Final Project

November 15, 2022

1 Final Project

1.1 EE3326 Fall 2022

This Final Project has four sections.

- Section I - General project information and document generation using Jupyter notebook. You may start here or just go to section II to start your project.
- Section II - Simple Linear Regression (SLR) analysis. Please follow the directions and complete all parts.
- Section III - Simple data display using f'strings.
- Section IV - Data generation for different R Values.

2 Section I - Document Preparation using Jupyter Notebook

During this project, you will learn how to write a scientific paper using Jupyter Notebook (or JupyterLab). You need to complete all these parts to get full credit. You need to submit your project as a .zip file. First, create a folder and name it "YourNumber_LastName_Final_Project." Create a document file `Notebook` and put all your figures and codes here. After completing all your work, please zip your folder and submit it to OneDrive.

This section can be seen as creating a document template for your project. You need to write your project paper using `markdown` and/or `code` in Jupyter notebook. After completing your project you may need to edit your final paper using MS Word, Adobe Acrobat, or LaTeX.

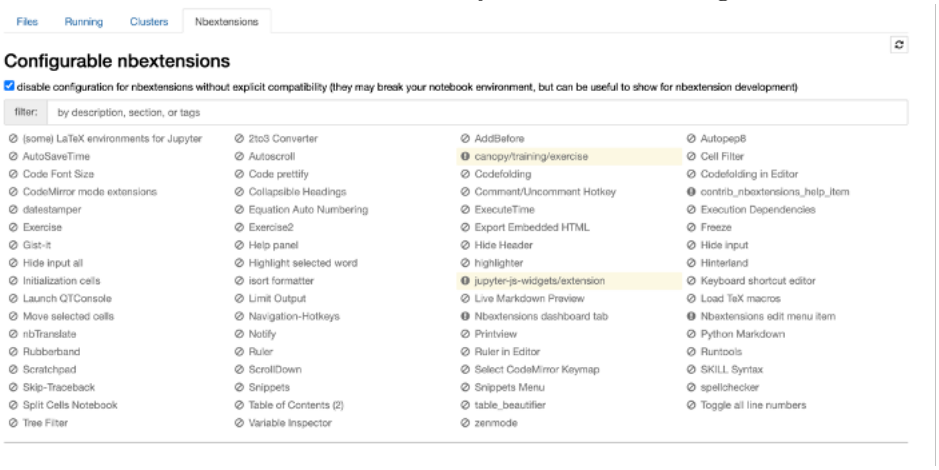
2.0.1 Part I – Writing a paper using Jupyter Notebook:

As described above, make a project folder and create your Final Project Jupyter Notebook in this folder. After notebook file is created, you may need to enable some nbextensions you want to work with such as "Equation auto numbering", " Exercise", Table of Contents", "Load TeX macros" etc. to make your final editing much easier. Complete sections II, III and IV using `markdown` and `code`. You can work anyway you want. When you write your paper you need to make sure you have followings: * Minimum one Figure, one Table and One Equation. * Minimum 6 and maximum 10 pages.

2.0.2 Part II - Document generation and Conversion

During this project you will need to use:

- Jupyter Notebook (suggested)
- Enable "nbextensions" – You may need to Google it for further information



- Pandoc for document conversion (https://pandoc.org/)

After completing alls sections and ready for your document generation use;

- `Export Notebook to Markdown`
- `Export Notebook to PDF`
- `Export Notebook to LaTeX`
- `Export Notebook to ReStructured Text`

If you know `LaTeX` or how to edit pdf using `Adobe Acrobat` please finilize you final report and create a pdf file.

If you want to use `MS word` for editing for your documents please use Pandoc to convert your `markdown` to `MS Word`. Please see `https://pandoc.org/demos.html` for more help.

[ ]:

3 Section II - Simple Linear Regression Formulas:

Please read the "SLR_main.pdf" and/or "SLR_Supplement.pdf" for this project. These are such a good resources regarding `Simple Linear Regression`. While you do that please create following functions to solve following SLR Examples. I solve these using `sklearn.linear_model -> LinearRegression` package.

