Topic	Linear Regression	
Class Description	**Students learn about regression and write their own prediction algorithm.**	
Class	**C114**	
Class time	**45 mins**	
Goal	• Learn about regression • Write prediction algorithm to predict the weight of a person by given weight • Apply hit and trial method to find the proper values of slope and intercept of the line.	

Resources Required	• Teacher Resources
	  ○ Google Colaboratory (Colab)
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	• Student Resources
	  ○ Google Colaboratory (Colab)
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

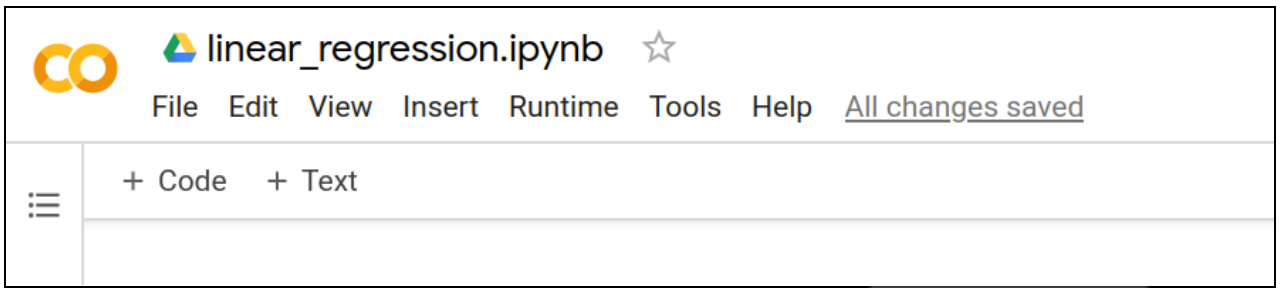
Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

CONTEXT

- **Introduce regression and write their own prediction algorithm and compare the output with the predefined computer algorithm.**

Class Steps	Teacher Action	Student Action

Step 1: **Warm Up** **(5 mins)**	Hi <Student Name>! How are you doing today? Let's quickly revise what we did in last class.	**ESR:** - In our last class we did a data analysis using visualizations to write our data story.
	Till now we just visualised the data that we already had and made conclusions. Today we'll learn to predict the future by doing a small analysis . Sounds exciting?	**ESR:** Yes!
	Let's get started then	-
Teacher Initiates Screen Share		
CHALLENGE • **Use hit and trial method to find the values of slope and intercept and compare the output with a predefined algorithm.**		
Step 2: **Teacher-led Activity** **(15 min)**	So we'll predict a person's weight based on his/her height.	-
	*<Teacher downloads the data from the **Teacher Activity 1**>* *Teacher opens Colab from **Teacher Activity 2** and creates a new notebook and names it linear regression.* Can you tell me what is the meaning of the word regression in statistics?	**ESR:** varied



	Regression takes a group of random variables, thought to be predicting Y, and tries to find a mathematical relationship between them. This relationship is typically in the form of a straight line (linear **regression**) that best approximates all the individual data points *<Teacher uploads the height and weight data in the colab>* Now let's plot the data on the scatter plot . *Teacher codes to plot the height as x-coordinate and the weight as y-coordinate.* Code:- **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()**	*Student helps the teacher to plot the graph.*

Code to upload files:-

Let's start by uploading the data first

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

Choose Files data.csv

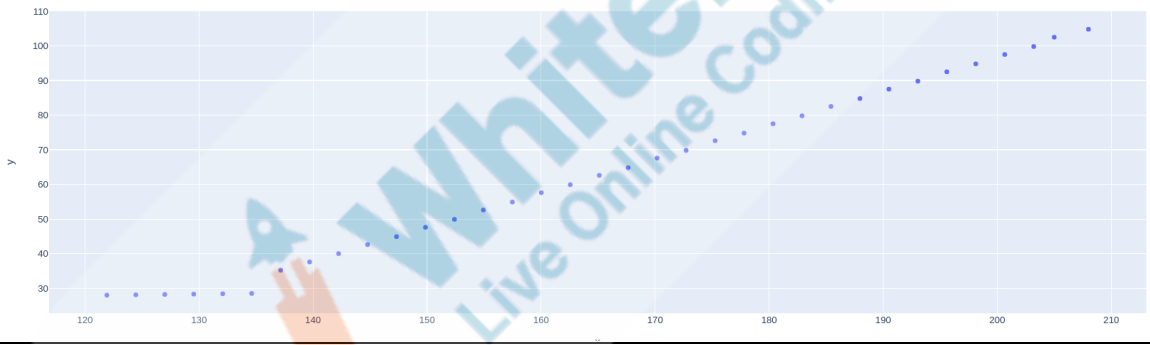
- **data.csv**(text/csv) - 453 bytes, last modified: 29/07/2020 - 100% done

