Topic	Logistic Regression	
Class Description	**Students use sigmoid function to model classification problems with the linear regression.**	
Class	**C115**	
Class time	**45 mins**	
Goal	• Learn about the using sigmoid function on the data which has binary values	
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

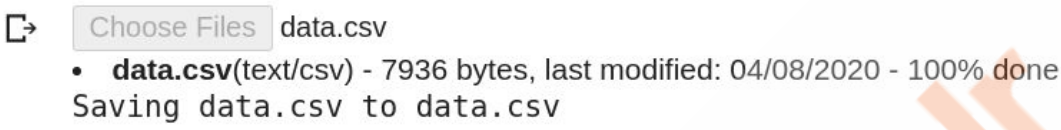
- **Learn to deal with the data which has binary values in it.**

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 the last class	ESR: - In the last class we wrote a prediction algorithm to predict the saving

	In this class we'll see how to deal with the data which has binary values. Sounds exciting?	ESR: varied
	Let's quickly get started	-
Teacher Initiates Screen Share		
CHALLENGE • **Learn the usage of sigmoid function in linear regression.** • **plot a graph using sigmoid function** • **Write a prediction algorithm.**		
Step 2: Teacher-led Activity (15 min)	So here we have a data of 1000 students that applied to a university. We know the scores of these students, and we also know if they were accepted to the university or not. So let's try to build a model which will predict if the student will get enrolled or not based on the marks obtained by him/her.	-
	<Teacher downloads the data from Teacher Activity 1> <Teacher opens the Colab and creates a new Colab notebook form Teacher Activity 2 and names it as Sigmoid function> Teacher codes to upload the data to the notebook. Code for reference:- #Uploading the csv	Student helps the teacher write the code.

<table>
<tr><td></td><td>from google.colab import files
data_to_load = files.upload()</td><td></td></tr>
<tr><td colspan="3">

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



</td></tr>
<tr><td></td><td>Now we have the code let's plot and see how the data looks on the scatter plot.

<Teacher writes code to plot the data on the scatter plot.>

Code for reference:-

#Plotting the data on the graph
import pandas as pd
import plotly.express as px

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

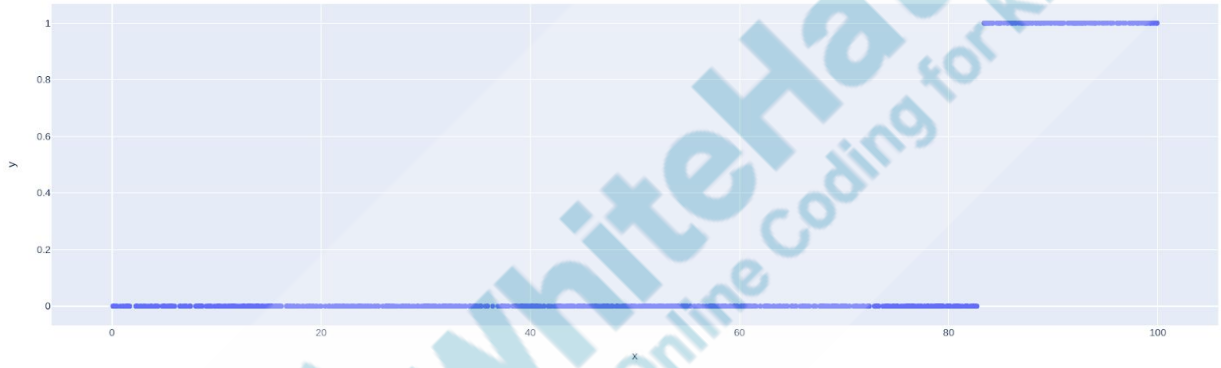
score_list = df["Score"].tolist()
accepted_list =
df["Accepted"].tolist()

fig = px.scatter(x=score_list,
y=accepted_list)
fig.show()

What do you see?</td><td>Student helps the teacher write the code.

ESR:-
We can see that the plot is completely split in two parts.