> Main idea here is to fit a data in y = mx + b form. We will use y = b1 x + b0 here.
>
> Another good website using classes "https://dziganto.github.io/data%20science/linear%20regression/machin
> Regression-101-Basics/" and "https://dziganto.github.io/data%20science/linear%20regression/machine%20l
> Regression-101-Metrics/"

```
import numpy as np
import matplotlib.pyplot as plt

X = [.........] # Your data X-values as a list
Y = [.........] # Your data Y-values as a list
```

```
x =   # convert above X-data to a numpy array
Y =   # convert above Y-data to a numpy array

# Write a function to find b1

def b1(x,y):
''' Please use Formula in Page 7-5 --- SLR_main.pdf --- '''
    return b1

# Write a function to find b0

def b0(x,y):
''' Please use Formula in Page 7-6 --- SLR_main.pdf --- '''
    return b0

# Write a function to find R

def R(x,y):
''' Please use Formula in Page 7-13 --- SLR_main.pdf --- '''
    return R

# Write a function to find SSE

def SSE(y,yh):
''' Please use Formula in Page 7-3 --- SLR_main.pdf --- '''
    return SSE

# Write a function to find SSR

def SSR(y,yh):
''' Please use Formula in Page 7-3 --- SLR_main.pdf --- '''
    return SSR

# Write a function to find SST

def SST(y,yh):
''' Please use Formula in Page 7-3 --- SLR_main.pdf --- '''
    return SSR

# Check if SST = SSR + SSE as shown on page 7-13

# Write a function to find MSE

def MSE(y,yh):
''' Please use Formula in Page 7-5 --- SLR_main.pdf --- '''
    return SSR
```

```
# Check if MSE = SSE (n-2) as shown on page 7-5
```

After you complete your functions please check you results using the below examples:

Please check your results using `sklearn.linear_model LinearRegression`

3.0.1 Part I - Example 1 (page 7-1)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[6]:
```
# install scikit-learn
#pip install -U scikit-learn
```

[7]:
```
X=[70,70,70,80,80,80,90,90,90,100,100,100]
Y=[2.3,2.6,2.1,2.5,2.9,2.4,3.0,3.1,2.8,3.3,3.5,3.0]

x= np.array(X)
y=np.array(Y)
```

[8]:
```
# Example 1 using 'sklearn.linear_model LinearRegression'

from sklearn.linear_model import LinearRegression

XX=np.array(X).reshape(-1,1) # make it column array
YY=np.array(Y) # row array
```

[ ]:
```
model=LinearRegression()  # create a model
model.fit(XX,YY) # fit data with this model
b0=model.intercept_ # find bo
b1=model.coef_ # find b1
print(b0,b1)
R2=model.score(XX,YY)  # find R^2
R=np.sqrt(R2) # Find R
(R2,R)
```

[3]:
```
# Example 1 using 'functions from Project Part I'


####### Your Functions Goes Here ########
import numpy as np
import matplotlib.pyplot as plt

X = [.........] # Your data X-values as a list
Y = [.........] # Your data Y-values as a list

x =   # convert above X-data to a numpy array
```

```
Y =   # convert above Y-data to a numpy array

# Write a function to find b1

def b1(x,y):
''' Please use Formula in Page 7-5 --- SLR_main.pdf --- '''
    return b1

# Write a function to find b0

def b0(x,y):
''' Please use Formula in Page 7-6 --- SLR_main.pdf --- '''
    return b0

# Write a function to find R

def R(x,y):
''' Please use Formula in Page 7-13 --- SLR_main.pdf --- '''
    return R

# Write a function to find SSE

def SSE(y,yh):
''' Please use Formula in Page 7-3 --- SLR_main.pdf --- '''
    return SSE

# Write a function to find SSR

def SSR(y,yh):
''' Please use Formula in Page 7-3 --- SLR_main.pdf --- '''
    return SSR

# Write a function to find SST

def SST(y,yh):
''' Please use Formula in Page 7-3 --- SLR_main.pdf --- '''
    return SSR

# Check if SST = SSR + SSE as shown on page 7-13

# Write a function to find MSE

def MSE(y,yh):
''' Please use Formula in Page 7-5 --- SLR_main.pdf --- '''
    return SSR

######
```

```
X=[70,70,70,80,80,80,90,90,90,100,100,100]
Y=[2.3,2.6,2.1,2.5,2.9,2.4,3.0,3.1,2.8,3.3,3.5,3.0]

x= np.array(X)
y=np.array(Y)
```

