ANURAG UNIVERSITY

MACHINE LEARNING AND APPLICATIONS PROJECT

TOPIC: Placement predictions using logistic regression.

Team members :

i. V.Karthik (20eg112347)
ii. C.Ramakrishna (20eg112340)
iii. P.Ram Saketh (20eg112339)
iv. M.Sri Harsha(20eg112333)
v. C.Rohith(20eg112354)
vi. J.Kamal (20eg112315)

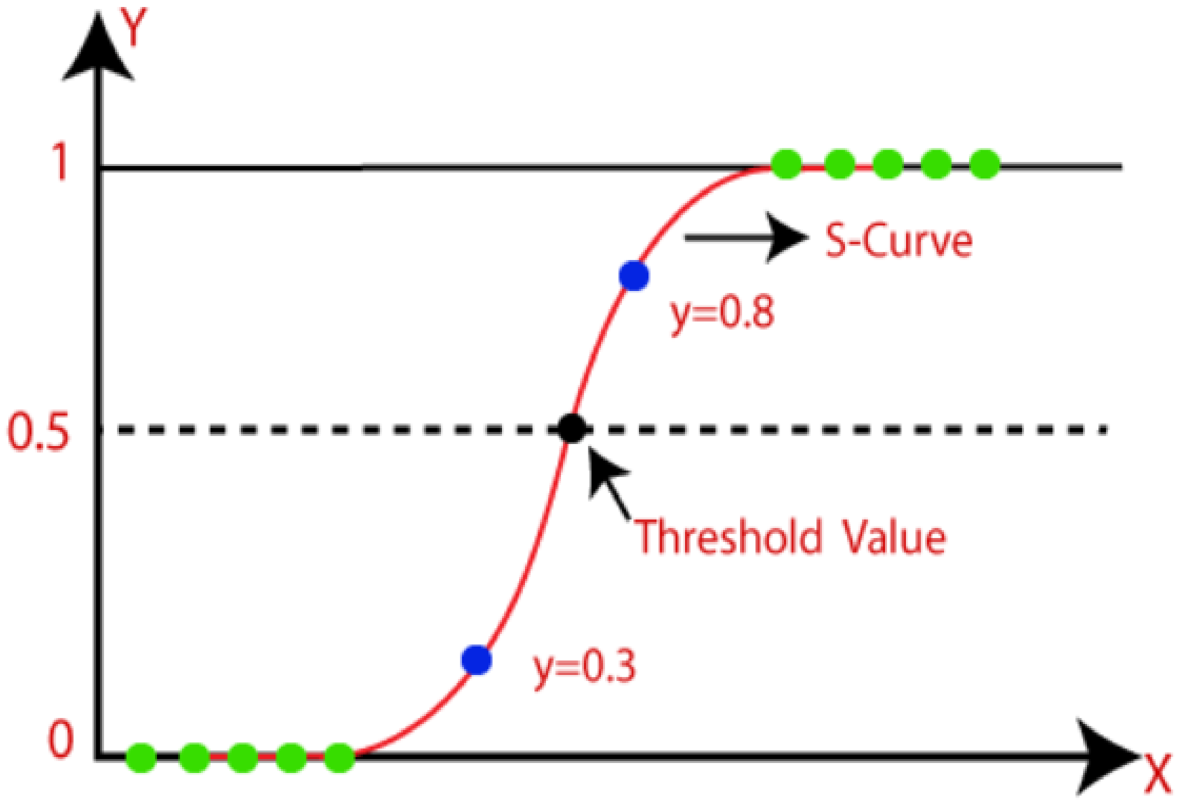
Logistic Regression:

Logistic regression, despite its name, is a classification model rather than regression model. Logistic regression is a simple and more efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. It is an extensively employed algorithm for classification in industry. The logistic regression model, like the Adaline and perceptron, is a statistical method for binary classification that can be generalized to multiclass classification. Scikit-learn has a highly optimized version of logistic regression implementation, which supports multiclass classification task.