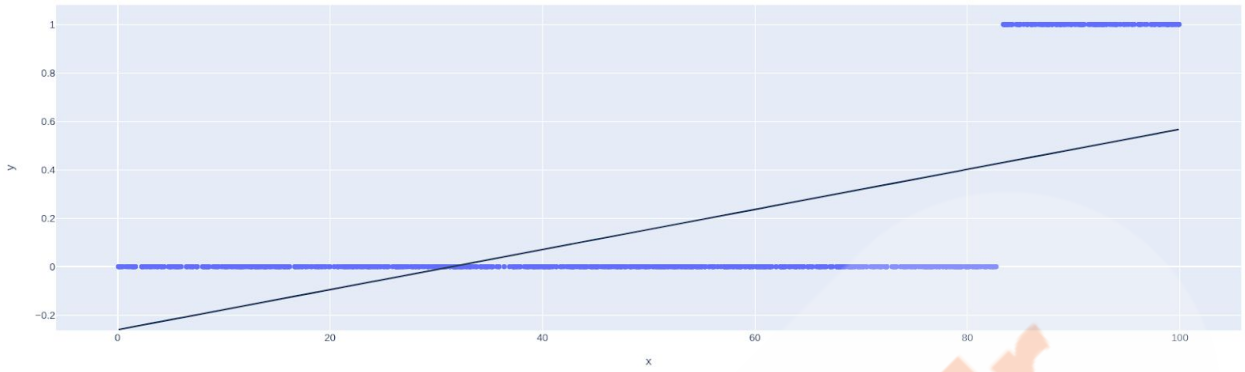
Varied!</td></tr>
</table>

	How do you think the line of regression(best fit line) look?	
``] import pandas as pd import plotly.express as px df = pd.read_csv("data.csv") score_list = df["Score"].tolist() accepted_list = df["Accepted"].tolist() fig = px.scatter(x=score_list, y=accepted_list) fig.show() ```  		
	Let's plot the line of regression and see. <Teacher uses the prebuilt function to plot the line of regression> Code for reference:- **import numpy as np** **score_array = np.array(score_list)** **accepted_array = np.array(accepted_list)** **#Slope and intercept using pre-built function of Numpy**	<Student helps the teacher with the code> ESR:- The line is not proper and it dosen't make sense as the students who have scored 82 in the exams were not accepted .

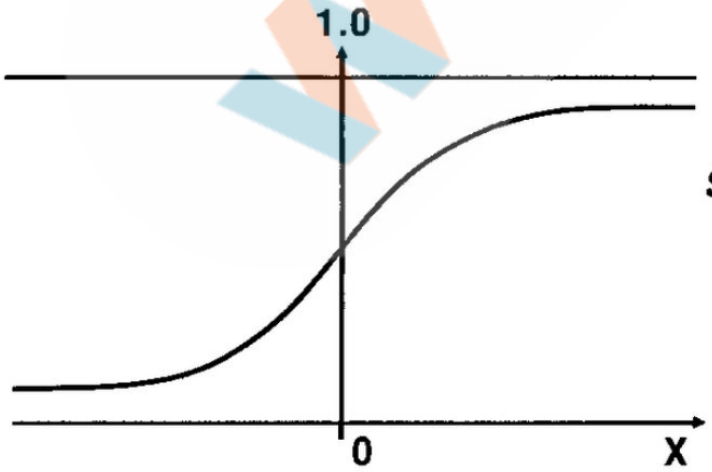
```
m, c = np.polyfit(score_array,
accepted_array, 1)

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

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

How does the line look?

```
import numpy as np
score_array = np.array(score_list)
accepted_array = np.array(accepted_list)

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

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

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



	Yes! And at times like this when we have a little vague data we use sigmoid function to represent the graph in an understable way. The reason we choose a sigmoid function to model classification problems solved with logistic regression is that we want to make sure that predicted value has a defined range which aids us in differentiating the classes. This has proved to be very useful in binary classification problems.	-