[12]:
```
bb1=b1(x,y)
bb1
```

[12]: 0.03166666666666667

[13]:
```
bb0=b0(x,y)
bb0
```

[13]: 0.09999999999999964

[4]:
```
yh=bb0+bb1*x
yl=np.average(y)
```

[5]:
```
SSE1=SSE(y,yh)
```

[6]:
```
SSR1=SSR(y,yh)
```

[7]:
```
SST1=SST(y,yl)
```

[8]:
```
SSR1+SSE1
```

[8]: 1.949166666666667

[9]:
```
SST1
```

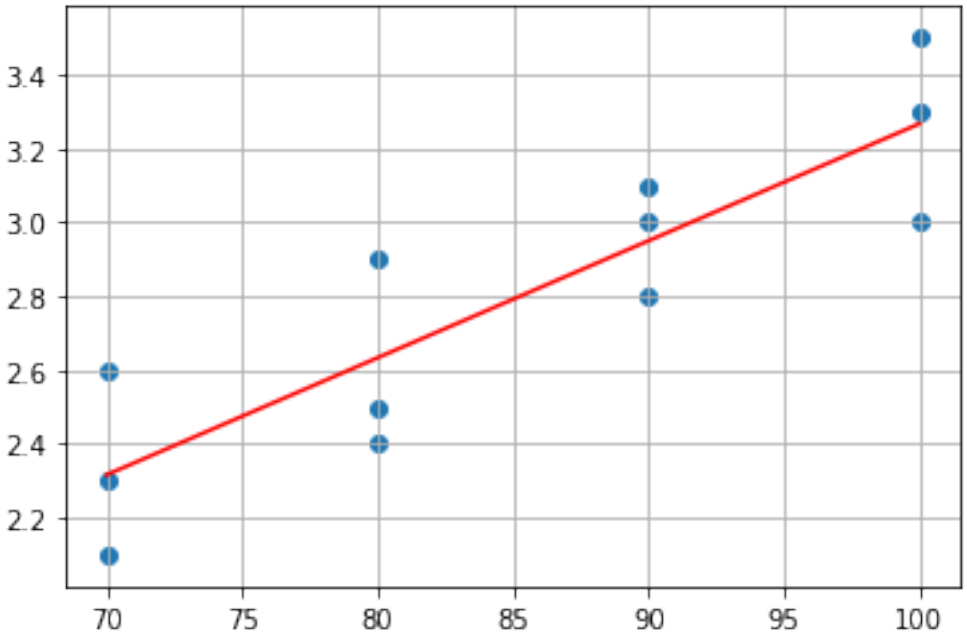
[9]: 1.9491666666666665

[10]:
```
R1=R(x,y)
R1
```

[10]: 0.8784630365252997

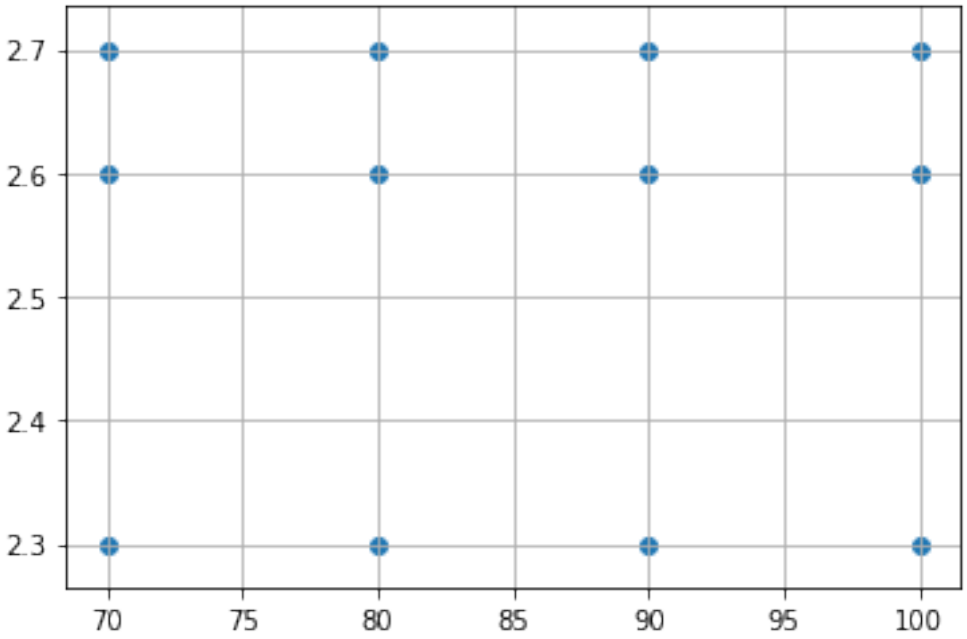
[16]:
```
## Print data and fitted model
# we will plot (x,y) data and fitted model Ymod = b0 + x*b1

Ymod=bb0+bb1*x

plt.scatter(x,y) # data
```

```
plt.plot(x,Ymod,'red')

plt.grid()
```



3.0.2 Part I - Example 2

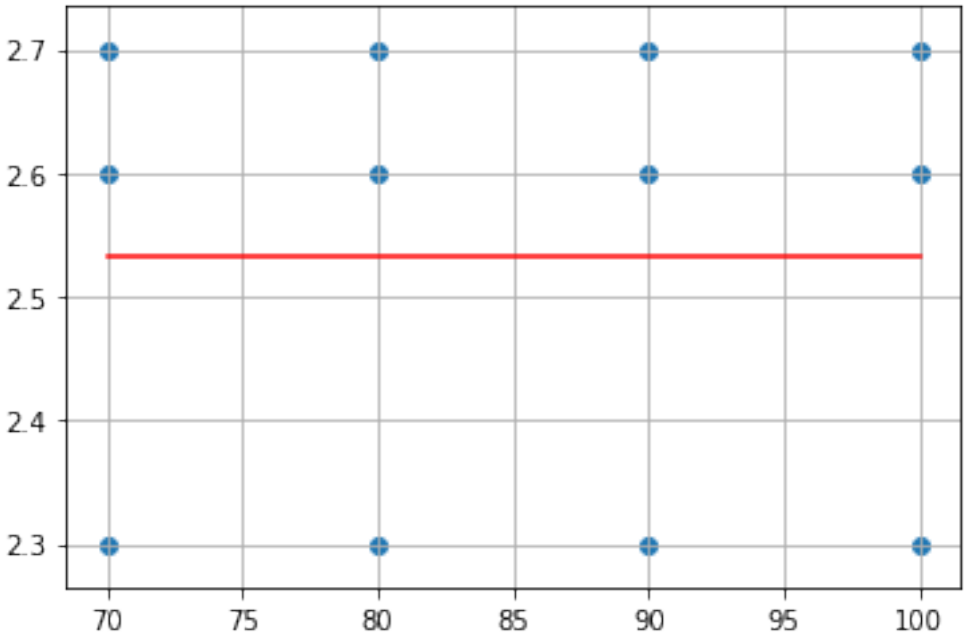
[19]:
```
from sklearn.linear_model import LinearRegression

X=np.array([70,70,70,80,80,80,90,90,90,100,100,100]).reshape(-1,1)
Y=np.array([2.3,2.6,2.7,2.3,2.6,2.7,2.3,2.6,2.7,2.3,2.6,2.7])
plt.scatter(X,Y)
plt.grid()
```



