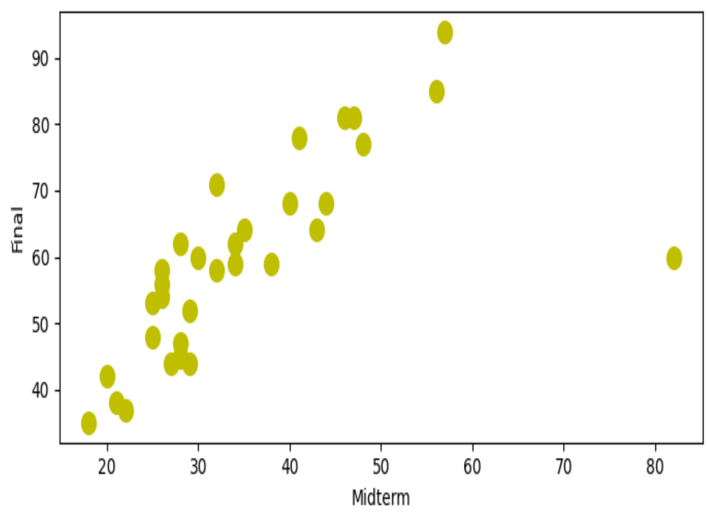
And if we plot the relationship as below:



5







As seen in the above charts, the Mid's grades produce the best linear curve that can be built with the least square errors, providing another another method for observing the best variable to predict the final mark.

[Appendex 2]

3- Implement the closed form solution of linear regression and use it to learn a linear model to predict the final exam from the variable you selected in part 2.

Finding a linear regression equation that gives us the least square error—that is, one that minimizes the square error , the equation to use is: F(x) = y = W0 + w1x

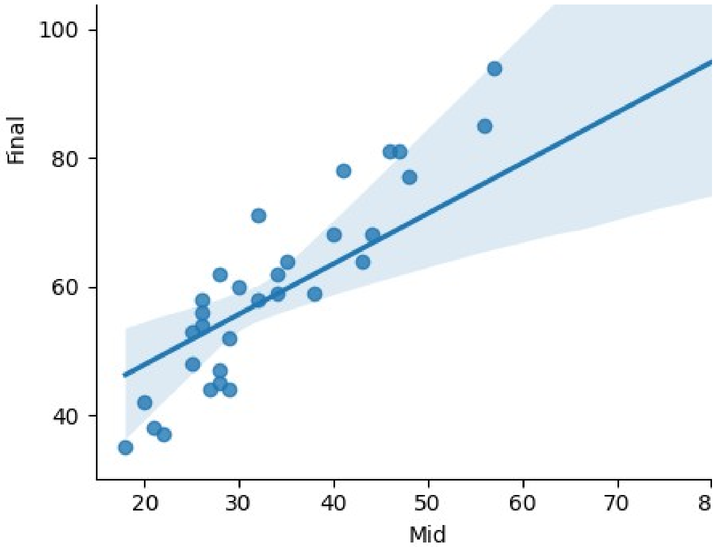
X: Mid Value , then w find best measure to predict final exam value
To find W0 , W1:

$$w_0 = \frac{\sum_{i=1}^{n} y_i}{n} - w_1 \ \frac{\sum_{i=1}^{n} x_i}{n}$$

$$w_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i}{n}}$$

W1 = 1.36
W0 = 14.62
So Now the F(x)is: F(x) = y = 14.62 + 1.36x

and if we plot it:



[Appendex 3]

4- Repeat part 3 but now by implementing the gradient descent algorithm.

