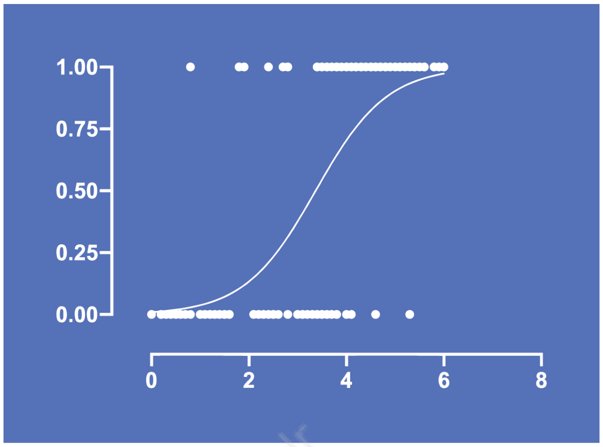
PRO-C116

Multilinear Logistics Regression



What is our GOAL for this MODULE?

The goal of this module is to learn about multilinear logistics regression and create a prediction model using the machine learning libraries.

What did we ACHIEVE in the class TODAY?

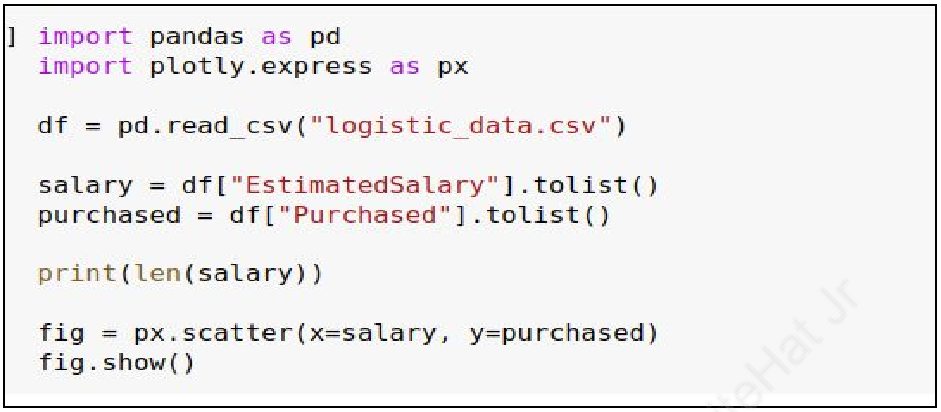
We learned about the multilinear logistics regression and using training and testing methods we created the prediction model.

Which CONCEPTS/CODING BLOCKS did we cover today?

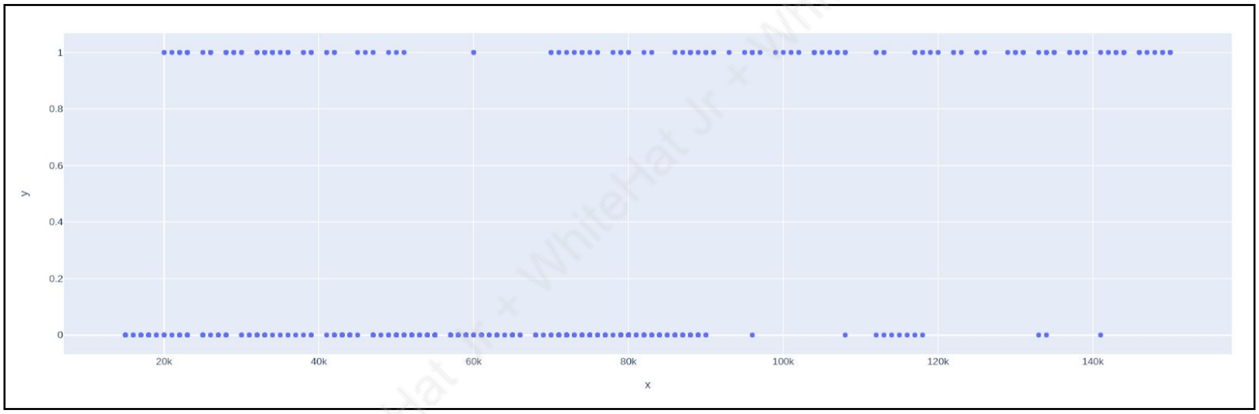
- Usage of sklearn library
- Training and testing of the prediction model

How did we DO the activities?