[23]:
```
model=LinearRegression()
model.fit(X,Y)
b0=model.intercept_
b1=model.coef_
print(b0)
print(b1)
print(b0,b1)
R2=model.score(X,Y)  # find R^2
R=np.sqrt(R2) # Find R
print(R2)
print(R)
```

```
2.5333333333333337
[1.08279195e-18]
2.5333333333333337 [1.08279195e-18]
0.0
0.0
```

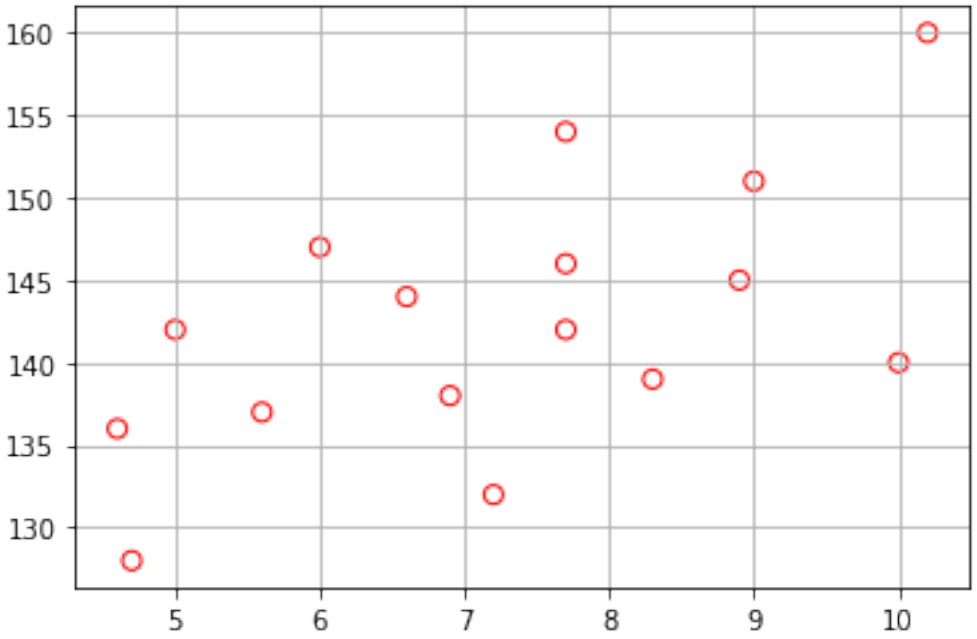
[21]:
```
y=b0+b1*X
plt.scatter(X,Y),plt.plot(X,y,'red')
plt.grid()
```



3.1 PLEASE TRY YOUR FUNCTIONS TO FIND SLR VALUES

3.1.1 Part I - Example 3 (page 7-10)

[33]:
```
x=[4.6,4.7,5,5.6,6,6.6,6.9,7.2,7.7,7.7,7.7,8.3,8.9,9,10,10.2]
y=[136,128,142,137,147,144,138,132,142,146,154,139,145,151,140,160]
plt.scatter(x,y,s=50,color='r',marker="o",facecolors='none')
plt.grid()
```



[34]:
```
X=np.array(x).reshape(-1,1)
Y=np.array(y)
model=LinearRegression()
model.fit(X,Y)
b0=model.intercept_
b1=model.coef_
print(b0)
print(b1)
ym=b0+b1*X
R2=model.score(X,Y)  # find R^2
R=np.sqrt(R2) # Find R
print(R2)
print(R)
```

```
123.10759468517759
[2.6811239]
0.3446818691176132
0.587096132773512
```

