Topic	Multilinear Logistic regression	
Class Description	**Students learn the concept of multilinear logistic regression where the outcome (yes or no) is dependent on two variables.** **Students plot the data visually and use a multilinear regression algorithm to predict the outcome of future events.**	
Class	**C116**	
Class time	**45 mins**	
Goal	• Plot the data visually where the outcome (yes or no) is dependent on the two variables • Use the machine learning libraries to create a prediction model	

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

- **Introduction to multilinear logistic regression**

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 previous class.	**ESR:** - We learned about the sigmoid function. - We learned to plot the graph for variables which have only two types of outcomes and are dependent on other variables.
	Remember at the end of the last class, we discussed a problem which people working in sales often ask themselves. On what factors do people decide if they want to buy something? For example - on what factors would a person decide if they want to buy an iPhone? Can you think about what these factors can be?	**ESR:** - Having enough money - Urgency to buy the phone - Specification of the device - Etc.
	Yes and today we'll see how multiple variables affect the results. And create a prediction model to predict the results. We will use the same logistic regression model to do so. Sounds exciting?	**ESR:** varied
	Let's get started.	
Teacher Initiates Screen Share		

<table>
<tr><td colspan="3"><u>CHALLENGE</u>
• Learn about the machine learning libraries
• Use the machine learning libraries for prediction</td></tr>
<tr><td>Step 2:
Teacher-led Activity
(15 min)</td><td>Teacher opens the google Colab notebook from Teacher Activity 1. Teacher downloads the data from Teacher Activity 2.</td><td>-</td></tr>
<tr><td></td><td>Teacher uploads the data in the Colab notebook.
Here, we have data of the people - their age, their salary and whether they decided to buy an iPhone or not. Now that we have uploaded the data let's see how the data looks visually if we plot the salary of people and whether they decided to buy an iphone or not.

Code for reference:-
import pandas as pd
import plotly.express as px

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

salary =
df["EstimatedSalary"].tolist()
purchased =
df["Purchased"].tolist()

print(len(salary))

fig = px.scatter(x=salary,
y=purchased)
fig.show()</td><td>The student helps the teacher upload the data. The student helps the teacher plot the graph.</td></tr>
</table>

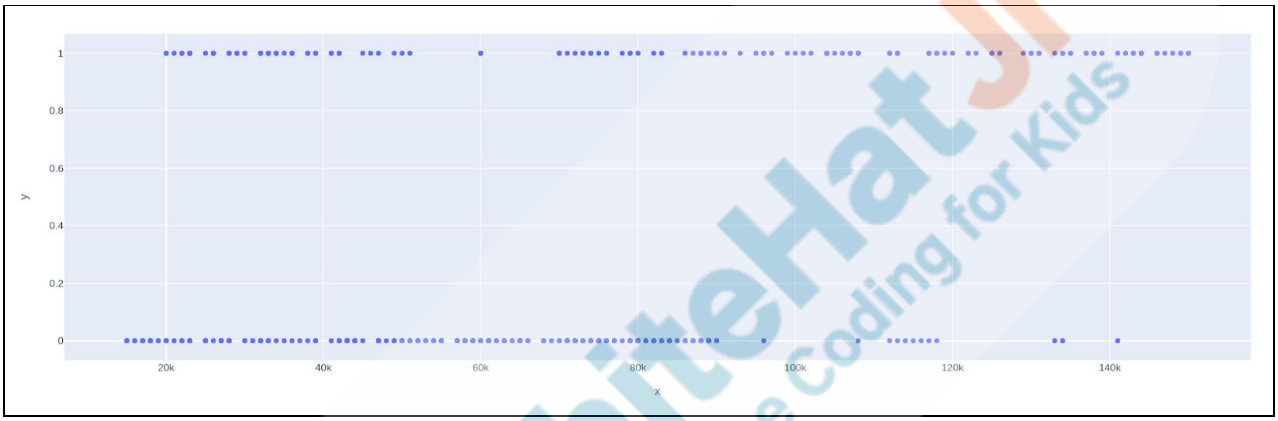
```
] import pandas as pd
  import plotly.express as px

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

  salary = df["EstimatedSalary"].tolist()
  purchased = df["Purchased"].tolist()

  print(len(salary))

  fig = px.scatter(x=salary, y=purchased)
  fig.show()
```



	How does the graph look?	**ESR:** We can see two different lines formed on the graph.
	Do you see any connection between salary and the decision to purchase?	We can see that people who have higher income are more likely to buy an iPhone but there are also significant people with high income who decide not to buy an iPhone.
	Yes, can we predict who will buy an iPhone based on this data?	**ESR:** No. It will be difficult.