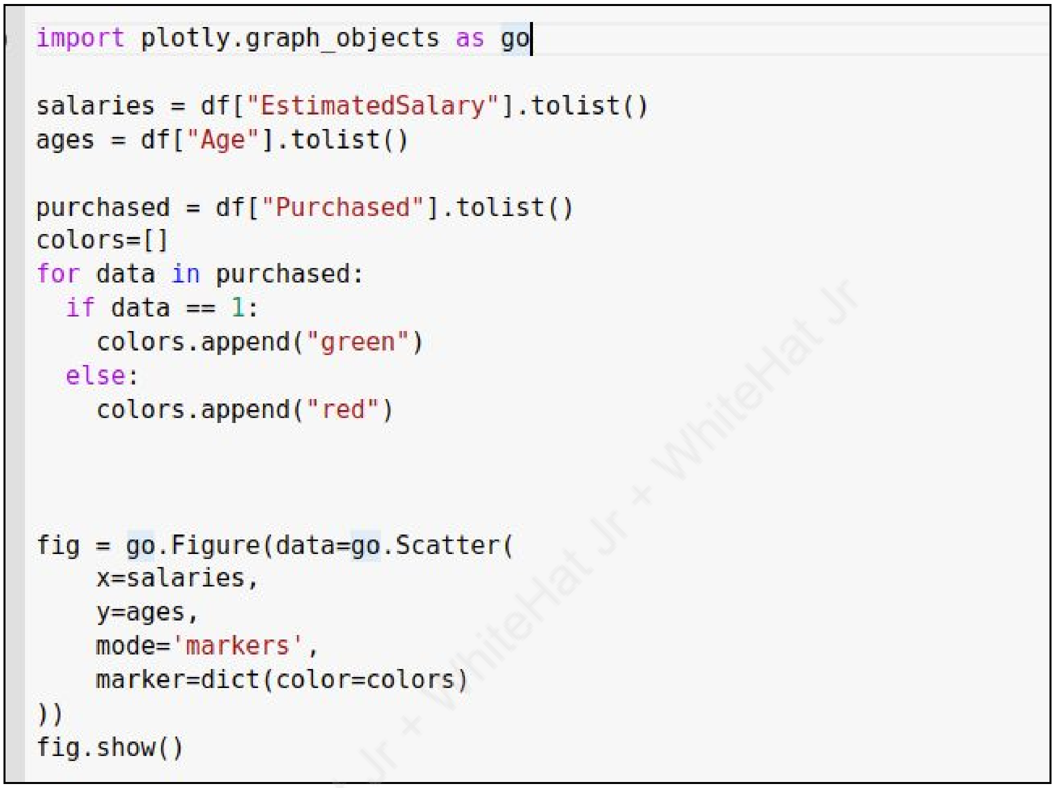
1. We uploaded the data of people who did and did not purchase the Iphone and plotted it on the scatter plot.



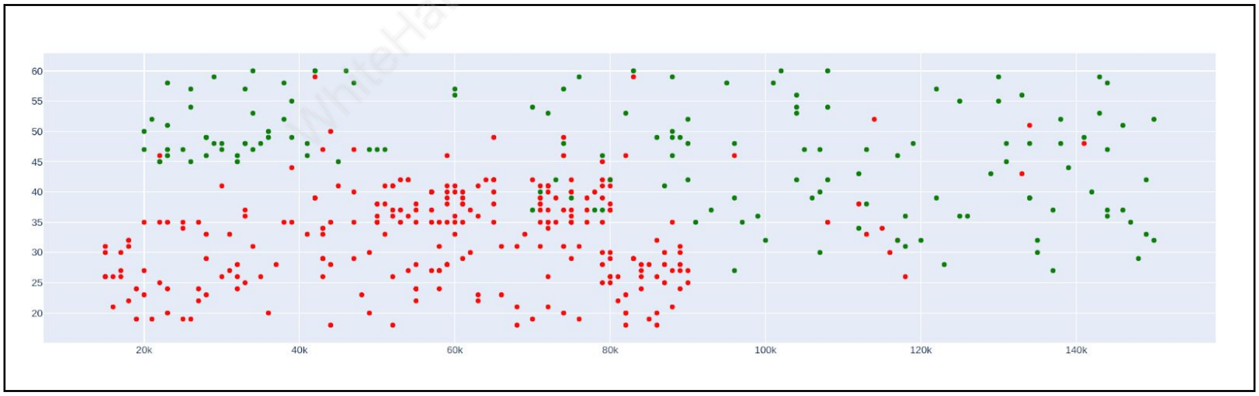
```
] import pandas as pd
  import plotly.express as px

  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()
```



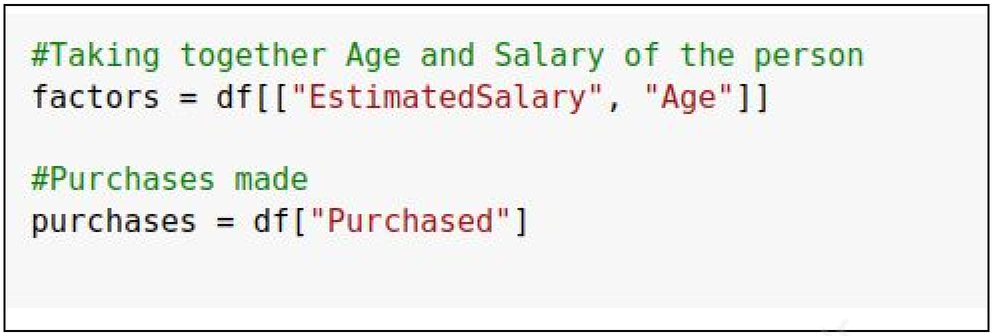
2. We then visualize the data with all three variables on the scatter plot - age and salary can be axes. We represented red dots for people who refuse to buy an iPhone while green dots represent people who decide to buy the iPhone.



```
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()
```