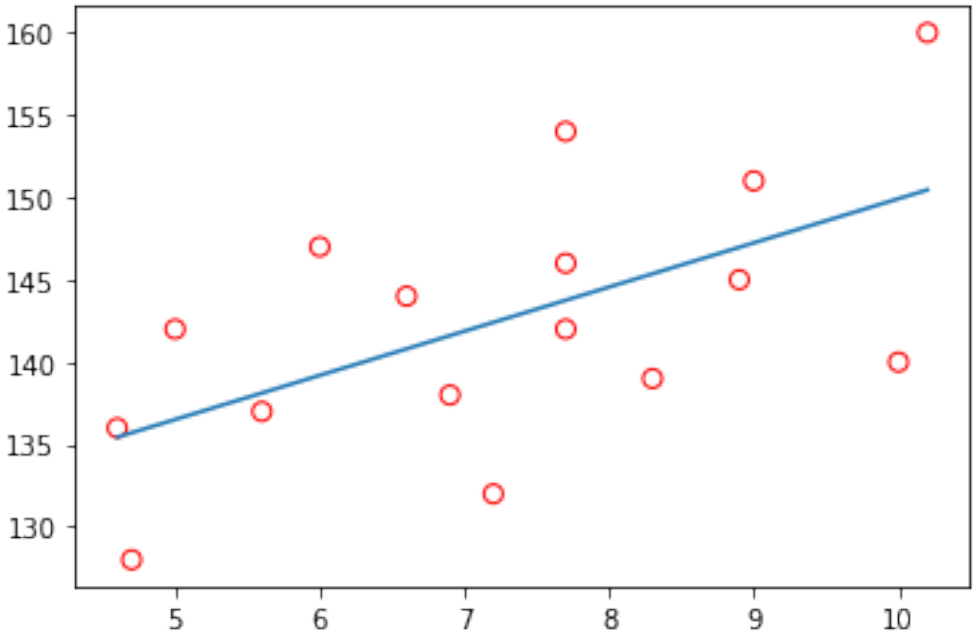
[29]:
```
ym
```

[29]:
```
array([[135.44076463],
       [135.70887703],
       [136.5132142 ],
       [138.12188854],
       [139.1943381 ],
```

```
[140.80301244],
[141.60734961],
[142.41168678],
[143.75224873],
[143.75224873],
[143.75224873],
[145.36092307],
[146.96959741],
[147.2377098 ],
[149.91883371],
[150.45505849]])
```

[35]:
```
plt.scatter(x,Y,s=50,color='r',marker="o",facecolors='none'),plt.plot(X,ym)
```

[35]:
```
(<matplotlib.collections.PathCollection at 0x7fb0d8bd9990>,
 [<matplotlib.lines.Line2D at 0x7fb0d8c1add0>])
```



3.2 PLEASE TRY YOUR FUNCTIONS TO FIND SLR VALUES

3.2.1 Part I - Example 4

[51]:
```
X=np.array([40,50,60,70,80,90,100]).reshape(-1,1)
Y=.25*X+0.4
Y
```

[51]:
```
array([[10.4],
       [12.9],
       [15.4],
       [17.9],
       [20.4],
       [22.9],
       [25.4]])
```

[52]:
```
model=LinearRegression()
model.fit(X,Y)
b0=model.intercept_
b1=model.coef_
print(b0)
print(b1)
print(b0,b1)
R2=model.score(X,Y)  # find R^2
R=np.sqrt(R2) # Find R
print(R2)
print(R)
```

```
[0.4]
[[0.25]]
[0.4] [[0.25]]
1.0
1.0
```

3.3 PLEASE TRY YOUR FUNCTIONS TO FIND SLR VALUES

4 Section II - Data Display using f'string

This section of the project is a simple way to display your data using f'string in Python.

There are two parts

- Part I - Please display Table 7.2 that is shown in Page 7-6 `SLR_main.pdf`. Try to get as close as possible to original table.
- Part II - Please display Table 7.3 that is shown in Page 7-10 `SLR_main.pdf`. Try to get as close as possible to original table.