The primary difference between linear regression and logistic regression is that logistic regression's range is bounded between 0 and 1. In addition, as opposed to linear regression, logistic regression does not require a linear relationship between inputs and output variables. This is due to applying a nonlinear log transformation to the odds ratio (will be defined shortly).

$$Logistic\ function = \frac{1}{1+e^{-x}}$$

Graphical representation:



Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y} \text{ ; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"

- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Libraries and tools used:

Python:
Python is one of the most popular, open source programming language widely adopted by machine learning community. It was designed by Guido van Rossum and was first released in 1991.
Python has libraries for data loading, visualization, statistics, natural language processing, image processing, and more.
Python has very strong libraries for advanced mathematical functionalities (NumPy), algorithms and mathematical tools (SciPy) and numerical plotting (matplotlib). Built on these libraries, there is a machine learning library named **scikit learn**, which has various classification, regression, and clustering algorithms embedded in it.

scikit-learn:
scikit-learn is an open source project, and it contains a number of state-of-the-art machine
learning algorithms, as well as comprehensive documentation about each algorithm. scikit-learn is a very popular tool, and the most prominent Python library for machine learning. It is widely used in industry and academia, and a wealth of tutorials and code snippets are available online. scikit-learn works well with a number of other scientific Python tools

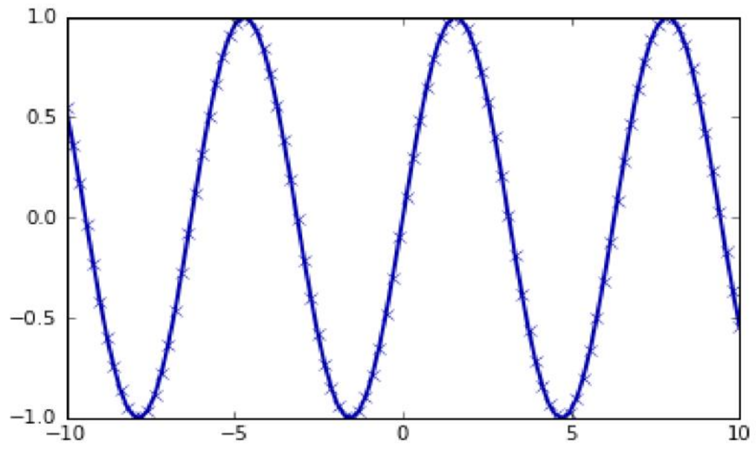
- Scikit-learn depend on two other Python packages, *NumPy* and *SciPy*. For plotting and interactive development, you should also install matplotlib, IPython, and the Jupyter Notebook.
- **Jupyter Notebook:** (browser-based interactive programming environment)

The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis and is widely used by data scientists.

NumPy:
NumPy is one of the fundamental packages for scientific computing in Python. It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations and the Fourier transform, and pseudorandom number generators.
In scikit-learn, the NumPy array is the fundamental data structure. scikit-learn takes in data in the form of NumPy arrays.
The core functionality of NumPy is the ndarray class, a multidimensional (*n*-dimensional) array. All elements of the array must be of the same type.

SciPy:
SciPy is a collection of functions for scientific computing in Python. It provides, among other functionality, advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical functions, and statistical distributions.
scikit-learn draws from SciPy’s collection of functions for implementing its algorithms. The most important part of SciPy for us is scipy.sparse: this provides *sparse matrices*, which are another representation that is used for data in scikitlearn.

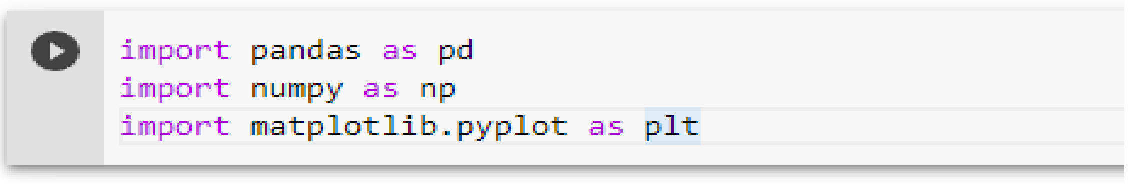
Matplotlib:
matplotlib is the primary scientific plotting library in Python. It provides functions for making publication-quality visualizations such as line charts, histograms, scatter plots, and so on.
For Example:



Pandas:
Pandas is a Python library for data wrangling and analysis. It is built around a data structure called the DataFrame that is modeled after the R DataFrame. Simply put, a pandas DataFrame is a table, similar to an Excel spreadsheet and it allows SQL-like queries and joins of tables. In contrast to NumPy, which requires that all entries in an array be of the same type, pandas allows each column to have a separate type (for example, integers, dates, floating-point numbers, and strings).

