PRO-C114

Linear Regression

What is our GOAL for this MODULE?

The goal of this module is to learn about linear regression.

What did we ACHIEVE in the class TODAY?

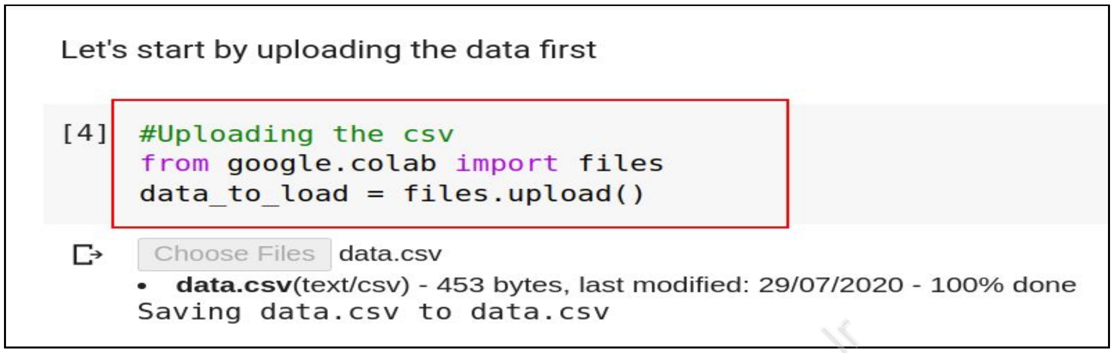
In this class we learned about the linear regression and wrote our own prediction algorithm.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Linear regression
- Prediction algorithm

How did we DO the activities?

1. We uploaded the height and weight data and plotted the height and weight data on the scatter plot.



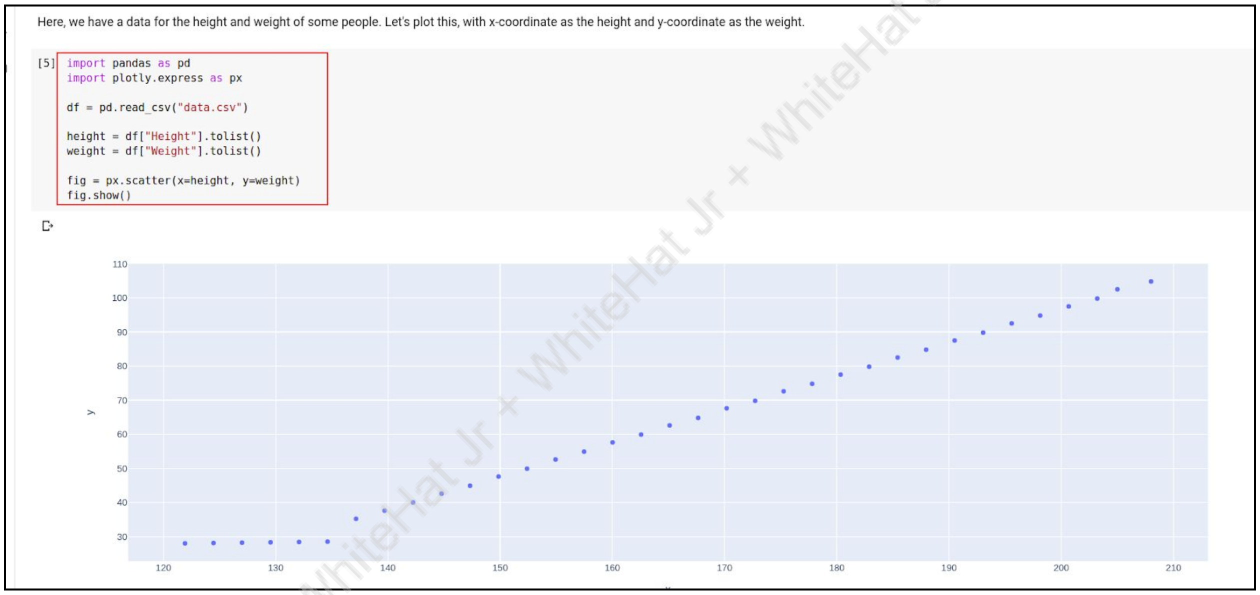
Let's start by uploading the data first

```
[4] #Uploading the csv
    from google.colab import files
    data_to_load = files.upload()
```