5 Section III - Data generation and SLR fitting in Python

This section of the project is creating a simple data set and data fit using SLR. You may use `SLR functions` for this section.

- Part I - Create a `X` and `Y` with 20 values with R = 1.
- Part II - Create a `X` and `Y` with 20 values with 0.5 < R < 1.
- Part III - Create a `X` and `Y` with 20 values with R = -1.

- Part IV - Create a `X` and `Y` with 20 values with -0.5 < R < -1.

[ ]:

6 EXTRA REGRESSION ANALYSIS INFORMATION

6.1 Multiple Linear Regression With scikit-learn

[ ]:

[54]:
```
x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)
```

[55]:
```
x
```

[55]:
```
array([[ 0,  1],
       [ 5,  1],
       [15,  2],
       [25,  5],
       [35, 11],
       [45, 15],
       [55, 34],
       [60, 35]])
```

[56]:
```
y
```

[56]:
```
array([ 4,  5, 20, 14, 32, 22, 38, 43])
```

[57]:
```
model = LinearRegression().fit(x, y)
```

[58]:
```
r_sq = model.score(x, y)
```

[59]:
```
r_sq
```

[59]:
```
0.8615939258756776
```

[60]:
```
model.intercept_
```

[60]:
```
5.52257927519819
```

[61]:
```
model.coef_
```

[61]:
```
array([0.44706965, 0.25502548])
```

[62]:
```
y_pred = model.predict(x)
y_pred
```

[62]:
```
array([ 5.77760476,  8.012953  , 12.73867497, 17.9744479 , 23.97529728,
       29.4660957 , 38.78227633, 41.27265006])
```

[63]:
```
y-y_pred
```

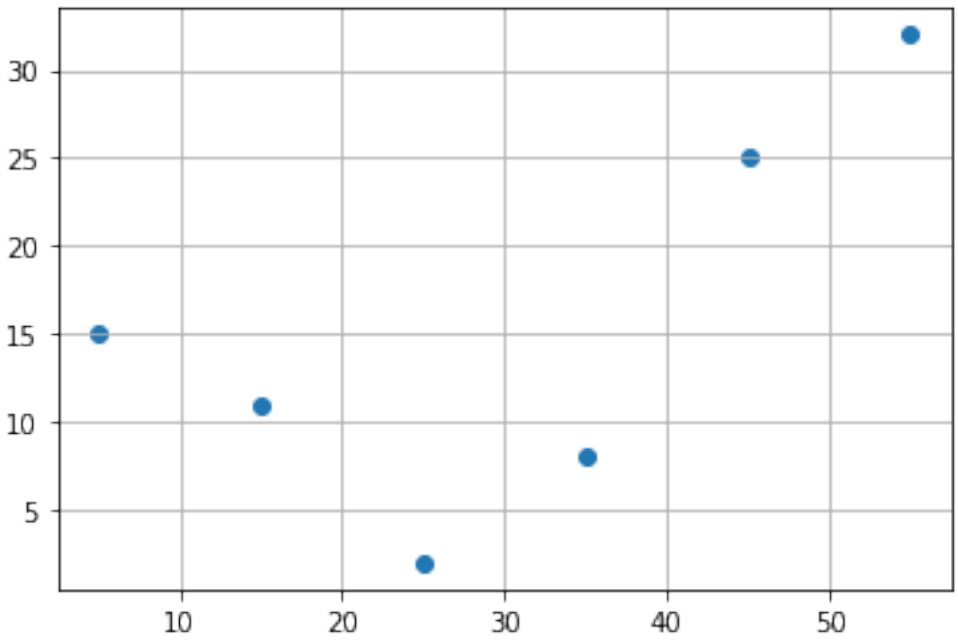
[63]:
```
array([-1.77760476, -3.012953  ,  7.26132503, -3.9744479 ,  8.02470272,
       -7.4660957 , -0.78227633,  1.72734994])
```