Yes , there might be another variable which might be affecting the decision to purchase an iPhone.

Let us try to see if age along with salary can better predict who makes a decision to buy an iPhone.

Let's visualize the data with all three variables on the scatter plot - age and salary can be axes. We can represent red dots for people who refuse to buy an iPhone while green dots represent people who decide to buy the iPhone.

Teacher codes to visualize the data with all three variables on a scatter plot.

Code for reference:-

```
import plotly.graph_objects as go

salaries = df["EstimatedSalary"].tolist()
ages = df["Age"].tolist()

purchased = df["Purchased"].tolist()
colors=[]
for data in purchased:
 if data == 1:
   colors.append("green")
 else:
   colors.append("red")
```

```
fig = go.Figure(data=go.Scatter(
    x=salaries,
    y=ages,
    mode='markers',
    marker=dict(color=colors)
))
fig.show()
```

The green dots represent the people who bought the phone and red dots represent the people who didn't buy the phone.

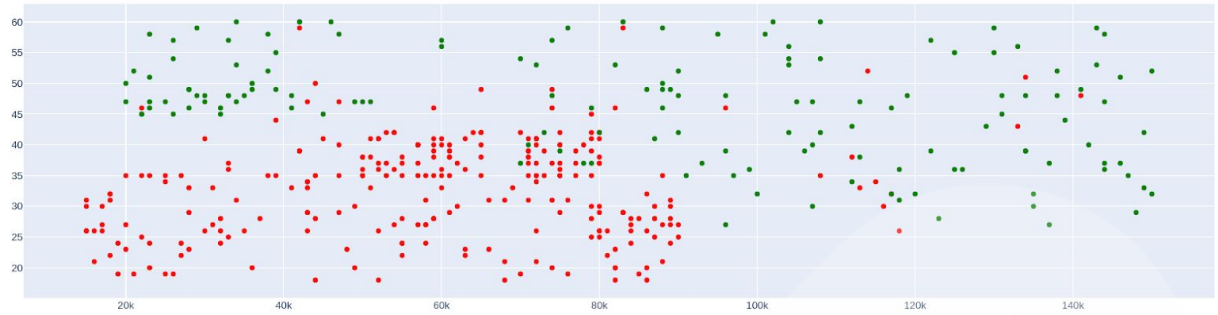
```
import plotly.graph_objects as go

salaries = df["EstimatedSalary"].tolist()
ages = df["Age"].tolist()

purchased = df["Purchased"].tolist()
colors=[]
for data in purchased:
  if data == 1:
    colors.append("green")
  else:
    colors.append("red")



fig = go.Figure(data=go.Scatter(
    x=salaries,
    y=ages,
    mode='markers',
    marker=dict(color=colors)
))
fig.show()
```



	Now what do you see? Yes we can clearly see that there is some relationship between age, salary and whether people decide to buy an iPhone or not. Since the outcome is still binary (whether people decide to buy the phone or not), we can use logistic regression techniques we learned earlier to calculate the chance for someone buying an iPhone. We will be using the machine learning libraries for logistic regression to	**ESR:** We can see that the plot is divided into two parts. Red dots (people who didn't purchase the phone) are clustered together. Green dots (people who decided to purchase the phone) are clustered together.

	build a model to predict if the person will buy an iPhone or not. To do that we'll divide the data into two parts. We'll use the first part to train this model to predict if a person will buy the phone or not. And we will use the second part to test our model. Dividing the data and using a part of it to train the model on it and then using the other part of the data to test the model is a general practice used by data scientists.	
	We will consider the age of the person and the salary to determine if they will purchase the product or not. Let's create these two data frame, salary and purchases. Code for reference:- **#Taking together the Age and Salary of the person** **factors = df[["EstimatedSalary", "Age"]]** **#Purchases made** **purchases = df["Purchased"]**	-

```
#Taking together Age and Salary of the person
factors = df[["EstimatedSalary", "Age"]]

#Purchases made
purchases = df["Purchased"]
```