data.csv

- **data.csv**(text/csv) - 453 bytes, last modified: 29/07/2020 - 100% done

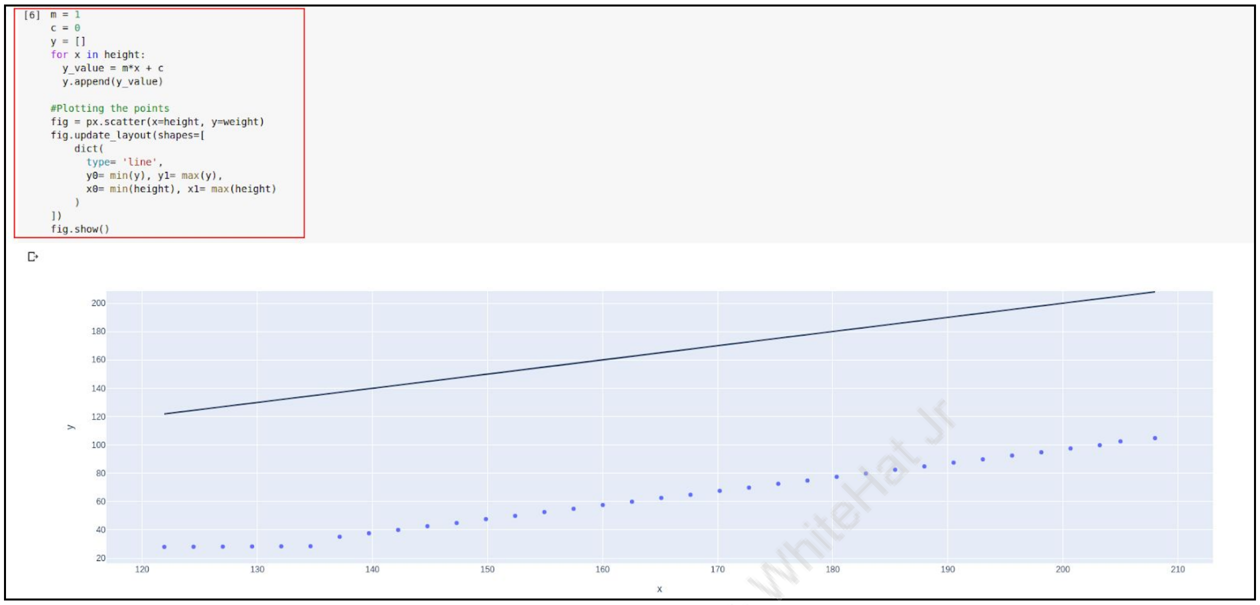
```
Saving data.csv to data.csv
```



Here, we have a data for the height and weight of some people. Let's plot this, with x-coordinate as the height and y-coordinate as the weight.

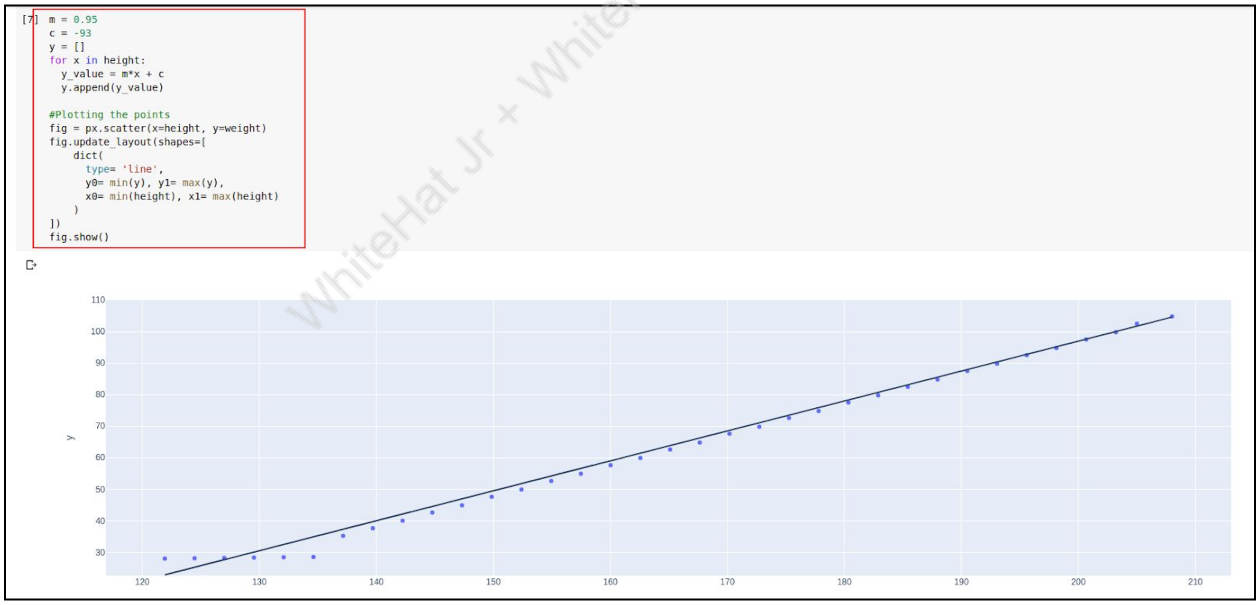
```
[5] import pandas as pd
    import plotly.express as px

    df = pd.read_csv("data.csv")

    height = df["Height"].tolist()
    weight = df["Weight"].tolist()

    fig = px.scatter(x=height, y=weight)
    fig.show()
```