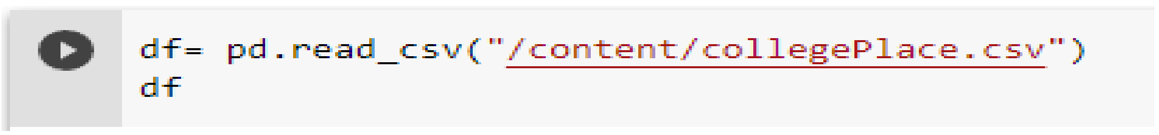
Developing the Model

Step 1: We are importing all the necessary libraries like pandas, numpy and matplotlib and sklearn.



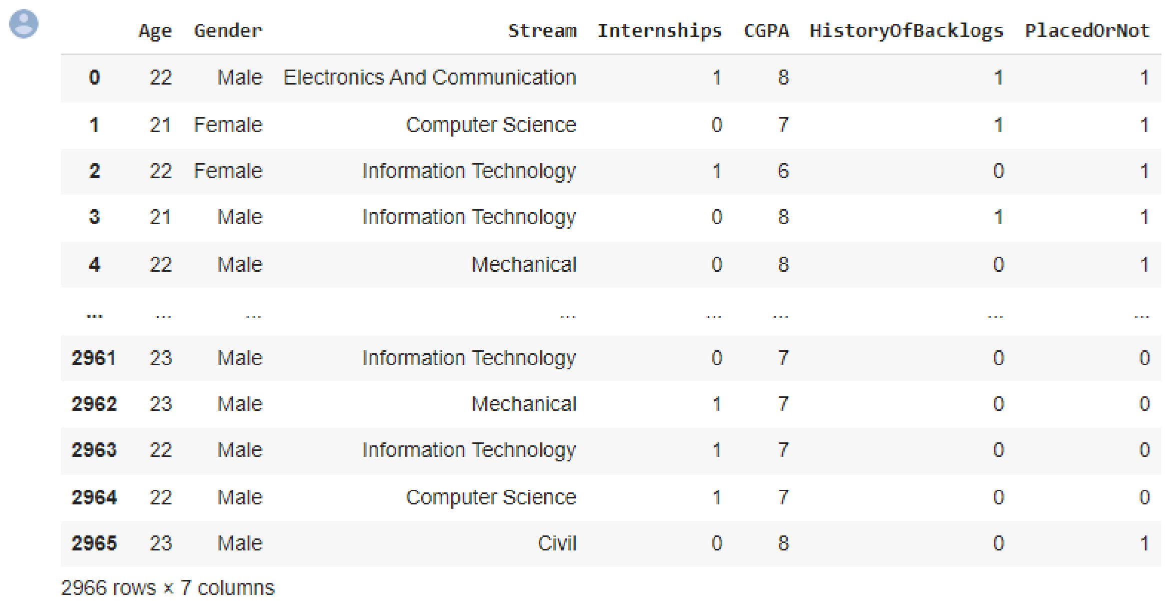
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Step 2: We are loading the dataset.



```
df= pd.read_csv("/content/collegePlace.csv")
df
```

The corresponding output for the code is:



	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	Male	Electronics And Communication	1	8	1	1
1	21	Female	Computer Science	0	7	1	1
2	22	Female	Information Technology	1	6	0	1
3	21	Male	Information Technology	0	8	1	1
4	22	Male	Mechanical	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	Male	Information Technology	0	7	0	0
2962	23	Male	Mechanical	1	7	0	0
2963	22	Male	Information Technology	1	7	0	0
2964	22	Male	Computer Science	1	7	0	0
2965	23	Male	Civil	0	8	0	1