	Out of the data that we have, we will use 75% data for training and then we will test our model on the remaining 25% percent of the data to test and determine the accuracy of our model. Let's start by splitting the data. To split the data we'll use the **train_test_split()** function from the **sklearn.model_selection** library. **train_test_split()** function takes the data to split, test_size and random_state as parameters. Code for reference:- **from sklearn.model_selection import train_test_split** **salary_train, salary_test, purchase_train, purchase_test = train_test_split(factors, purchases, test_size = 0.25, random_state = 0)**	*The student observes and learns.*

```
from sklearn.model_selection import train_test_split

salary_train, salary_test, purchase_train, purchase_test = train_test_split(factors, purchases, test_size = 0.25, random_state = 0)
```

	Now we have divided the data into 75% and 25% ratios. If we look at the data, we are considering the age and the salary together so what do we see? Yes so we have to make sure that	**ESR:** The age of the person is in years and the salary can be in INR or Dollars.

	they are using the same unit before we proceed further. For this, we are going to use sklearn's StandardScaler. This will compare values with each other, and determine a score for the values. Both the age and the salary of the person will be given a score in comparison to others. Let's see how our data changes after doing this. The StandardScaler assumes your data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1. Code for reference:- # here we are preprocessing the data to fit in our training model **from sklearn.preprocessing import StandardScaler** **sc_x = StandardScaler()** **salary_train = sc_x.fit_transform(salary_train)** **salary_test = sc_x.transform(salary_test)** **print (salary_train[0:10])** *<teacher can print and check the values>*	

```
print(salary_train[0:10])

from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()

salary_train = sc_x.fit_transform(salary_train)
salary_test = sc_x.transform(salary_test)

print (salary_train[0:10])
```

```
     EstimatedSalary  Age
250            39000   44
63            120000   32
312            50000   38
159           135000   32
283            21000   52
340           104000   53
81             42000   39
349            61000   38
153            50000   36
295            63000   36
[[-0.88670699  0.58164944]
 [ 1.46173768 -0.60673761]
 [-0.5677824  -0.01254409]
 [ 1.89663484 -0.60673761]
 [-1.40858358  1.37390747]
 [ 0.99784738  1.47293972]
 [-0.79972756  0.08648817]
 [-0.24885782 -0.01254409]
 [-0.5677824  -0.21060859]
 [-0.19087153 -0.21060859]]
```

	What do we see here? Now, we are sure that each and every feature will contribute equally to the decision making of our machine learning model, where we want to predict if the user will buy the product or not.	**ESR:** We can see that both the age and the salary of the person are given points.

	Now, let's train a logistic regression model on the training data that we separated out earlier. For this, we will use sklearn's pre-built class LogisticRegression. Code for reference: **from sklearn.linear_model import LogisticRegression** **classifier = LogisticRegression(random_state = 0)** **classifier.fit(salary_train, purchase_train)** Here, the **random_state = 0** means that we are providing the model with the Training data. If the value would have been 1, it would mean that we are providing the testing data. Now that our model for Logistic Regression is trained, what's left to do?	**ESR:** We have to test it now.

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)
classifier.fit(salary_train, purchase_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

	Yes let's test it now. Code for reference:- **purchase_pred = classifier.predict(salary_test)** **from sklearn.metrics import accuracy_score** **print ("Accuracy : ", accuracy_score(purchase_test, purchase_pred))** What accuracy do we see?	**ESR:** We see the accuracy of 0.89.

```
] purchase_pred = classifier.predict(salary_test)

  from sklearn.metrics import accuracy_score
  print ("Accuracy : ", accuracy_score(purchase_test, purchase_pred))

  Accuracy :  0.89
```