2. We saw the line equation and tried to plot a line on the points in our graph.



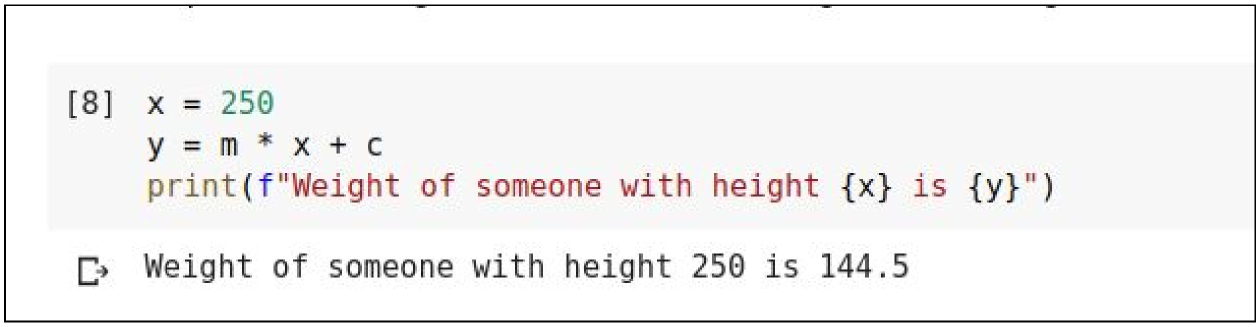
```
[6]  m = 1
     c = 0
     y = []
     for x in height:
       y_value = m*x + c
       y.append(y_value)

     #Plotting the points
     fig = px.scatter(x=height, y=weight)
     fig.update_layout(shapes=[
         dict(
           type= 'line',
           y0= min(y), y1= max(y),
           x0= min(height), x1= max(height)
         )
     ])
     fig.show()
```

3. We did the hit and trial method to plot the line on the points in graph.



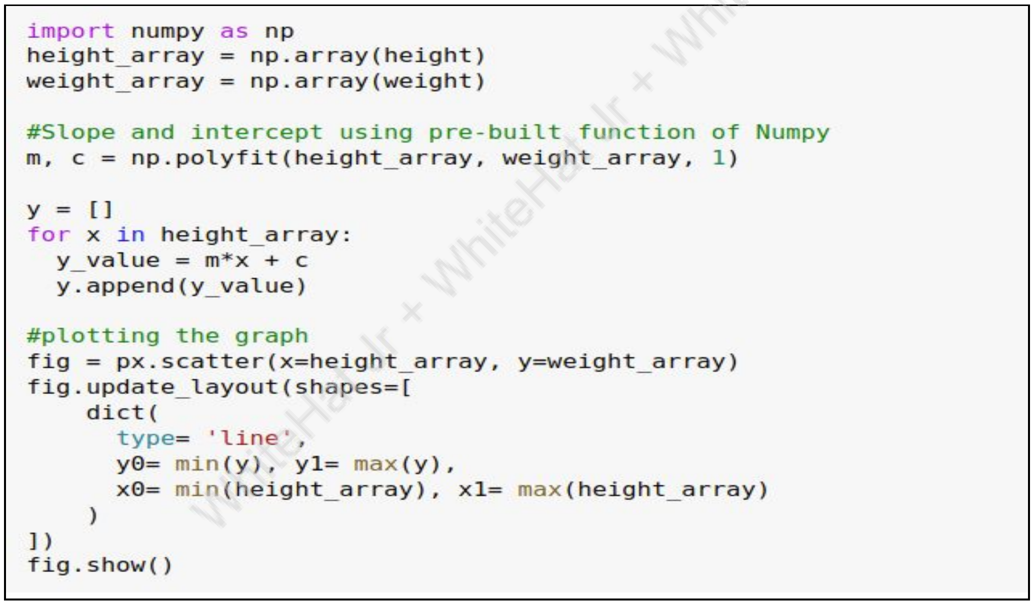
```
[7]  m = 0.95
     c = -93
     y = []
     for x in height:
       y_value = m*x + c
       y.append(y_value)

     #Plotting the points
     fig = px.scatter(x=height, y=weight)
     fig.update_layout(shapes=[
         dict(
           type= 'line',
           y0= min(y), y1= max(y),
           x0= min(height), x1= max(height)
         )
     ])
     fig.show()
```