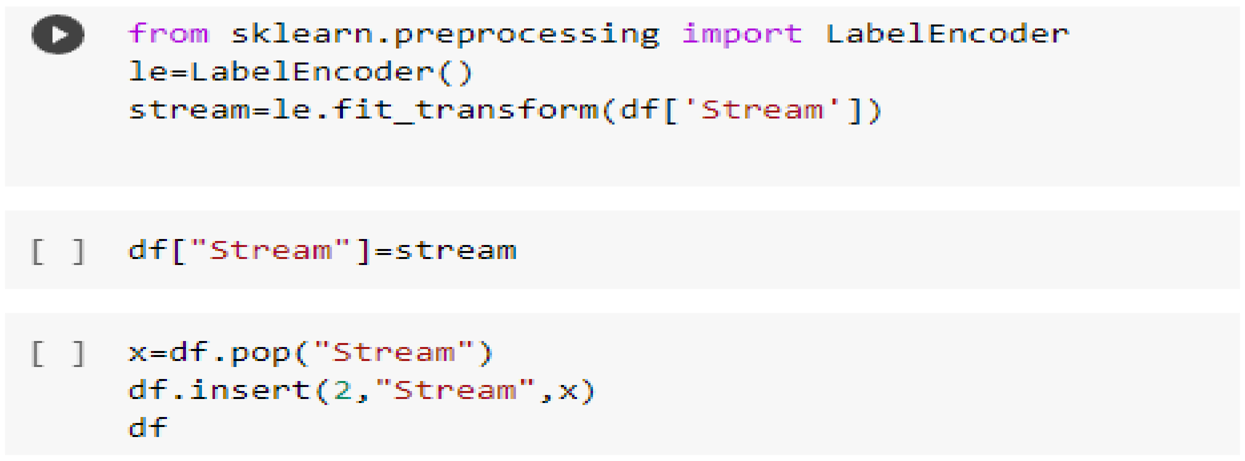
2966 rows × 7 columns

<u>Step 3:</u> Here as we can see the data contains non-numerical values under the columns designated as Gender and Stream and we need to convert them to numerical data. This can be done using

1.Dummy variable conversion method.

2. Label Encoder method.

We used the Label encoder method to change the values of Gender and Stream.



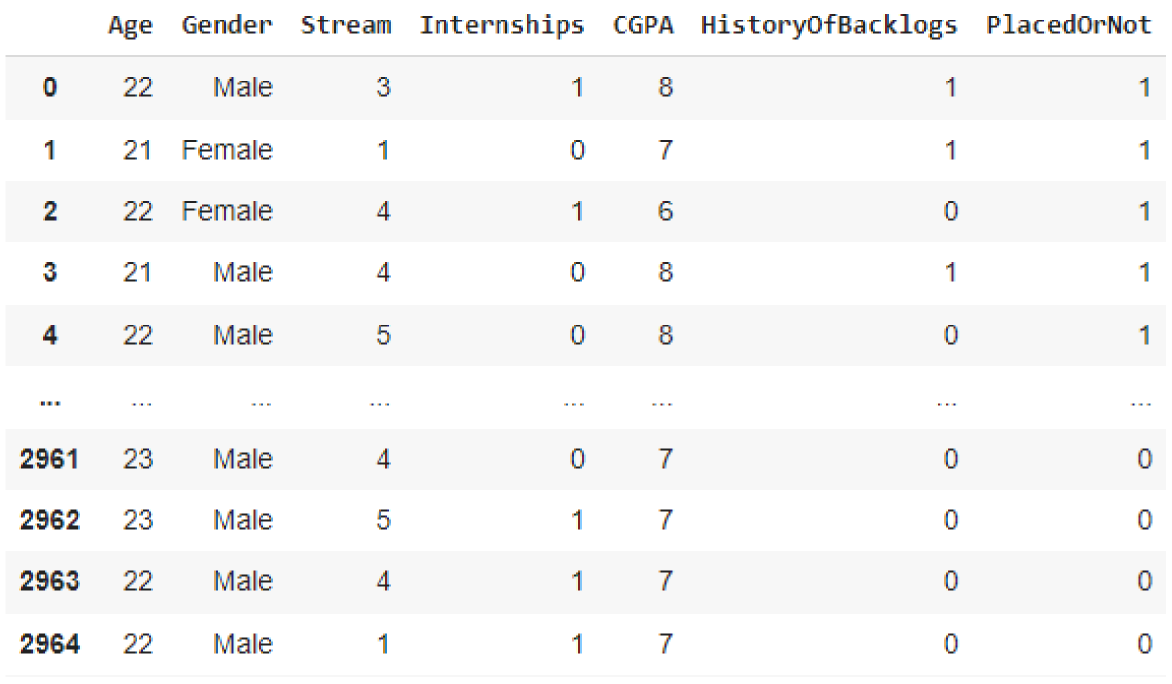
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
stream=le.fit_transform(df['Stream'])
```