	An accuracy of 1 would have been perfectly accurate - 100% likely to buy, 0.89 is an excellent accuracy - 89% likely to buy. Remember we divided the data into 2 parts, ones who purchased the iPhone and ones who didn't purchase it.	**ESR:** Varied.
	Yes! Now the final thing that is left to do is to test the model and see if the user will purchase the iPhone or not. Code for reference:-	*The student helps the teacher test the model.*

	user_age = int(input("Enter age of the customer -> ")) **user_salary = int(input("Enter the salary of the customer -> "))** **#we are transforming the values here** **user_test = sc_x.transform([[user_salary, user_age]])** **#predicting the values using the function.** **user_purchase_pred = classifier.predict(user_test)** **if user_purchase_pred[0] == 1:** **print("This customer may purchase the product!")** **else:** **print("This customer may not purchase the product!")** *The teacher tests the code with the values from the given data set to check if the module works or not.*	

```
user_age = int(input("Enter age of the customer -> "))
user_salary = int(input("Enter the salary of the customer -> "))

user_test = sc_x.transform([[user_salary, user_age]])

user_purchase_pred = classifier.predict(user_test)

if user_purchase_pred[0] == 1:
  print("This customer may purchase the product!")
else:
  print("This customer may not purchase the product!")
```

```
Enter age of the customer -> 23
Enter the salary of the customer -> 120000
This customer may not purchase the product!
```

	So as we see the model works fine. Now I have a challenge for you, can you create your own model for another such data	
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	

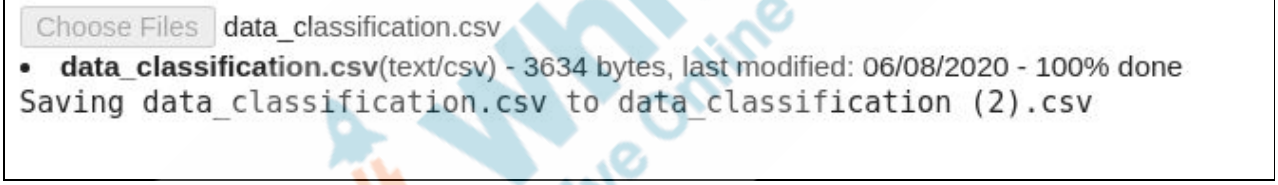
- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

ACTIVITY

- **Creating a prediction model using machine learning libraries.**

<table>
<tr><td>Step 3:
Student-Led Activity
(15 min)</td><td>Teacher helps the student open Colab notebook.
Teacher helps the student download data.</td><td>Student opens the new Colab notebook from Student activity 1.
Student downloads data from Student Activity 2.</td></tr>
<tr><td></td><td>Teacher helps the student with the code.</td><td>Student Codes to upload the data on the Colab notebook.</td></tr>
<tr><td colspan="3">

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



</td></tr>
<tr><td></td><td>Teacher helps the student with the code.</td><td>The student plots the data on the graph.</td></tr>
</table>

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

df = pd.read_csv("data_classification (2).csv")

hours_slept = df["Hours_Slept"].tolist()
hours_studied = df["Hours_studied"].tolist()



fig = px.scatter(x=hours_slept, y=hours_studied)
fig.show()
```

	Teacher helps the student with the code.	*Student codes to plot the scatter plot with hours_slept, hours_studied and results variables line.*

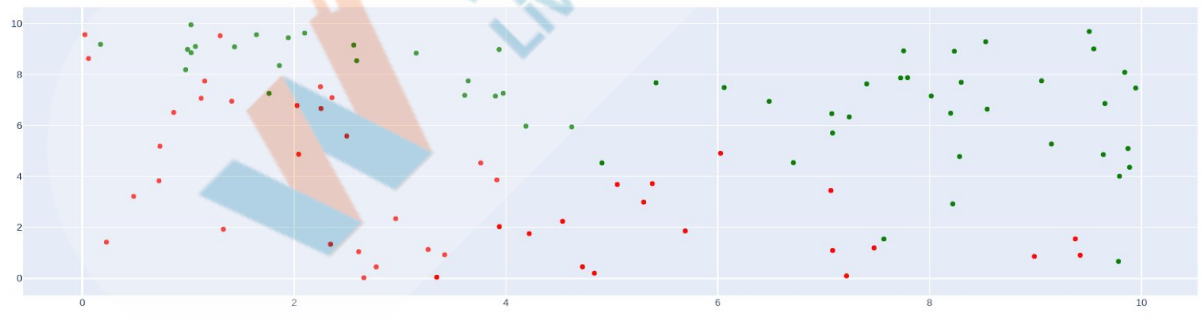
```
5]  import plotly.graph_objects as go