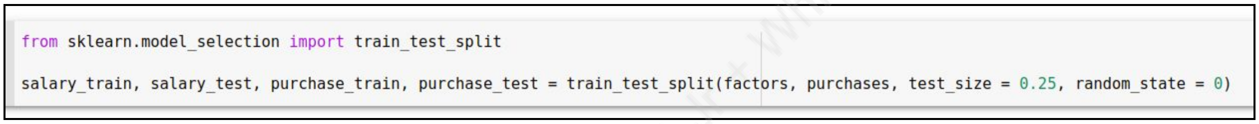


3. We divided the data into 2 parts to use the training and testing method on the prediction model.



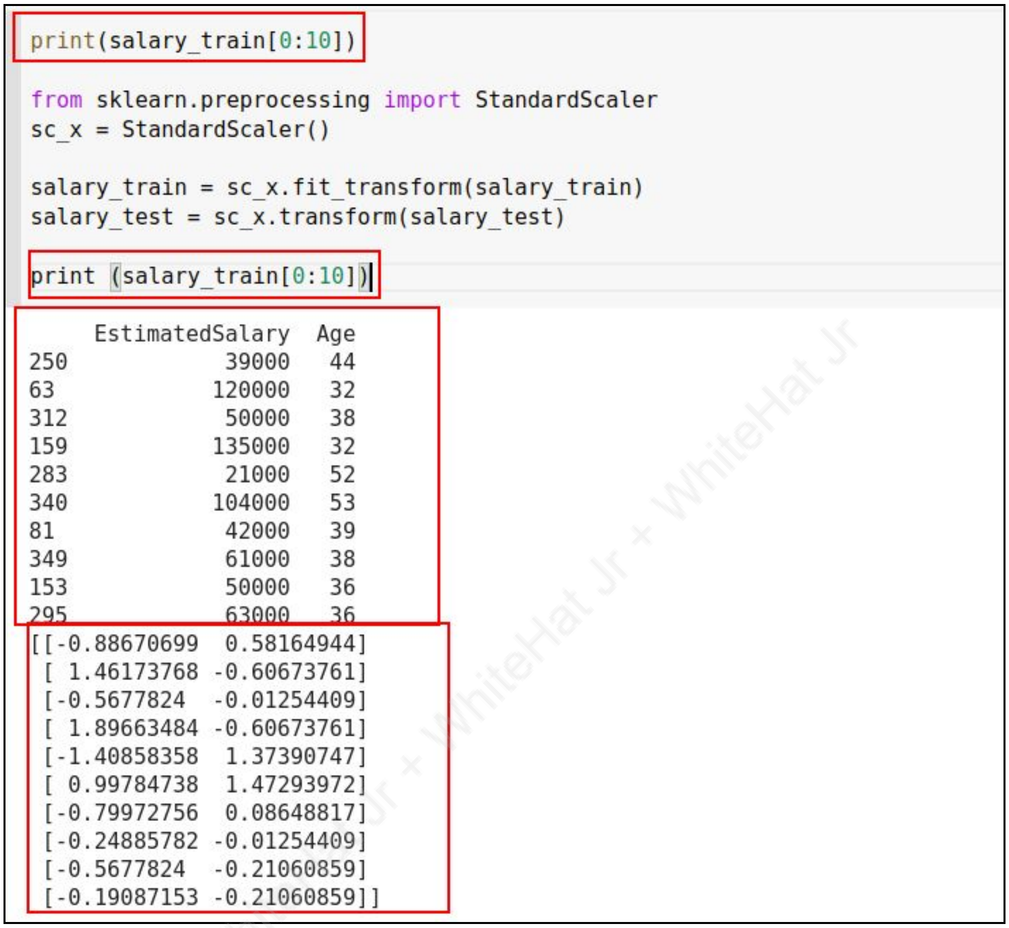
```
#Taking together Age and Salary of the person
factors = df[["EstimatedSalary", "Age"]]

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

4. We split the data into 75% and 25% ratios.