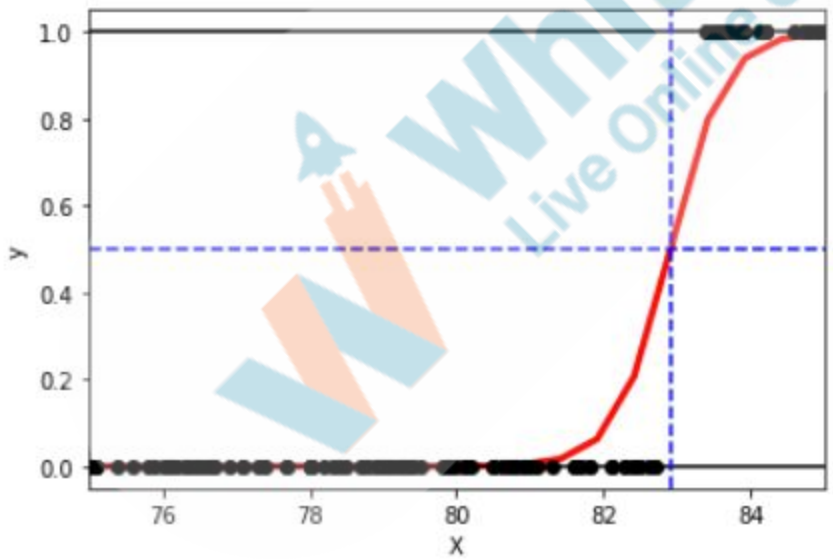
Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

	The logistic sigmoid function is defined as **(1/(1 + e^-x))** and it takes an input x of any real number and returns an output value in the range of -1 and 1. As we see here the X in the image is value of **mx+c** Let's plot the graph using the sigmoid function. Here we'll use hit and trial on the X_test to find the value which will lie on the line at the intersection Code for reference:- **import matplotlib.pyplot as plt** **from sklearn.linear_model import LogisticRegression** **X = np.reshape(score_list, (len(score_list), 1))** **Y = np.reshape(accepted_list, (len(accepted_list), 1))** **lr = LogisticRegression()** **lr.fit(X, Y)** **plt.figure()** **plt.scatter(X.ravel(), Y, color='black', zorder=20)** **def model(x):** **return 1 / (1 + np.exp(-x))** **X_test = np.linspace(0, 100, 200)** **chances = model(X_test * lr.coef_ +**	Student observes and learns. ESR:- We can see a s shaped form of line on the graph and the line is divided in different classes.

	lr.intercept_).ravel() **plt.plot(X_test, chances, color='red', linewidth=3)** **plt.axhline(y=0, color='k', linestyle='-')** **plt.axhline(y=1, color='k', linestyle='-')** **plt.axhline(y=0.5, color='b', linestyle='--')** **# do hit and trial on the value of X_test to find the value which lies on the line at the intersection.** **plt.axvline(x=X_test[165], color='b', linestyle='--')** **plt.ylabel('y')** **plt.xlabel('X')** **plt.xlim(75, 85)** **plt.show()** np.reshape = reshapes the arrays without changing the data. np.linspace =returns evenly spaced numbers over a specified interval. axhline stands for axis horizontal line. ravel() is used to convert 2 arrays into one. What do we see here?	

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

X = np.reshape(score_list, (len(score_list), 1))
Y = np.reshape(accepted_list, (len(accepted_list), 1))

lr = LogisticRegression()
lr.fit(X, Y)

plt.figure()
plt.scatter(X.ravel(), Y, color='black', zorder=20)

def model(x):
  return 1 / (1 + np.exp(-x))

#Using the line formula
X_test = np.linspace(0, 100, 200)
chances = model(X_test * lr.coef_ + lr.intercept_).ravel()

plt.plot(X_test, chances, color='red', linewidth=3)
plt.axhline(y=0, color='k', linestyle='-')
plt.axhline(y=1, color='k', linestyle='-')
plt.axhline(y=0.5, color='b', linestyle='--')