4. By doing the hit and trial method we got the values for slope and intercept. Using the line equation we predicted the weight of a person based on the height.

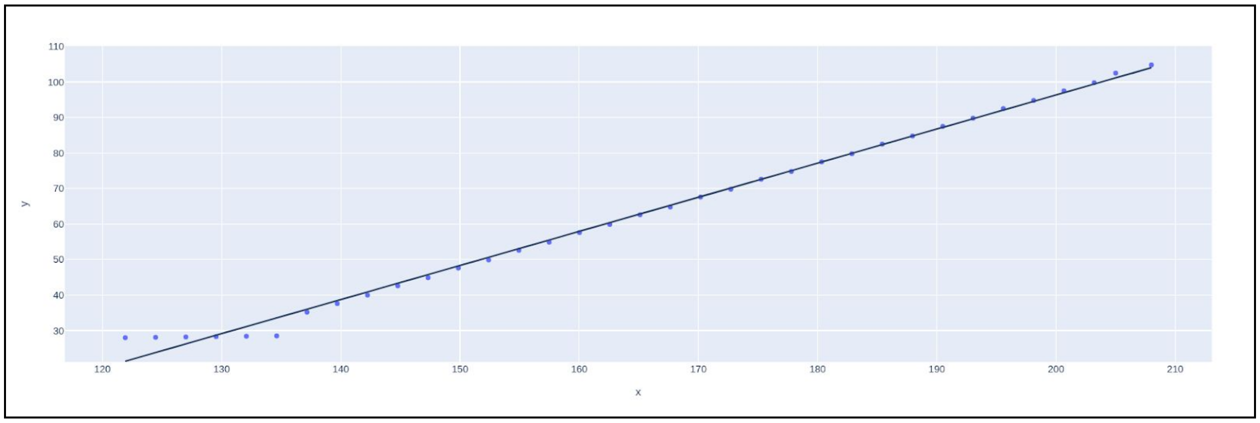


```
[8]  x = 250
     y = m * x + c
     print(f"Weight of someone with height {x} is {y}")

     Weight of someone with height 250 is 144.5
```

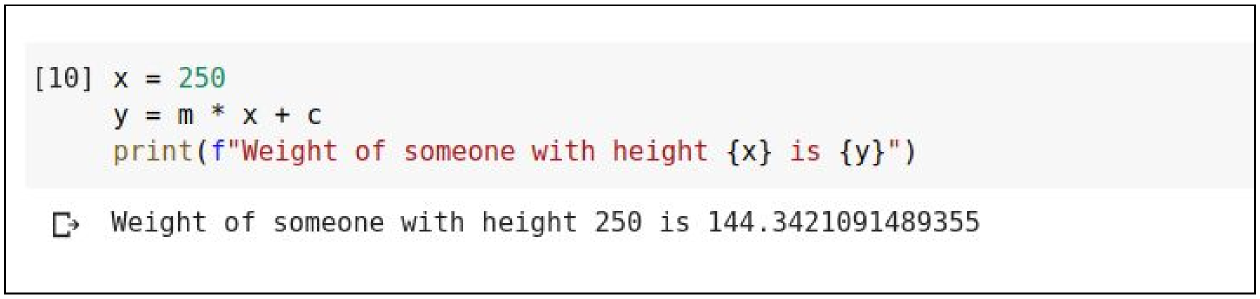
5. We then used a predefined computer algorithm for the best fit line and we got our values of slope and intercept.



```
import numpy as np
height_array = np.array(height)
weight_array = np.array(weight)

#Slope and intercept using pre-built function of Numpy
m, c = np.polyfit(height_array, weight_array, 1)

y = []
for x in height_array:
  y_value = m*x + c
  y.append(y_value)

#plotting the graph
fig = px.scatter(x=height_array, y=weight_array)
fig.update_layout(shapes=[
    dict(
      type= 'line',
      y0= min(y), y1= max(y),
      x0= min(height_array), x1= max(height_array)
    )
])
fig.show()
```