```
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)
```

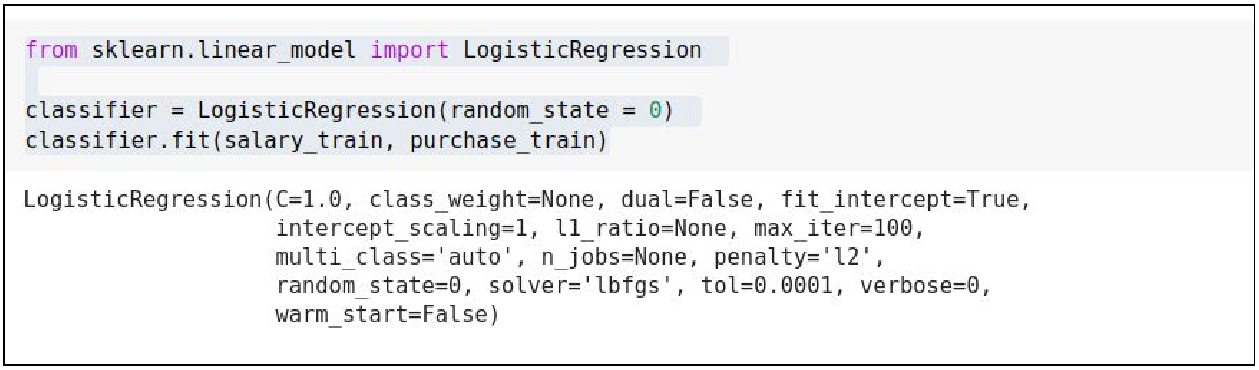
5. We then transformed the data into the scalar value using the standard.Scalar function of the sklearn library.



```
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]]
```

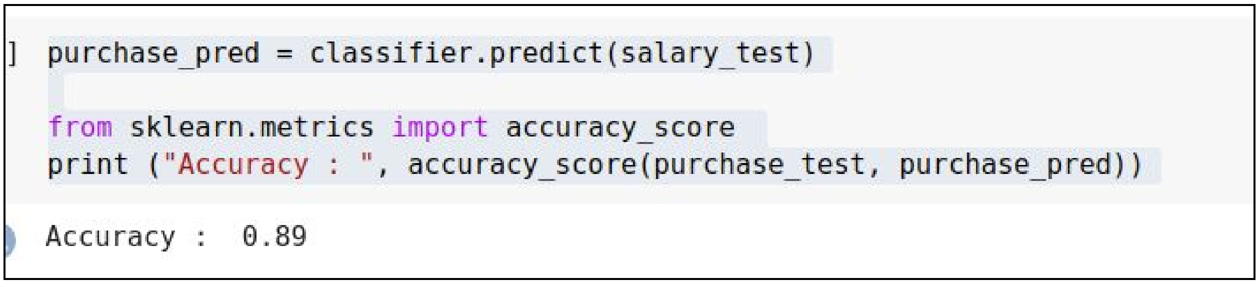
6. We then trained the model using the sklearn's pre-built class LogisticRegression.



```
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)
```

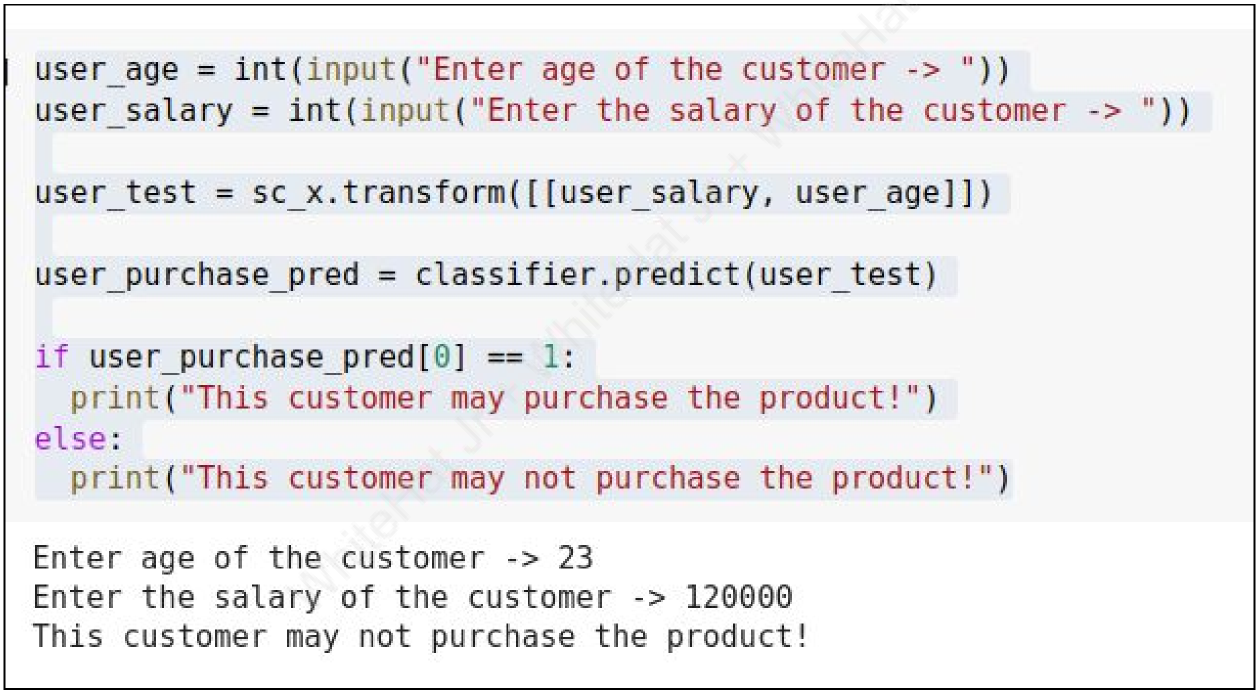
7. Then we got the prediction accuracy of the model.