6.2 Polynomial Regression With scikit-learn

[64]:
```
from sklearn.preprocessing import PolynomialFeatures
```

[65]:
```
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([15, 11, 2, 8, 25, 32])
```

[66]:
```
plt.scatter(x,y)
plt.grid()
```



[67]:
```
transformer = PolynomialFeatures(degree=2, include_bias=False)
```

[68]:
```
transformer.fit(x)
```

[68]:
```
PolynomialFeatures(degree=2, include_bias=False, interaction_only=False,
                   order='C')
```

[69]:
```
x_ = transformer.transform(x)
```

[70]:
```
x_
```

[70]:
```
array([[   5.,   25.],
       [  15.,  225.],
       [  25.,  625.],
       [  35., 1225.],
       [  45., 2025.],
       [  55., 3025.]])
```

[71]:
```
model = LinearRegression().fit(x_, y)
```

[72]:
```
r_sq = model.score(x_, y)
```

[73]:
```
model.intercept_
```

[73]:
```
21.372321428571418
```

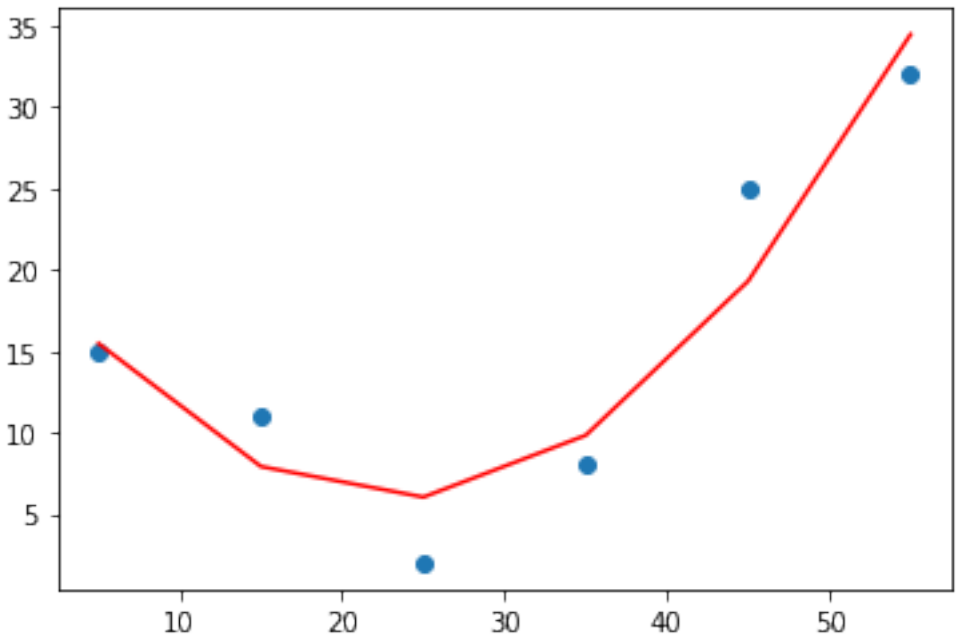
[74]:
```
model.coef_
```

[74]:
```
array([-1.32357143,  0.02839286])
```

[75]:
```
y_pred = model.predict(x_)
```

[76]:
```
plt.scatter(x,y), plt.plot(x,y_pred,'red')
```

[76]:
```
(<matplotlib.collections.PathCollection at 0x7fb0b8b5ad10>,
 [<matplotlib.lines.Line2D at 0x7fb0c0111b90>])
```



[77]:
```
# Step 1: Import packages
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Step 2a: Provide data
x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)

# Step 2b: Transform input data
x_ = PolynomialFeatures(degree=2, include_bias=False).fit_transform(x)

# Step 3: Create a model and fit it
model = LinearRegression().fit(x_, y)

# Step 4: Get results
r_sq = model.score(x_, y)
intercept, coefficients = model.intercept_, model.coef_

# Step 5: Predict
y_pred = model.predict(x_)
```

[ ]:
```

```