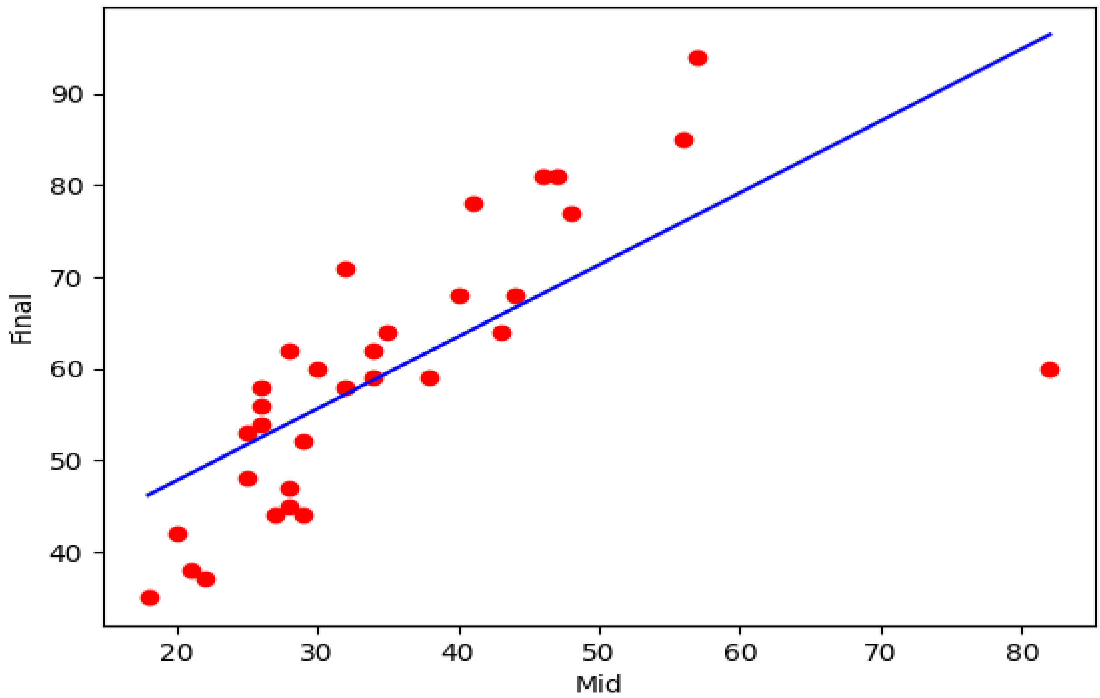
To find gradient descent we should minimize the errors in every iteration to get the least squares error between predicted and actual value .
in the data set we have n = 32
we will use :

$$error = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w_1 x_i + w_0))^2$$

$$\frac{d(error)}{dw1} = \frac{-2}{n}\sum_{i=1}^{n} x_i(y_i - (w_1 x_i + w_0))$$

$$\frac{d(error)}{dw0} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - (w_1 x_i + w_0))$$

to implements the above equations in pythone:



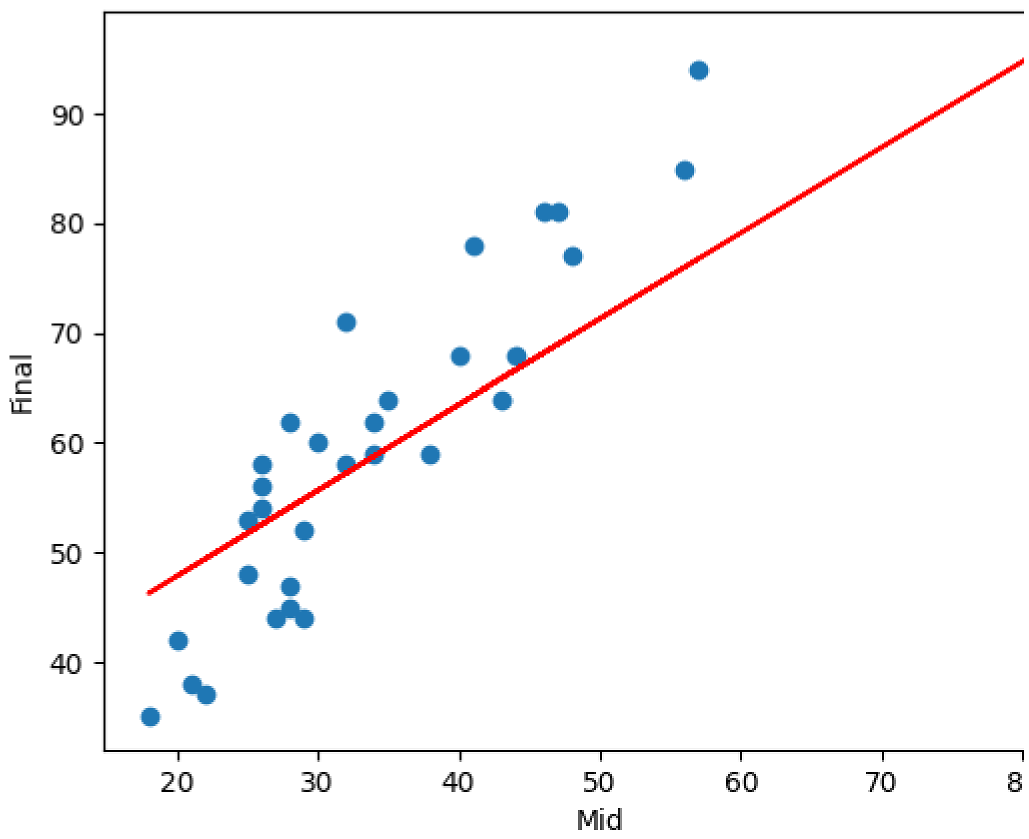
w1: 1.2856412532816974
w0: 14.06040405058389
[Appendex 4]

5- Repeat part 3 but now using the linear regression implementation of scikit-learn python library.

My using python code in [Appendex5] we get :

W1: [1.34802079]
W0: [14.55890067]

they are   close to the previous results
linear regression plot:



[Appendex 5]

6- Compare the answers from part 3, 4, and 5. In each case compute the error of the learned models.

error manual regression: 33.1995202
error gradient decent: 33.241701
error scikit learn: 33.412789
AS we can see the first one is better

➢ Appendices

Appendex 1 :

```
import csv
import pandas as pd
from collections import Counter

column_totals = Counter()
with open("grades.csv", "r") as f:
    reader = csv.reader(f)
    row_count = 0.0
    for row in reader:
        for column_idx, column_value in enumerate(row):
            try:
                if(column_value.isdigit()):
                    n = float(column_value)
                    if (n != 0):
                        column_totals[column_idx] += n
            except ValueError:
                print("Error -- ({}) Column({}) could not be converted to
float!".format(column_value, column_idx))
        row_count += 1.0

# row_count is now 1 too many so decrement it back down
row_count -= 1.0
# make sure column index keys are in order
column_indexes = column_totals.keys()
sorted(column_indexes)

# calculate per column averages using a list comprehension and casces it to Intger
averages = [int(column_totals[idx] / row_count) for idx in column_indexes]
print(averages)

df = pd.read_csv("grades.csv")
df['HW1'] = df['HW1'].replace({0: averages[0]})
df['HW2'] = df['HW2'].replace({0: averages[1]})
df['Midterm'] = df['Midterm'].replace({0: averages[2]})
df['Project'] = df['Project'].replace({0: averages[3]})
df['Final'] = df['Final'].replace({0: averages[4]})
df.to_csv("NewGrades.csv", index=False)
print("All 0 value fill with it's average")
```

Appendex 2 :

```python
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv("NewGrades.csv")
crr = df.corr(method='pearson')
print(crr)

Names = []
Values = []

plt.rcParams["figure.figsize"] = [7.00, 3.50]
plt.rcParams["figure.autolayout"] = True
#columns = ["HW1", "Final"]
#columns = ["HW2", "Final"]
#columns = ["Midterm", "Final"]
columns = ["Project", "Final"]

df = pd.read_csv("NewGrades.csv", usecols=columns)
print("Contents in csv file:", df)

#plt.xlabel('HW1')
#plt.xlabel('HW2')
#plt.xlabel('Midterm')
plt.xlabel('Project')

plt.ylabel('Final')

#plt.scatter(df.HW1, df.Final, color='g', s=100)
#plt.scatter(df.HW2, df.Final, color='r', s=100)
#plt.scatter(df.Midterm, df.Final, color='y', s=100)
plt.scatter(df.Project, df.Final, color='c', s=100)

plt.show()
```