```
purchase_pred = classifier.predict(salary_test)

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

```
Accuracy :  0.89
```

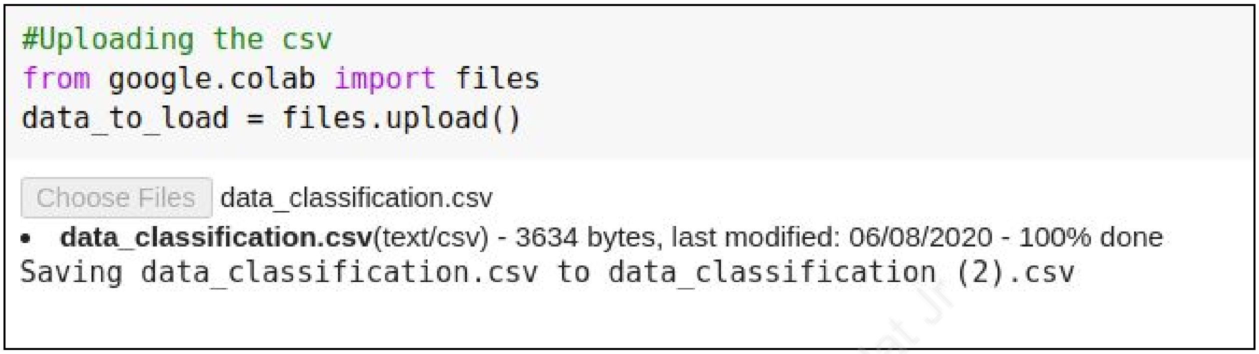
8. Then we tested the model by taking input from the user.



```
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!
```

9. We repeated the same process with another data of hours students studied and slept and their results.
    - We uploaded the data on the colab notebook.



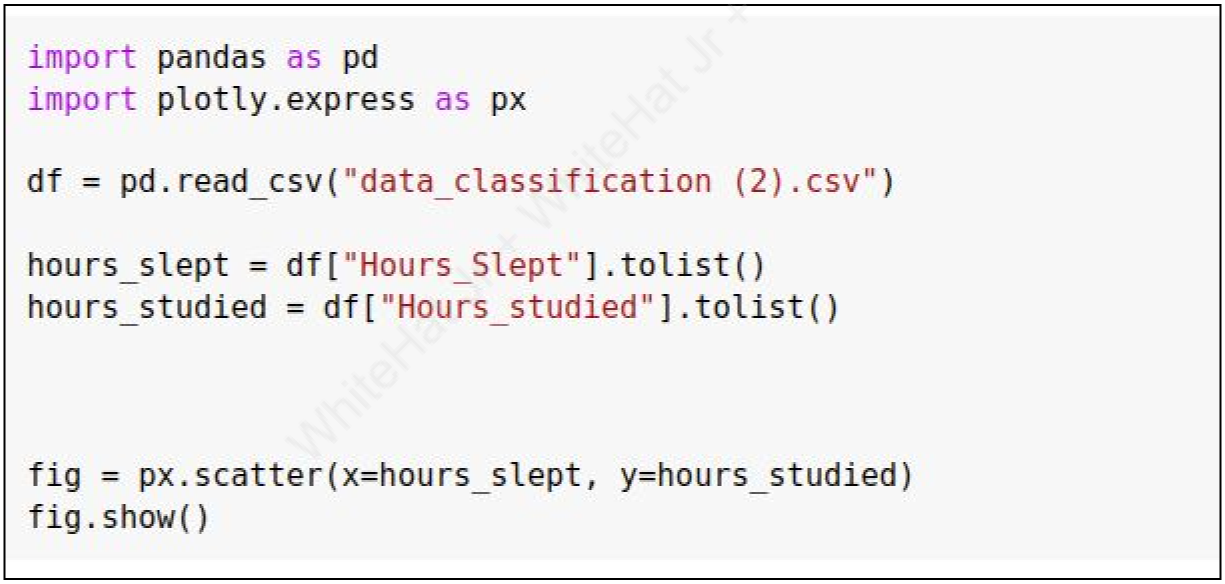
```
#Uploading the csv
from google.colab import files
data_to_load = files.upload()
```