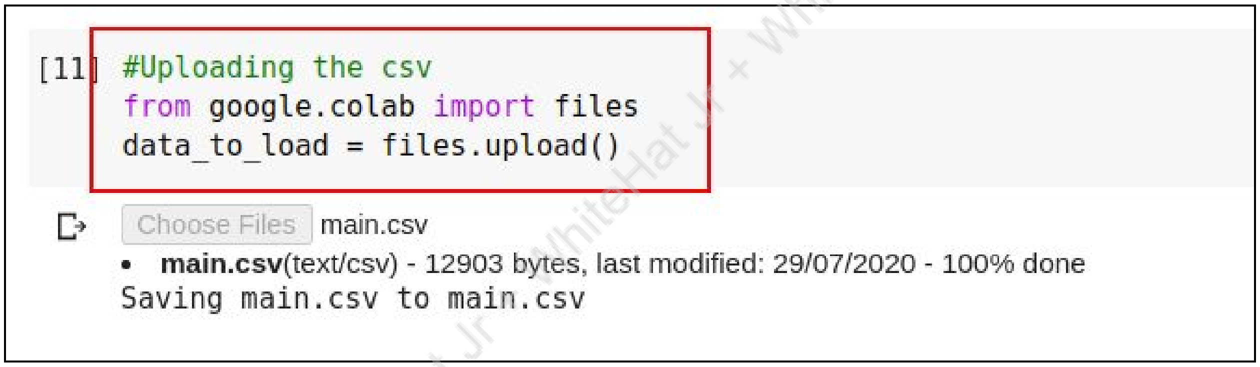
6. Then using the slope and intercept values that we got from the computer algorithm we predicted the weight of the person by their height.



```
[10] x = 250
     y = m * x + c
     print(f"Weight of someone with height {x} is {y}")
```

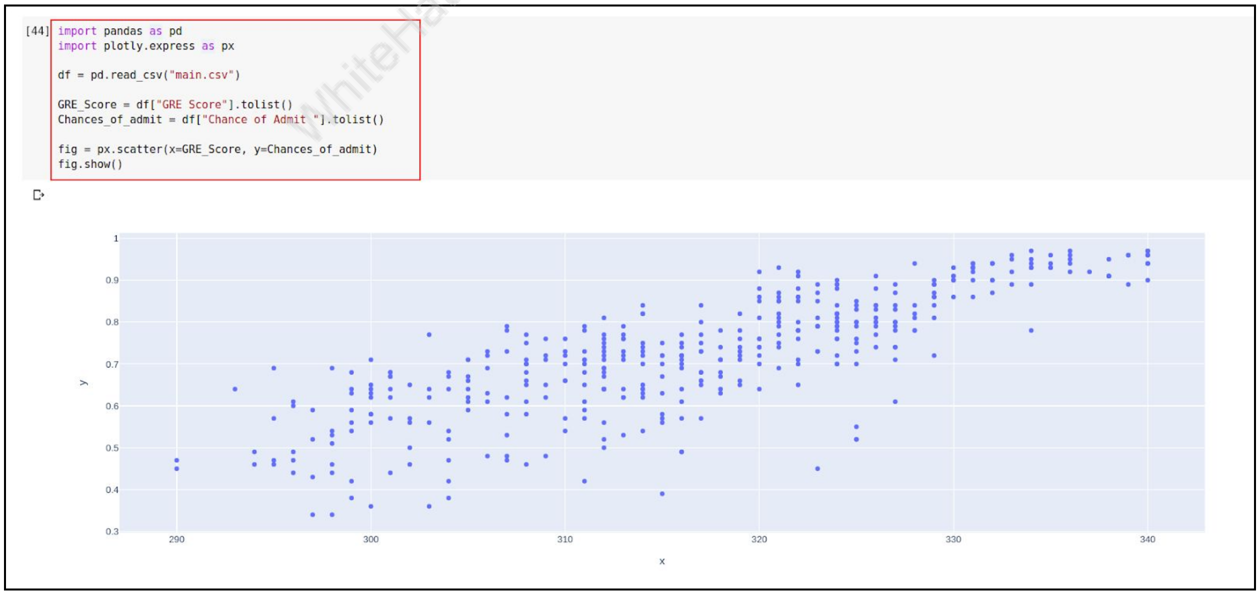
```
Weight of someone with height 250 is 144.3421091489355
```

7. We concluded that the weight that we found from our algorithm was the same as the weight that we got from the predefined algorithm.
8. We repeated the same process with another data set of scores.