Appendex 3 :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression



plt.rcParams["figure.figsize"] = [7.00, 3.50]
plt.rcParams["figure.autolayout"] = True
columns = ["Midterm", "Final"]

df = pd.read_csv("NewGrades.csv", usecols=columns)
print("Contents in csv file:", df)

x = df.Midterm
y = df.Final
data = pd.DataFrame([x, y]).T
data.columns = ['x', 'y']

sns.lmplot(x="x", y="y", data=data, order=1 )
plt.ylabel('Final')
plt.xlabel('Mid')
plt.show()
```

Appendex 4:

```python
import numpy as np
import matplotlib.pyplot as plt
import csv
import pandas as pd

columns = ["Midterm", "Final"]
df = pd.read_csv("NewGrades.csv", usecols=columns)
print("Contents in csv file:", df)
x = df.Midterm
y = df.Final

w1 = 0.1
w0 = 0.01
costs = []
weights = []
prev_error = None
learning_rate = 0.0001
for i in range(1000000):
    predict = (w1 * x) + w0
    err = np.sum((y - predict) ** 2) / 32
    if prev_error and abs(prev_error - err) <= 0.0000001:
        break
    prev_error = err
    costs.append(err)
    weights.append(w1)
    w1_1 = -(2 / 32) * sum(x * (y - predict))
    w0_0 = -(2 / 32) * sum(y - predict)
    w1 = w1 - (learning_rate * w1_1)
    w0 = w0 - (learning_rate * w0_0)


f = w1 * x + w0
print(w1)
print(w0)
print(f)
plt.scatter(x , y , marker='o', color='red')
plt.plot([min(x), max(x)], [min(f), max(f)])
plt.xlabel("Mid")
plt.ylabel("Final")
plt.show()
```

Appendex 5 :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pandas as pd




columns = ["Midterm", "Final"]
df = pd.read_csv("NewGrades.csv", usecols=columns)
x = df.Midterm
y = df.Final
x = np.array(x).reshape(-1,1)
y = np.array(y).reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.1)
reg = LinearRegression()
reg.fit(X_train, y_train)
print("W00: ", reg.intercept_)
print("W11: ", reg.coef_)


#
# Themodel = LinearRegression()
# x = np.expand_dims(x, 1)
# Themodel.fit(x,y)
# plt.scatter(x, y)
# f = Themodel.coef_*x + Themodel.intercept_
# plt.plot(x, f, 'r')
# plt.xlabel("Mid")
# plt.ylabel("Final")
# plt.show()

columns = ["Midterm", "Final"]
df = pd.read_csv("NewGrades.csv", usecols=columns)
x = df.Midterm
y = df.Final
x = np.array(x)
y = np.array(y)




Themodel = LinearRegression()
x = np.expand_dims(x, 1)
Themodel.fit(x,y)
print("w1: ", Themodel.coef_)
print("w0: ", Themodel.intercept_)
plt.scatter(x, y)
x = x
f = Themodel.coef_*x + Themodel.intercept_
plt.plot(x, f, 'r')
plt.xlabel("Mid")
plt.ylabel("Final")
plt.show()
```

➢ Reference

https://realpython.com/python-csv/ [1]
https://www.geeksforgeeks.org/python-pandas-dataframe-corr/ [2]
https://towardsdatascience.com/closed-form-solution-to-linear-regression-e1fe14c1cbef [3]
https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21 [4]
https://stackabuse.com/linear-regression-in-python-with-scikit-learn/ [5]