Choose Files data_classification.csv

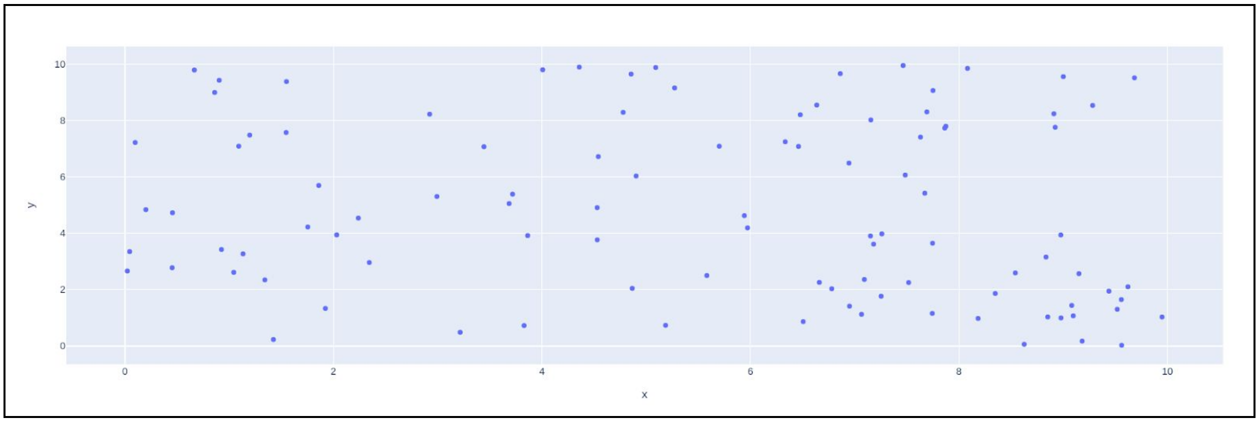
- **data_classification.csv**(text/csv) - 3634 bytes, last modified: 06/08/2020 - 100% done

```
Saving data_classification.csv to data_classification (2).csv
```

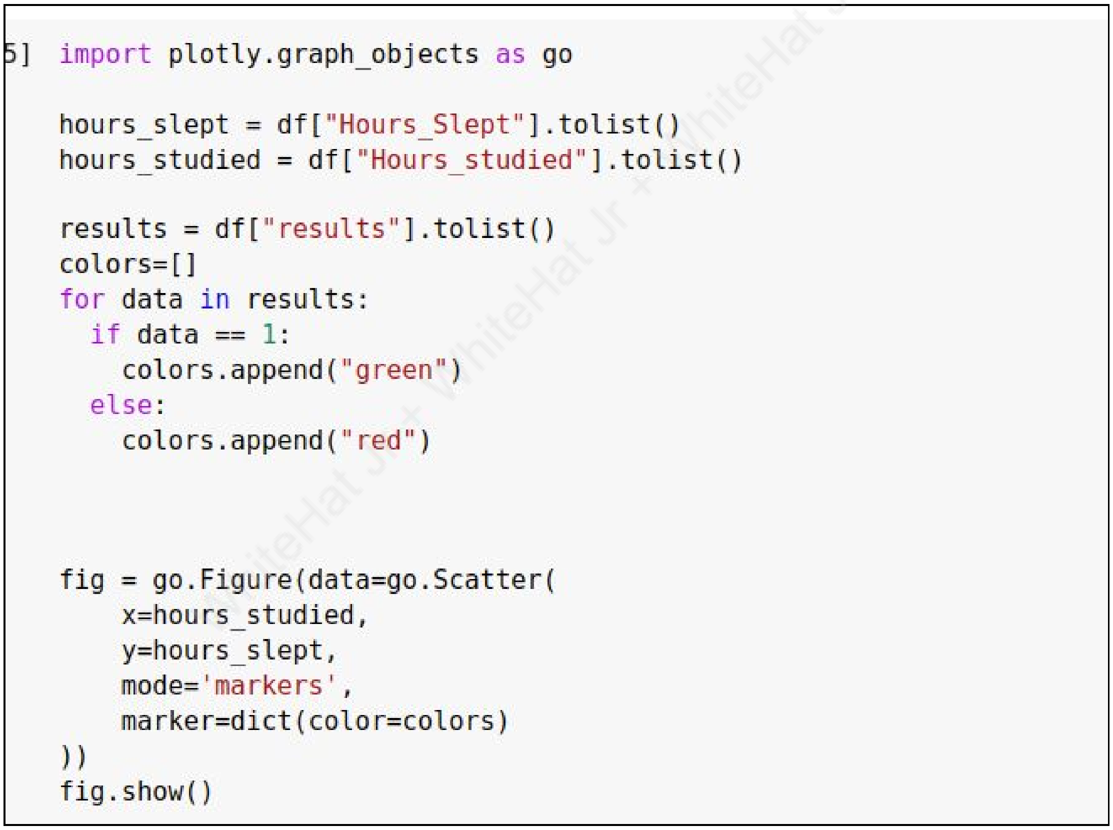
10. Plotted the data on the scatter plot.