    hours_slept = df["Hours_Slept"].tolist()
    hours_studied = df["Hours_studied"].tolist()

    results = df["results"].tolist()
    colors=[]
    for data in results:
      if data == 1:
        colors.append("green")
      else:
        colors.append("red")



    fig = go.Figure(data=go.Scatter(
        x=hours_studied,
        y=hours_slept,
        mode='markers',
        marker=dict(color=colors)
    ))
    fig.show()
```



	Teacher helps the student with the code.	*Student codes to divide the data in between hours and results.*

```
] #hours studied and slept of the person
  hours = df[["Hours_studied", "Hours_Slept"]]

  #results
  results = df["results"]
```

	Now we have to split the data for training and testing. *Teacher helps the student with the code.*	*Using the train_test_split function of sklearn library the student splits the data into 75 and 25 % for training and testing.*

```
15] from sklearn.model_selection import train_test_split

   hours_train, hours_test, results_train, results_test = train_test_split(hours, results, test_size = 0.25, random_state = 0)
   print(hours_train)

    Hours_studied  Hours_Slept
48       7.754217     8.922706
6        0.993438     8.978216
99       3.937267     2.030708
82       5.299210     2.991911
76       7.084377     1.091472
..            ...          ...
96       7.405351     7.630637
67       1.411293     6.949705
64       2.248929     7.517309
47       7.726383     7.863850
44       7.214513     0.098053
```

	We don't need to compare the values as both the values are the same.	-
	We have the data ready, we need to train our model. *Teacher helps the student to code to train their model.*	*Student codes to train the logistic regression model.*

```
] from sklearn.linear_model import LogisticRegression

  classifier = LogisticRegression(random_state = 0)
  classifier.fit(hours_train, results_train)

  LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                     intercept_scaling=1, l1_ratio=None, max_iter=100,
                     multi_class='auto', n_jobs=None, penalty='l2',
                     random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                     warm_start=False)
```

	Now we need to test the model for accuracy. *Teacher guides the student to code for accuracy.*	*Student codes to test the model for accuracy.*

```
] results_pred = classifier.predict(hours_test)

 from sklearn.metrics import accuracy_score
 print ("Accuracy : ", accuracy_score(results_test, results_pred))

 Accuracy :  0.92
```

	Teacher helps the student test the model.	*The student tests the model.*

```
user_hours_studied = int(input("Enter hours studied -> "))
user_hours_slept = int(input("Enter hours slept -> "))

user_test = sc_x.transform([[user_hours_studied, user_hours_slept]])

user_result_pred = classifier.predict(user_test)

if user_result_pred[0] == 1:
  print("This user may pass!")
else:
  print("This user may not pass!")

Enter hours studied -> 8
Enter hours slept -> 8
This user may pass!
```

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)**	So, in this class we learned about multilinear regression. How was your experience?	**ESR:** varied
	Amazing work in this class	-
Project Overview	**Multi linear Logistics Regression** **Goal of the Project:** In this project you will apply what you learned in the class and create your own prediction model. **Story:** I am very excited to see your project solution and I know you will do really well. Bye Bye!	
Teacher Clicks		

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	Google Colab notebook	https://colab.research.google.com/#create=true
Teacher Activity 2	Iphone purchase data	https://raw.githubusercontent.com/whitehatjr/datasets/master/pro-c116/logistics_data.csv
Teacher Activity 3	Teacher reference 1 Teacher side	https://colab.research.google.com/drive/10l3xV_pncxqV84AnSM_Ed5dYFcXLszTn?usp=sharing
Teacher Activity 4	Teacher reference 2 Student side	https://colab.research.google.com/drive/19sqtaqmUbTpaQYp7aSwrKdl8jewJkjN7?usp=sharing
Student Activity 1	Google Colab notebook	https://colab.research.google.com/#create=true

Student Activity 2	Hours studying and sleeping data	https://raw.githubusercontent.com/whitehatjr/datasets/master/pro-c116/data_classification.csv