```
[ ]  df["Stream"]=stream
```

```
[ ]  x=df.pop("Stream")
     df.insert(2,"Stream",x)
     df
```

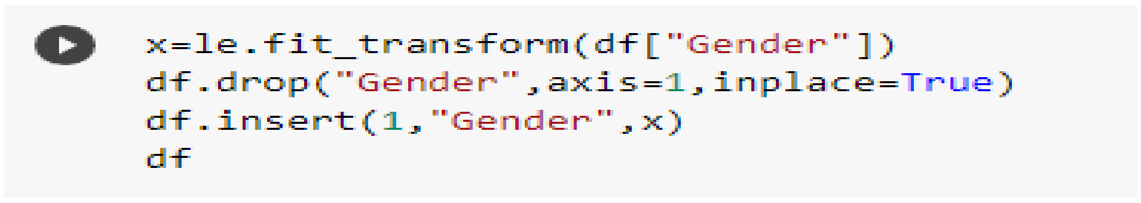
This will change thee values of Stream from non-numerical data to numerical data.

The corresponding output is as follows:



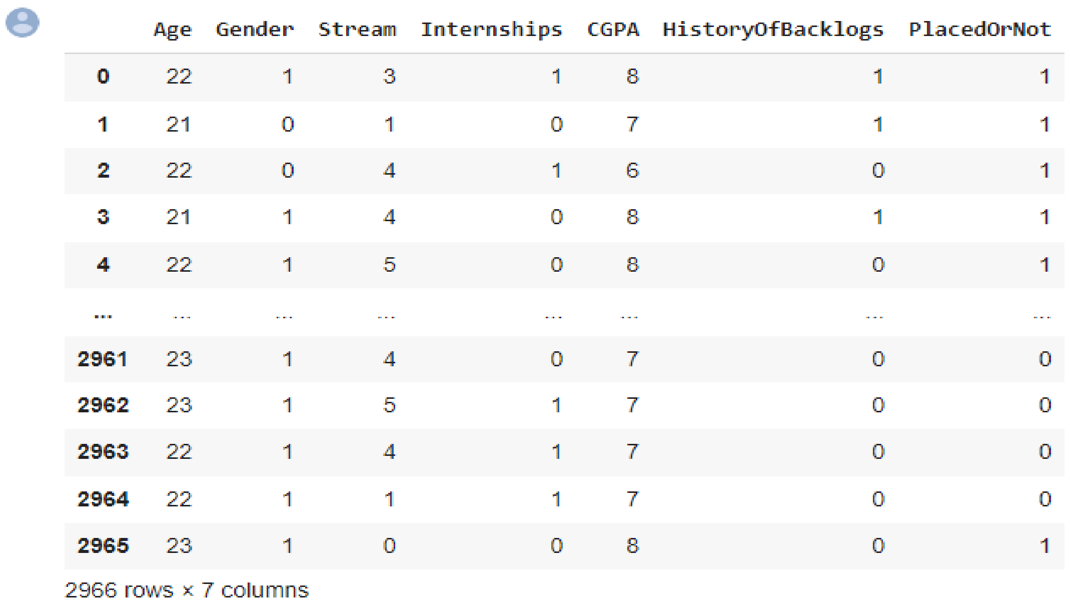
	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	Male	3	1	8	1	1
1	21	Female	1	0	7	1	1
2	22	Female	4	1	6	0	1
3	21	Male	4	0	8	1	1
4	22	Male	5	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	Male	4	0	7	0	0
2962	23	Male	5	1	7	0	0
2963	22	Male	4	1	7	0	0
2964	22	Male	1	1	7	0	0

Similarly the values under Gender attribute are also numerically encoded using Label Encoder method.