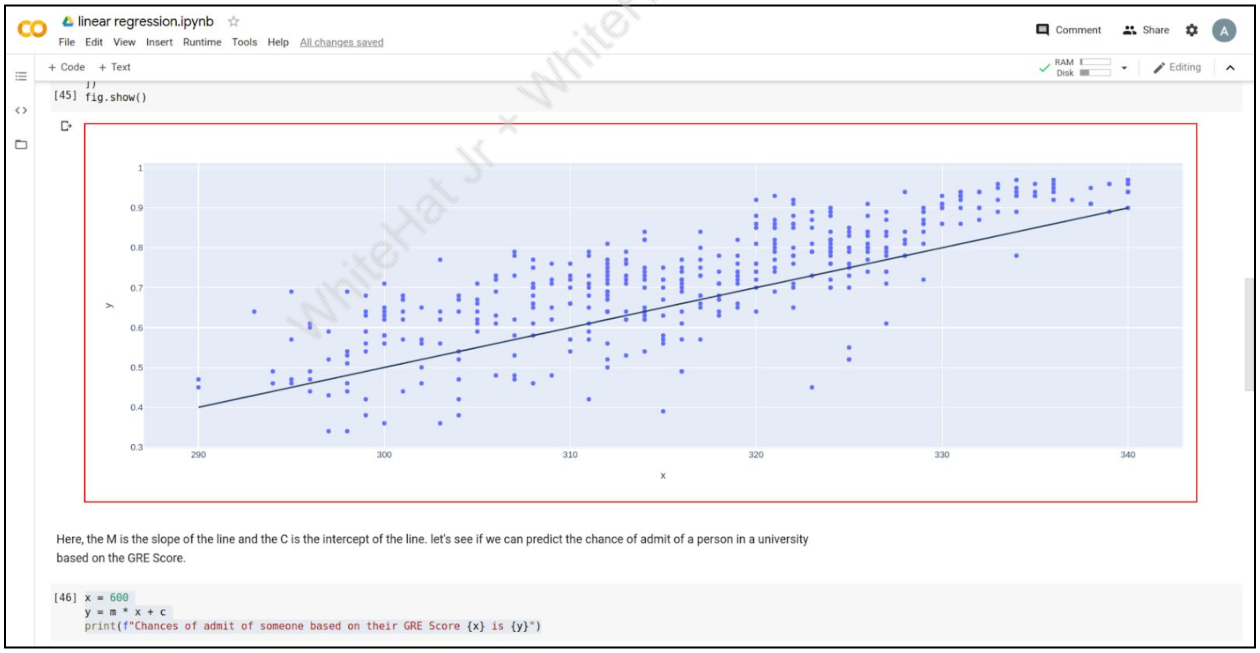
```
[11] #Uploading the csv
     from google.colab import files
     data_to_load = files.upload()
```

```
Choose Files main.csv
• main.csv(text/csv) - 12903 bytes, last modified: 29/07/2020 - 100% done
Saving main.csv to main.csv
```



```
[44] import pandas as pd
     import plotly.express as px

     df = pd.read_csv("main.csv")

     GRE_Score = df["GRE Score"].tolist()
     Chances_of_admit = df["Chance of Admit "].tolist()

     fig = px.scatter(x=GRE_Score, y=Chances_of_admit)
     fig.show()
```