# do hit and trial by changing the value of X_test
plt.axvline(x=X_test[165], color='b', linestyle='--')

plt.ylabel('y')
plt.xlabel('X')
plt.xlim(75, 85)
plt.show()
```



	Now we have got our values of slope and intercept . We'll write a small code where we'll give the marks scored by the student	Student gives random scores to the teacher to test the prediction algorithm.

	as input and it will tell us the chances of the student being accepted by the college. <Teacher codes to write a algo for prediction> Code for reference:- **user_score = float(input("Enter your marks here:- "))** **chances = model(user_score * lr.coef_ + lr.intercept_).ravel()[0]** **if chances <= 0.01:** **print("The student will not get accepted")** **elif chances >= 1:** **print("The student will get accepted!")** **elif chances < 0.5:** **print("The student might not get accepted")** **else:** **print("The student may get accepted")** <Teacher runs the code and gives some marks as input to check the output>	

```
user_score = float(input("Enter your marks here:- "))
chances = model(user_score * lr.coef_ + lr.intercept_).ravel()[0]
if chances <= 0.01:
  print("The student will not get accepted")
elif chances >= 1:
  print("The student will get accepted!")
elif chances < 0.5:
  print("The student might not get accepted")
else:
  print("The student may get accepted")
```

```
Enter your marks here:- 99
The student will get accepted!
```

	Alright! So this is how data scientists do predictions using the previous data to arrive at the necessary conclusions for effective planning where the data has one constantly changing variable and one variable which has binary values(0 or 1).	ESR:- Yes!!
	I have challenge for you. Can you write your own prediction algorithm to find if the the tungsten metal is melted at the the given temperature.?	
	Let's get started then	
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	

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

ACTIVITY

- **Plot a graph using sigmoid function.**
- **Perform analysis on the given data and write a prediction algorithm.**

Step 3: Student-Led Activity (15 min)	Teacher helps student download the data and upload it in new notebook.	Student downloads the data from **Student Activtiy 1** Student opens a new Google colab notebook from **Student Activity 2**

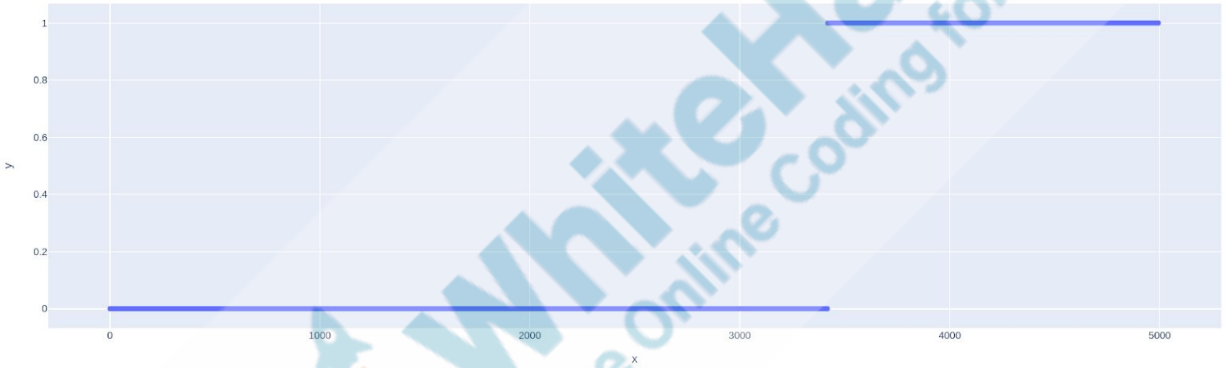
```
#Uploading the csv
from google.colab import files
data_to_load = files.upload()
```