```
x=le.fit_transform(df["Gender"])
df.drop("Gender",axis=1,inplace=True)
df.insert(1,"Gender",x)
df
```

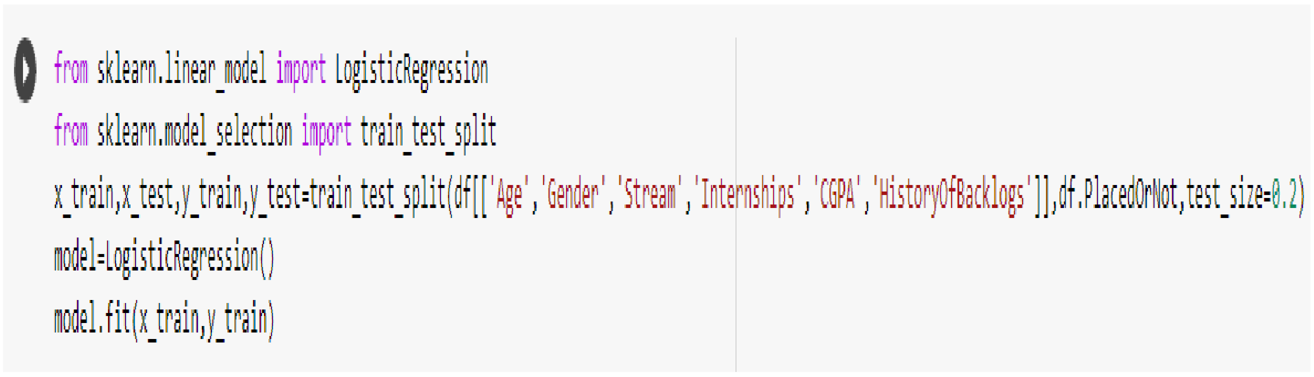
The corresponding output will be reflected in the original Data-Frame as below:



	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	1	3	1	8	1	1
1	21	0	1	0	7	1	1
2	22	0	4	1	6	0	1
3	21	1	4	0	8	1	1
4	22	1	5	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	1	4	0	7	0	0
2962	23	1	5	1	7	0	0
2963	22	1	4	1	7	0	0
2964	22	1	1	1	7	0	0
2965	23	1	0	0	8	0	1

2966 rows × 7 columns

Step 4: In this step we used the train_test_split() method to divide the datapoints in the dataset into training and testing data. These divided datapoints are used to train and then test the model.

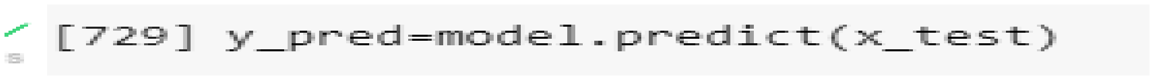


```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df[['Age','Gender','Stream','Internships','CGPA','HistoryOfBacklogs']],df.PlacedOrNot,test_size=0.2)
model=LogisticRegression()
model.fit(x_train,y_train)
```

We took 80% of the datapoints in the original dataset as training data. The remaining 20% of the datapoints are reserved for testing.

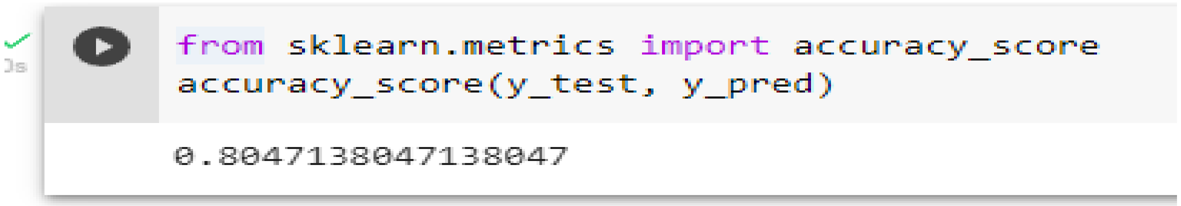
Now we train the LogisticRegression() model using x_train, y_train datapoints which are the training points corresponding to independent and dependent variables respectively.

Step 5: We used the training dataset to train thee mode. Now we will use the testing dataset find the accuracy of the model.



```
[729] y_pred=model.predict(x_test)
```

Now we used the x_test datapoints to find the predicted values.

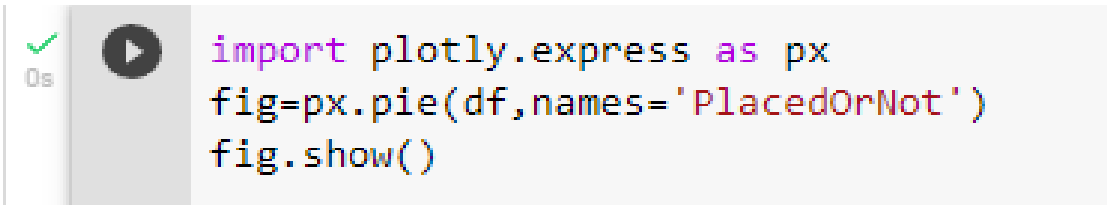