Saving data.csv to data.csv

Code to plot the data on the scatter plot.:-

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()
```

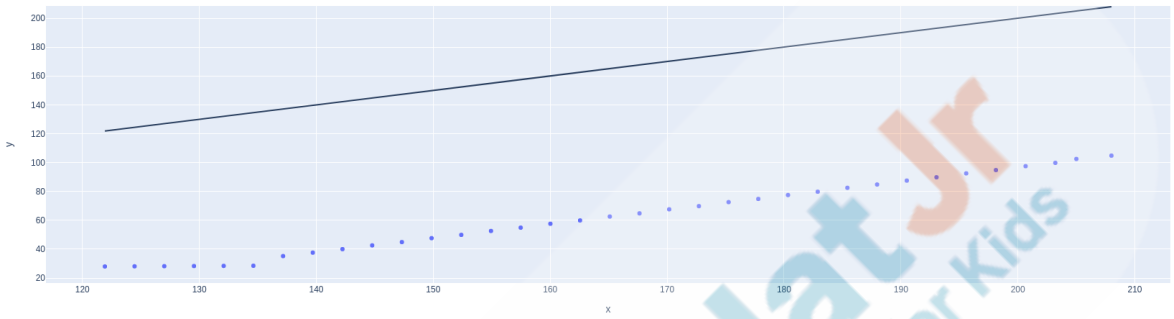


	What do we see here? Perfect! Do you know the formula for a slope of a line? Every line can be represented by an equation. The equation is as follows. **y = mx + c** Here **m** = slope **c** =intercept on y -axis **x** = values of x	**ESR:** We can see that the height and weight are related to each other, the more the height the more the weight. **ESR:** Varied!

	Let's use this formula to plot the line on the above graph. We know the values of x, we'll assume the values of **slope** and **intercept**. We'll be using the hit and trial method to plot. Hit and trial means to guess a solution and see if it's valid or not. We will do this until we get the proper values for slope and intercept. Code:- **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()** What do you see?	**ESR:** Our line is far from the points that we have on the graph.

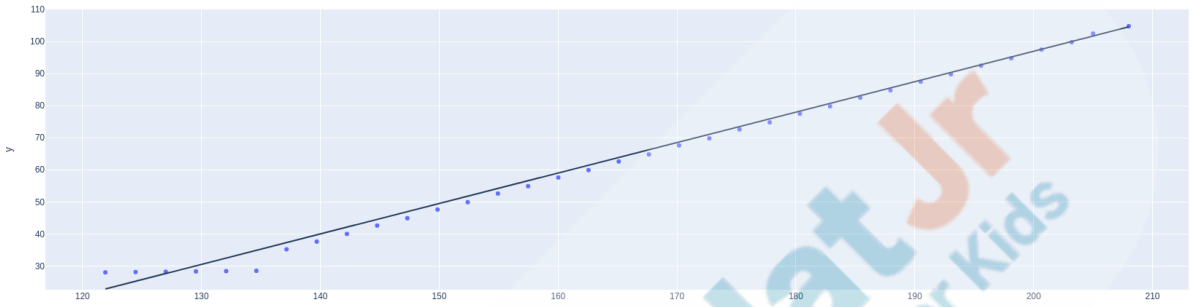
<table>
<tr>
<td colspan="3">

```
[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()
```



</td>
</tr>
<tr>
<td></td>
<td>Yes! means the values of slope(m) and intercept (c) are not correct and we need to change them to get the line on the graph.
<Teacher tries changing values of "m" and "c" to see the different outcomes.></td>
<td><The student suggest teacher different values of x></td>
</tr>
<tr>
<td></td>
<td>Teacher changes the value of "m" to 0.95 and "c" to -93.
Then checks the graph.
Now what do we see?</td>
<td>ESR:
We see that the line fits perfectly on the dots in the graph.</td>
</tr>
</table>

```
[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()
```



Perfect so now we have the proper values of m and c.
Let's use these values in our line formula to predict the weight of a person based on the height we provide.
<Teacher use the values in the formula to find weight of a person>
Code:

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

Let's take the value of x as 250 which is the height and we get the weight as 144.34.
<Teacher tries with multiple values of X to see the different weights we get>

	But as we got our values of slope and intercept by hit and trail we might be a little accurate.	