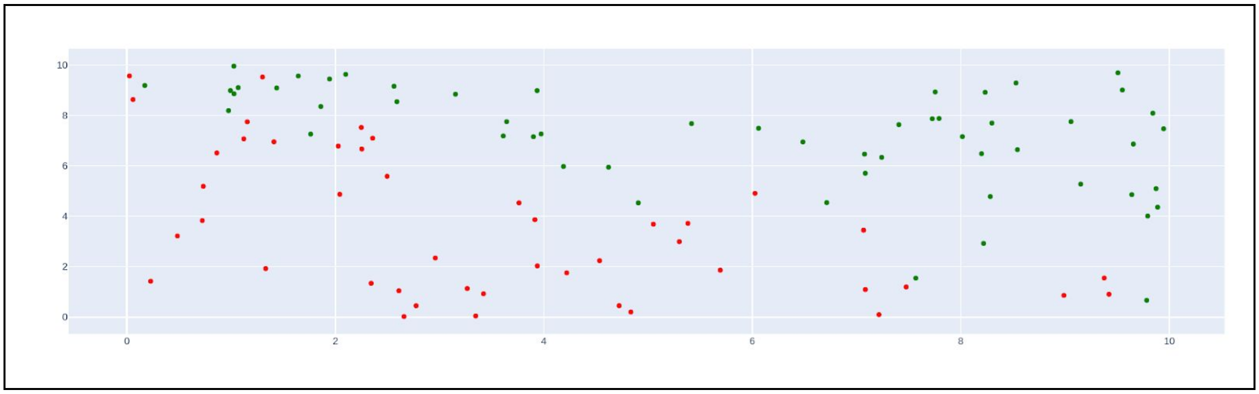
```
import pandas as pd
import plotly.express as px

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()
```



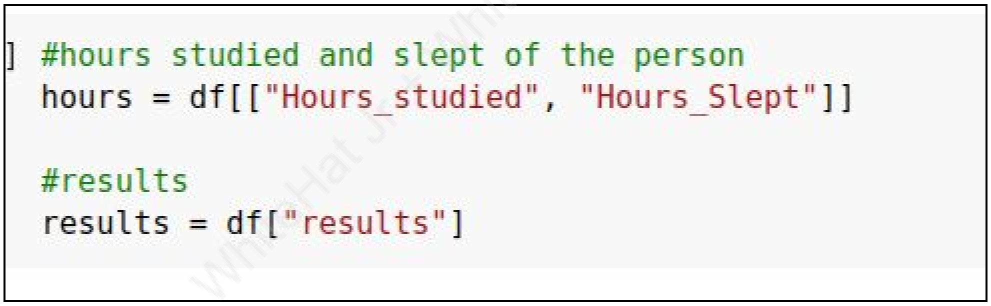
11. Then we plotted the scatter plot with all the variables.



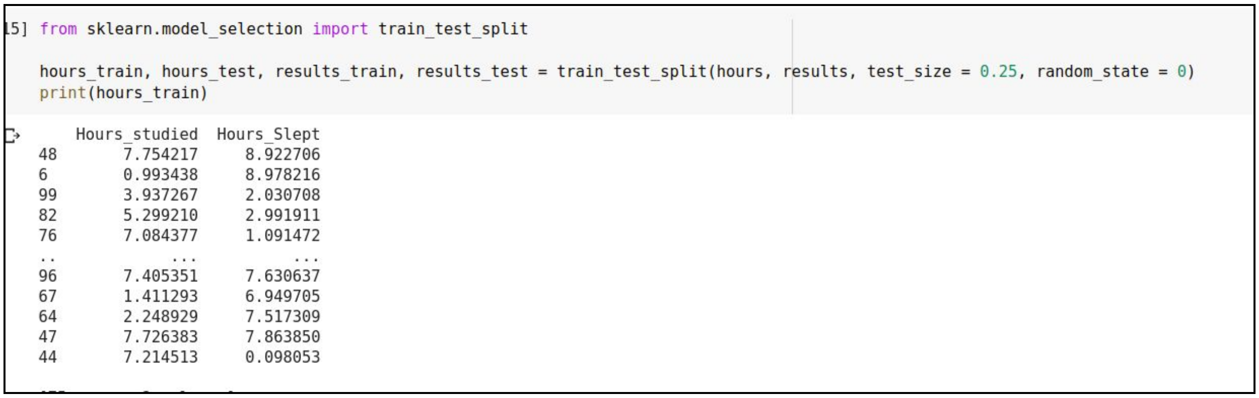
```
5]  import plotly.graph_objects as go

    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()
```



12. Then we created 2 different data frames and split them into 75% and 25% ratios for training and testing the model.



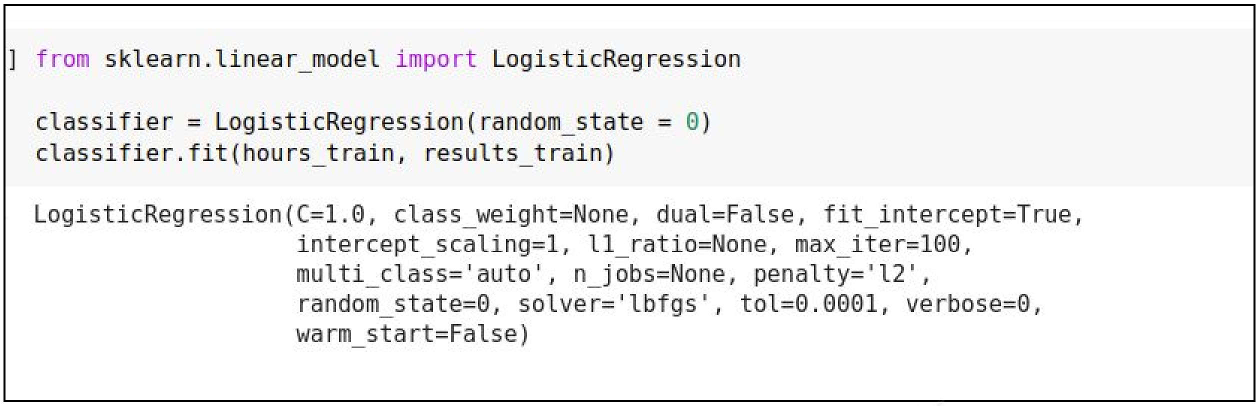
```
] #hours studied and slept of the person
  hours = df[["Hours_studied", "Hours_Slept"]]

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