```
Choose Files data2.csv
• data2.csv(text/csv) - 97824 bytes, last modified: 04/08/2020 - 100% done
Saving data2.csv to data2.csv
```

	Teacher helps student plot the data on scatter plot Code for reference:- **import pandas as pd** **import plotly.express as px** **df = pd.read_csv("data2.csv")** **temperature_list = df["Temperature"].tolist()** **melted_list = df["Melted"].tolist()** **fig = px.scatter(x=temperature_list,**	Student plots the data on the scatter plot

	y=melted_list) **fig.show()**	

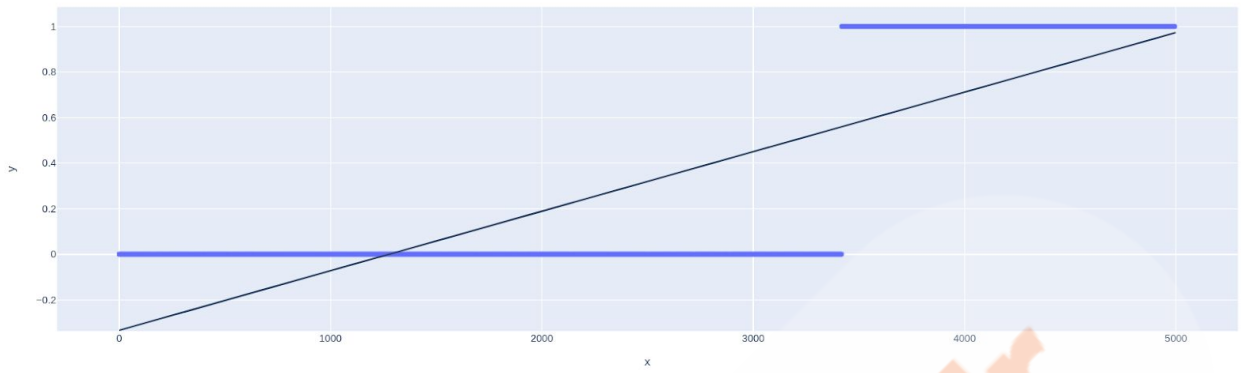
```
import pandas as pd
import plotly.express as px

df = pd.read_csv("data2.csv")

temperature_list = df["Temperature"].tolist()
melted_list = df["Melted"].tolist()

fig = px.scatter(x=temperature_list, y=melted_list)
fig.show()
```



	Teacher helps student to add the line of regression on the plot Code for reference:- **import numpy as np** **temperature_array = np.array(temperature_list)** **melted_array = np.array(melted_list)** **#Slope and intercept using pre-built function of Numpy** **m, c =**	Student adds the line of regression on the plot

```
np.polyfit(temperature_array,
melted_array, 1)

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

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

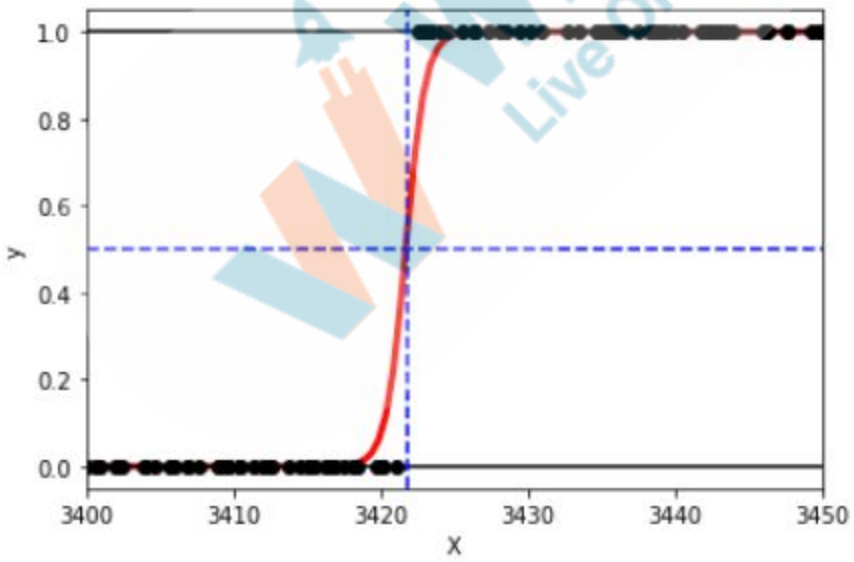
```
import numpy as np
temperature_array = np.array(temperature_list)
melted_array = np.array(melted_list)

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

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

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



	Teacher helps the student to plot the graph using the sigmoid function. And use hit and trial method on the X_test to find the value which lies on that line at the intersection. Code for reference:- **import matplotlib.pyplot as plt** **from sklearn.linear_model import LogisticRegression** **X = np.reshape(temperature_list, (len(temperature_list), 1))** **Y = np.reshape(melted_list, (len(melted_list), 1))** **lr = LogisticRegression()** **lr.fit(X, Y)** **plt.figure()** **plt.scatter(X.ravel(), Y, color='black', zorder=20)** **def model(x):** **return 1 / (1 + np.exp(-x))**	Using the sigmoid function student plots the graph and uses the hit and trial to change the values of X_test to find the values that lies on the line.

	#Using the line formula X_test = np.linspace(0, 5000, 10000) melting_chances = model(X_test * lr.coef_ + lr.intercept_).ravel() plt.plot(X_test, melting_chances, color='red', linewidth=3) plt.axhline(y=0, color='k', linestyle='-') plt.axhline(y=1, color='k', linestyle='-') plt.axhline(y=0.5, color='b', linestyle='--') plt.axvline(x=X_test[6843], color='b', linestyle='--') plt.ylabel('y') plt.xlabel('X') plt.xlim(3400, 3450) plt.show()	