```
[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
```

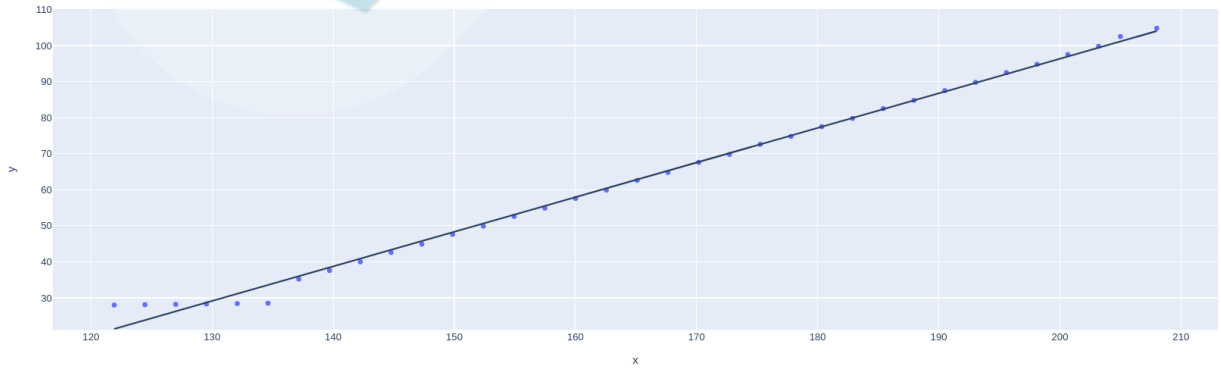
	There is also a computer algorithm which will help us find the proper values of m and c for us without hit and trial. Let's find the slope and the intercept of the data that we have using computer algorithms and plot again with the new values of m and c. <Teacher writes the following code Code: #Importing the needed libraries **import numpy as np** **height = np.array(height)** **weight = np.array(weight)** **#Slope and intercept using pre-built function of Numpy** **m, c = np.polyfit(height, weight, 1)** **y = []** **for x in height:** **y_value = m*x + c** **y.append(y_value)** **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()
```

What do we see?

Yes! It's called the best fit line.

```
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()
```



	This is the best fit line for the data that the computer algorithm has provided us. Let's see what will be the weight of someone with height 250 or the random values we used before using these new values for slope and intercept. Code:- **x = 250** **y = m * x + c** **print(f"Weight of someone with height {x} is {y}")** *<The teacher uses the different values of x that she tried before>* Do we find any similarities between the value we got earlier and the value we got now?	**ESR:** Yes the values are very similar. We earlier got the value of 144.5 and with the computer provided best fit line, we got a result of 144.34

```
[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
```

	I have a challenge for you. Can you build an algorithm to predict if the person will get into college or not by checking his/her GRE scores?	**ESR:** Yes!
	Alright let's do it.	

<table>
<tr><td colspan="3">Teacher Stops Screen Share</td></tr>
<tr><td></td><td>Now it's your turn. Please share your screen with me.</td><td></td></tr>
<tr><td colspan="3">

- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

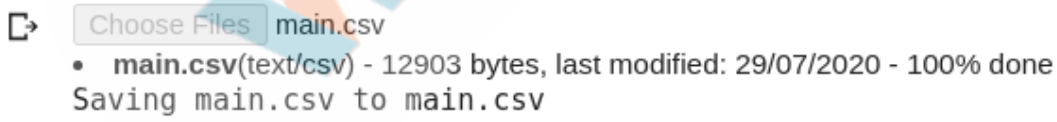
</td></tr>
<tr><td colspan="3">

ACTIVITY

- **Write a prediction algorithm and compare its output with a pre-built computer algorithm.**

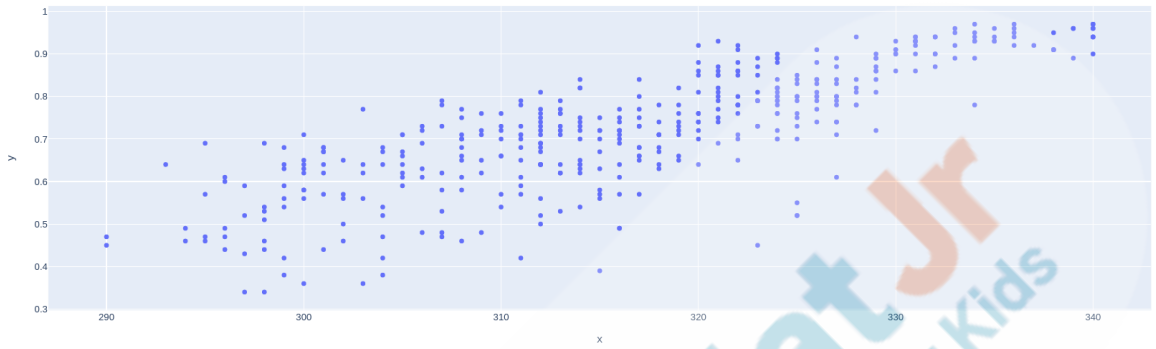
</td></tr>
<tr><td>Step 3:
Student-Led Activity
(15 min)</td><td><Teacher helps student download the data and plot it on the scatter plot.></td><td>Student downloads the data from Student Activity 1
Opens a new Colab Notebook from Student Activity 2, uploads the data and plots it on the scatter plot.</td></tr>
<tr><td colspan="3">