```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.8047138047138047
```

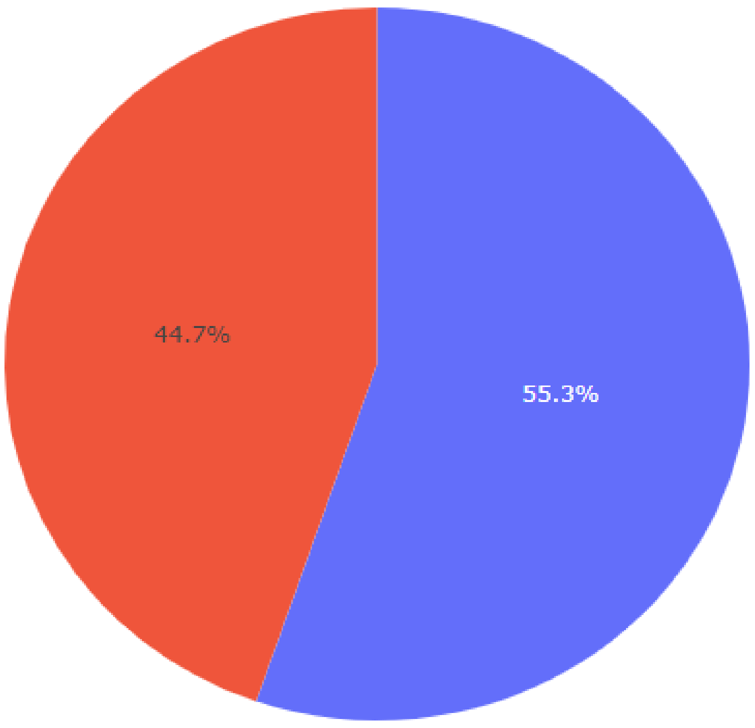
Step 6: Now we plot the graphs to analyse the dataset efficiently.

Figure 1: We will use pie-chart to analyse percentage of people placed and not-placed.



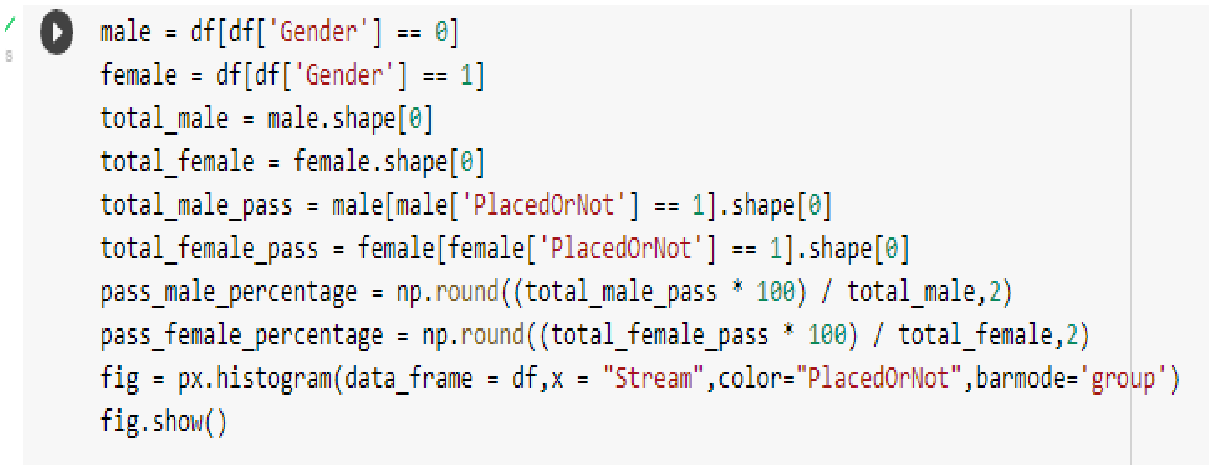
```
import plotly.express as px
fig=px.pie(df,names='PlacedOrNot')
fig.show()
```

The corresponding output is:



This indicates that 44.7% students are not placed and 55.3% students are placed.

Figure 2: Now we will plot a graph to see how many people are placed and not-placed streamwise.