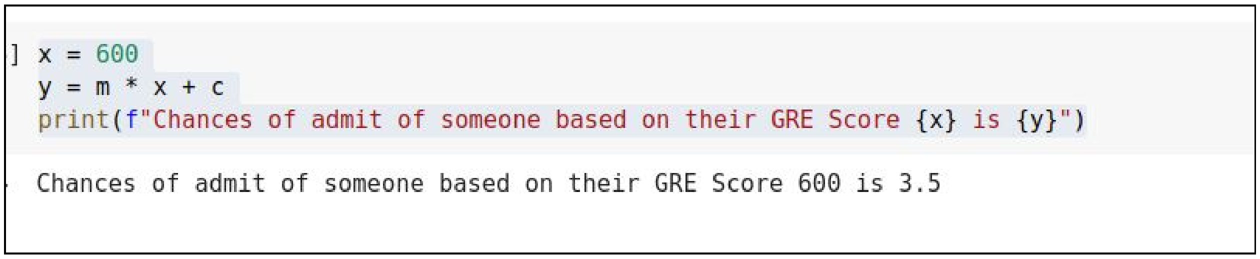
```
m = 0.01
c = -2.5
y = []
for x in GRE_Score:
  y_value = m*x + c
  y.append(y_value)

#Plotting the points
fig = px.scatter(x=GRE_Score, y=Chances_of_admit)
fig.update_layout(shapes=[
    dict(
      type= 'line',
      y0= min(y), y1= max(y),
      x0= min(GRE_Score), x1= max(GRE_Score)
    )
])
fig.show()
```



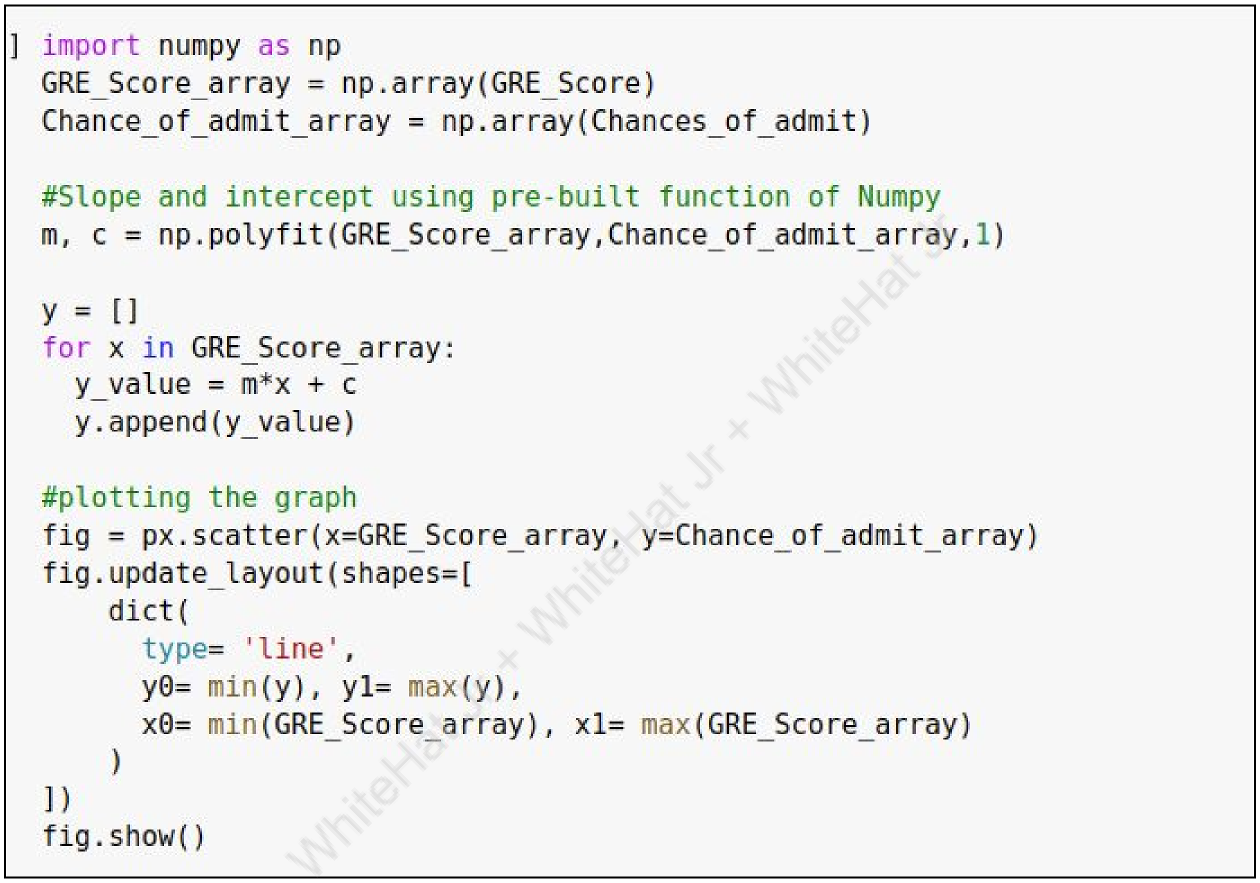
```
[45] fig.show()
```

Here, the M is the slope of the line and the C is the intercept of the line. let's see if we can predict the chance of admit of a person in a university based on the GRE Score.