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



</td></tr>
</table>

```
[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()
```



	Now we have to plot the line on the graph. *Teacher helps the student use hit and trial to find the best fit line for the graph.* *<Teacher can suggest values of m = 0.01 and c = -2.5 if student hasn't found the values after multiple tries.>*	*Student tries changing the value of slope and intercept to fit the line on the dots. And makes a note of the values when the line fits the dots.*

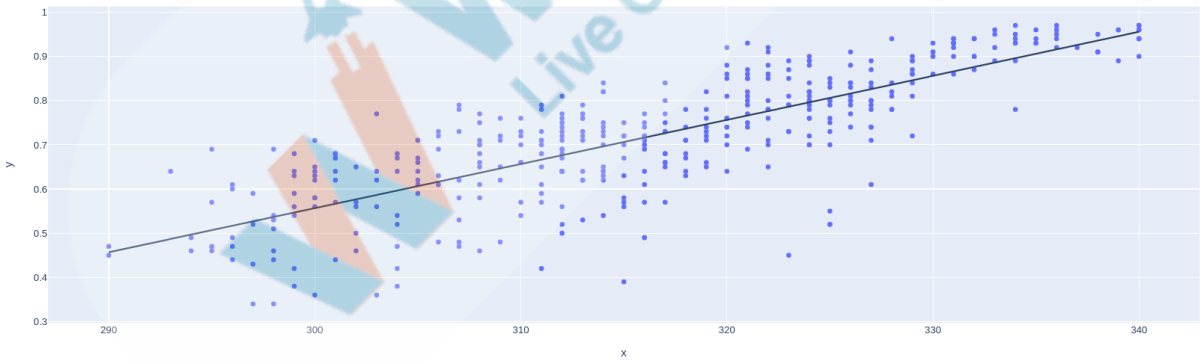
```
m = 0.01
c = -2.5
y = []
for x in GRE_Score:
  y_value = m*x + c
  y.append(y_value)

#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()
```

	Teacher helps the student add the values of slope and intercept to the code and find the different values of chance of admission for different values of GRE Score.	*The student adds the values of slope and intercept to the line formula and finds the different values for chance of admission for different GRE Scores.*

```
] 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
```

	Teacher helps the student to find the values of slope and intercept using the predefined computer algorithm. And then plot the graph with the line.	*The student codes to use the predefined computer algorithm to find the best values of slope and intercept and plot the graph.*

```
] 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()
```



	Teacher helps the student to use the values of slope and intercept found by the computer algorithm in the line formula and finds the different values for the chance of admission by changing the GRE Score values.	*Using the values of slope and intercept found by using the computer algorithm in the slope formula, the student finds the value for the chance of admission for the different GRE scores.*

```
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
```

	What do we observe?	**ESR:** We see that the score we got from the hit and trial method is the same as the score we got from the computer algorithm.

Teacher Guides Student to Stop Screen Share

FEEDBACK

- **Appreciate the student for their efforts**
- **Identify 2 strengths and 1 area of improvement for the student**

Step 4: **Wrap-Up** **(5 min)**	That was some awesome work today. How did it feel to write a prediction algorithm?	**ESR:** varied

	This is also a type of machine learning algorithm.	
	In the next class we'll learn more about machine learning.	-
Project Overview	**Linear Regression** **Goal of the Project:** In this project you will apply what you learned in the class and create your own prediction model. **Story:** In our journey of analyzing the articlele's data, you also want to understand how the results are changing after the introduction of an intervention. write a program to do the z test of a given sample. . I am very excited to see your project solution and I know you will do really well. Bye Bye!	
Teacher Clicks	✖ End Class	

Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? - Describe what happened - Code I wrote ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflection in a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	height and weight data	https://procodingclass.github.io/datasets/data.csv
Teacher Activity 2	Colab notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true
Teacher Activity 3	Teacher reference	https://colab.research.google.com/gist/shubhamwhj/e31b827010dc456b03e5c2a196e017c4/c114-ta3-reference-code.ipynb
Teacher Activity 4	Student side reference	https://colab.research.google.com/gi

		st/shubhamwhj/1fcbbe100d0327580ff9c98478653824/c114-ta4-reference-code.ipynb
Student Activity 1	Scores data	https://procodingclass.github.io/datasets/main.csv
Student Activity 2	Colab notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true