```
] import matplotlib.pyplot as plt
  from sklearn.linear_model import LogisticRegression

  X = np.reshape(temperature_list, (len(temperature_list), 1))
  Y = np.reshape(melted_list, (len(melted_list), 1))

  lr = LogisticRegression()
  lr.fit(X, Y)

  plt.figure()
  plt.scatter(X.ravel(), Y, color='black', zorder=20)

  def model(x):
    return 1 / (1 + np.exp(-x))

  #Using the line formula
  X_test = np.linspace(0, 5000, 10000)
  melting_chances = model(X_test * lr.coef_ + lr.intercept_).ravel()

  plt.plot(X_test, melting_chances, color='red', linewidth=3)
  plt.axhline(y=0, color='k', linestyle='-')
  plt.axhline(y=1, color='k', linestyle='-')
  plt.axhline(y=0.5, color='b', linestyle='--')

  #do hit and trial by changing the vlaue of X_test here.
  plt.axvline(x=X_test[6843], color='b', linestyle='--')

  plt.ylabel('y')
  plt.xlabel('X')
  plt.xlim(3400, 3450)
  plt.show()
```



	Teacher helps student to write the prediction algorithm.	Using the values of slope and the intercept student codes to write a prediction algorithm with multiple cases to predict if the tungsten is melted or not depending on the temperature provided by the user.

```
temp = float(input("Enter the temperature here:- "))
chances = model(temp * lr.coef_ + lr.intercept_).ravel()[0]
if chances <= 0.01:
  print("Tungsten will not be melted")
elif chances >= 1:
  print("Tungsten will be melted")
elif chances < 0.5:
  print("Tungsten might not get melted")
else:
  print("Tungsten might get melted")
```

```
Enter the temperature here:- 60
Tungsten will not be melted
```

Teacher Guides Student to Stop Screen Share

FEEDBACK

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

Step 4: Wrap-Up (5 min)	Awesome work! Let's quickly revise what we did?	ESR: we saw a vague data which had values of binary. We learned about the sigmoid function .And used it to find the proper values of slope and intercept. We wrote the prediction

		algorithm based on the previous found data.
	Amazing!. Let's say if we want to buy a mobile phone what are the factors that we keep in mind which will effect the buying ?	ESR:- -We check if we have the budget -We check if the model we want is available or not -We check if the phone has a certain specifications we want. etc
	So as we know there are multiple factors that have a effect. This is an example of multi variable regression.	
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?	*The student uses the markdown editor to write her/his reflection in a reflection journal.*

	• What aspects of the class helped me? What did I find difficult?	

Activity	Activity Name	Links
Teacher Activity 1	Scores data	https://raw.githubusercontent.com/whitehatjr/datasets/master/PRO-C115/data.csv
Teacher Activity 2	Google Colab notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true
Teacher Activity 3	Teacher reference code	https://colab.research.google.com/drive/1NY9a1BGkFhZcH5bBgq3xAh0YSjsM3dEZ?usp=sharing
Teacher Activity 4	Teacher reference code 2	https://colab.research.google.com/drive/1BfKYPpTRWsotYOMKBx9lRS4rZJ5rZglD?usp=sharing
Student Activity 1	Tungsten data	https://raw.githubusercontent.com/whitehatjr/datasets/master/PRO-C115/data2.csv
Student Activity 2	Google colab notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true