```
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
```

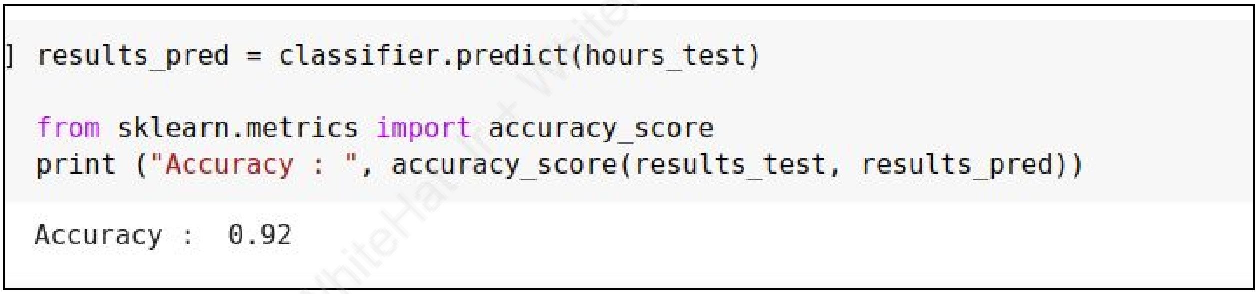
13. We didn’t need to transform the values as the values were the same.

14. Using the logistic regression class of the sklearn library we trained the module.



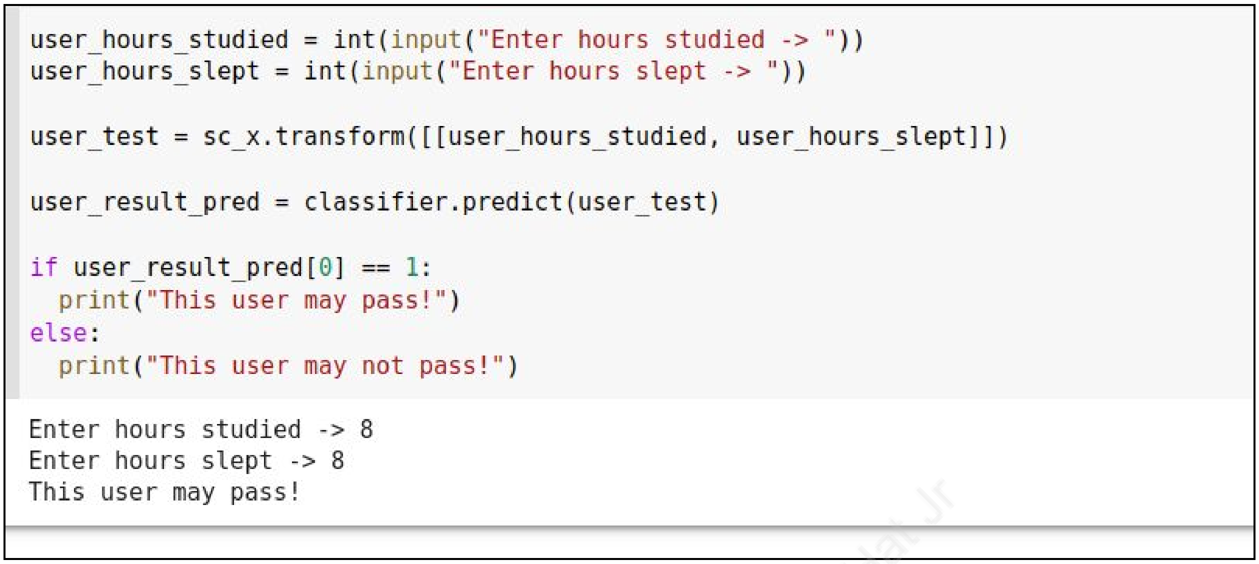
```
] from sklearn.linear_model import LogisticRegression

  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)
```

15. We check for the accuracy of the module.



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

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

  Accuracy :  0.92
```

16. Then we tested the module by taking different data from the user as input.



```
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!
```

We learned a way to write a prediction model for multilinear logistic regression.

What's NEXT?

In the next class, we will learn about the efficiency of machine learning algorithms.

EXTEND YOUR KNOWLEDGE:

Read the following blog to understand more about the multilinear logistic regression:
https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148