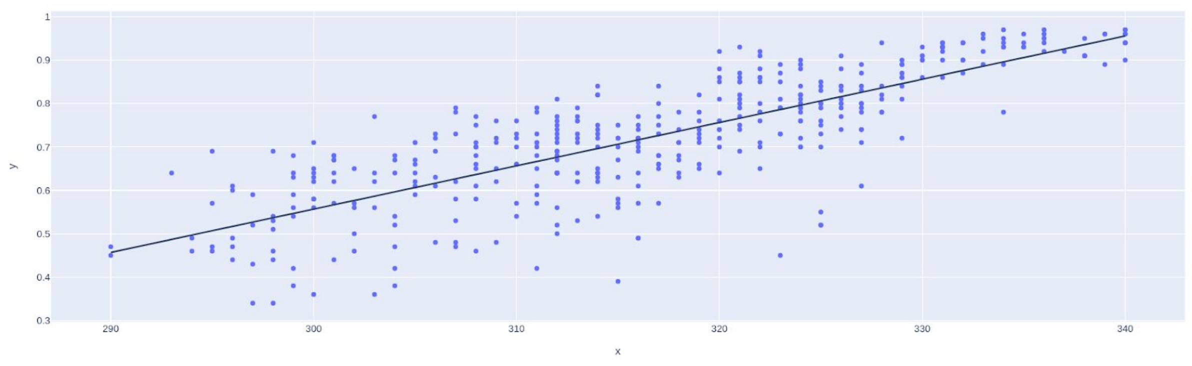
```
[46] x = 600
     y = m * x + c
     print(f"Chances of admit of someone based on their GRE Score {x} is {y}")
```

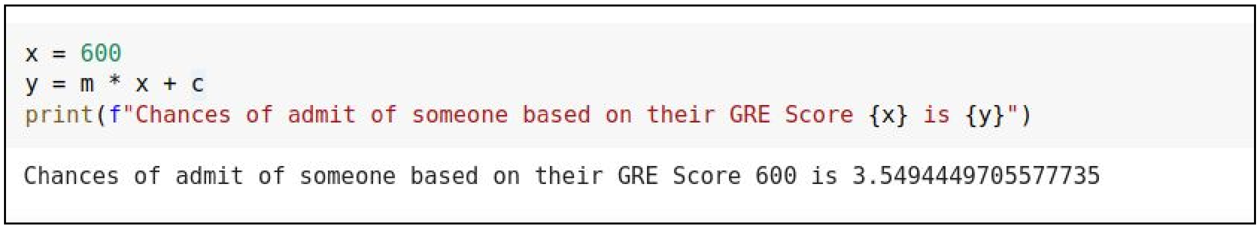


```
] x = 600
  y = m * x + c
  print(f"Chances of admit of someone based on their GRE Score {x} is {y}")

  Chances of admit of someone based on their GRE Score 600 is 3.5
```



```
] import numpy as np
  GRE_Score_array = np.array(GRE_Score)
  Chance_of_admit_array = np.array(Chances_of_admit)

  #Slope and intercept using pre-built function of Numpy
  m, c = np.polyfit(GRE_Score_array,Chance_of_admit_array,1)

  y = []
  for x in GRE_Score_array:
    y_value = m*x + c
    y.append(y_value)

  #plotting the graph
  fig = px.scatter(x=GRE_Score_array, y=Chance_of_admit_array)
  fig.update_layout(shapes=[
      dict(
        type= 'line',
        y0= min(y), y1= max(y),
        x0= min(GRE_Score_array), x1= max(GRE_Score_array)
      )
  ])
  fig.show()
```





```
x = 600
y = m * x + c
print(f"Chances of admit of someone based on their GRE Score {x} is {y}")
```

```
Chances of admit of someone based on their GRE Score 600 is 3.5494449705577735
```

We got the same value for chance of admission with our own algorithm as we got from the computer algorithm.

What's NEXT?
In the next class, we will learn about machine learning.

EXTEND YOUR KNOWLEDGE:
From the following link you can read about polyfit :
https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html