```
male = df[df['Gender'] == 0]
female = df[df['Gender'] == 1]
total_male = male.shape[0]
total_female = female.shape[0]
total_male_pass = male[male['PlacedOrNot'] == 1].shape[0]
total_female_pass = female[female['PlacedOrNot'] == 1].shape[0]
pass_male_percentage = np.round((total_male_pass * 100) / total_male,2)
pass_female_percentage = np.round((total_female_pass * 100) / total_female,2)
fig = px.histogram(data_frame = df,x = "Stream",color="PlacedOrNot",barmode='group')
fig.show()
```

The corresponding output is as below:

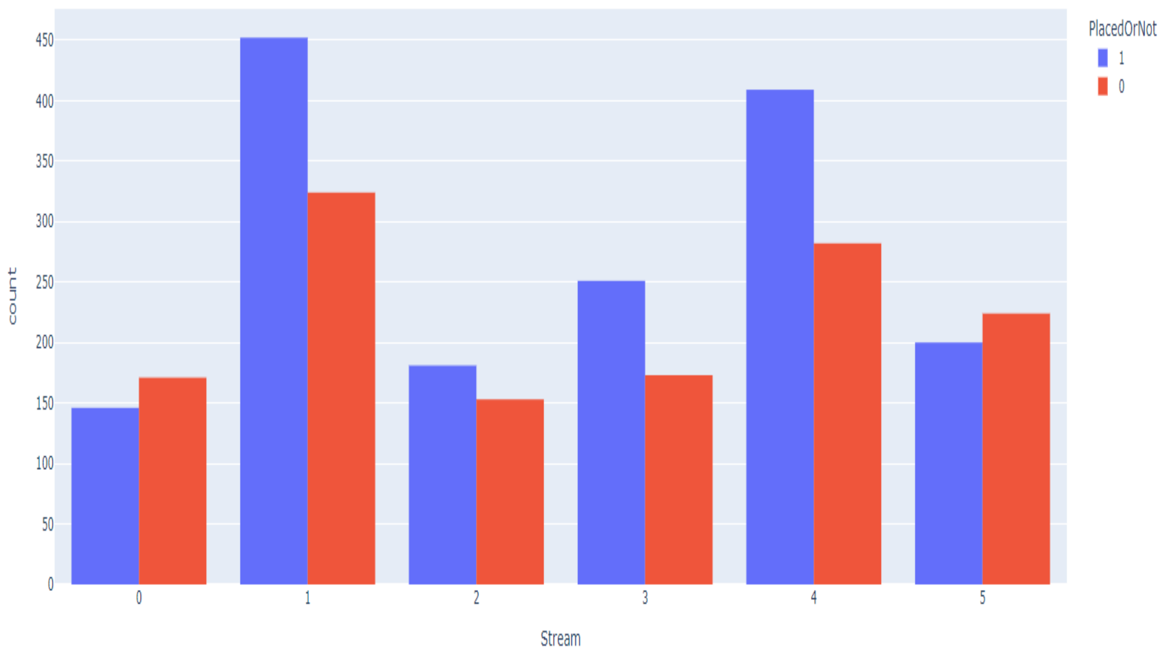


Figure 3:

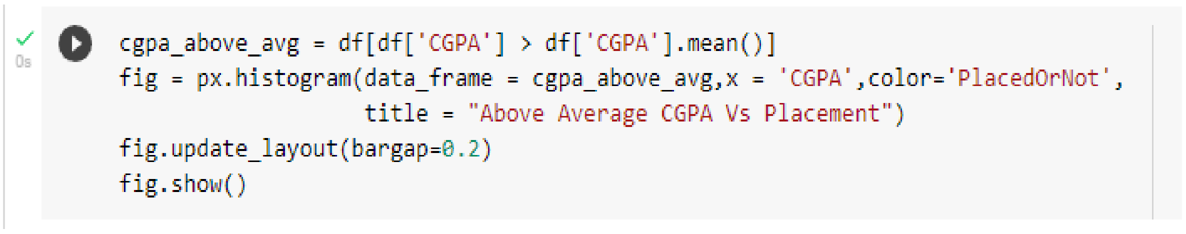
Now we plot a histogram to see how many students are placed who possess above-average CGPA.

Above average CGPA is the CGPA value that is greater than the average of all the CGPA values.

For this we take all the values of CGPA and then find the sum of the CGPA values.

After finding the sum of the CGPA values we will divide the sum with the number of datapoints we have in our dataset.

Using this data we will compute the Histogram.



```
cgpa_above_avg = df[df['CGPA'] > df['CGPA'].mean()]
fig = px.histogram(data_frame = cgpa_above_avg,x = 'CGPA',color='PlacedOrNot',
                   title = "Above Average CGPA Vs Placement")
fig.update_layout(bargap=0.2)
fig.show()
```

The Histogram for the above code is :

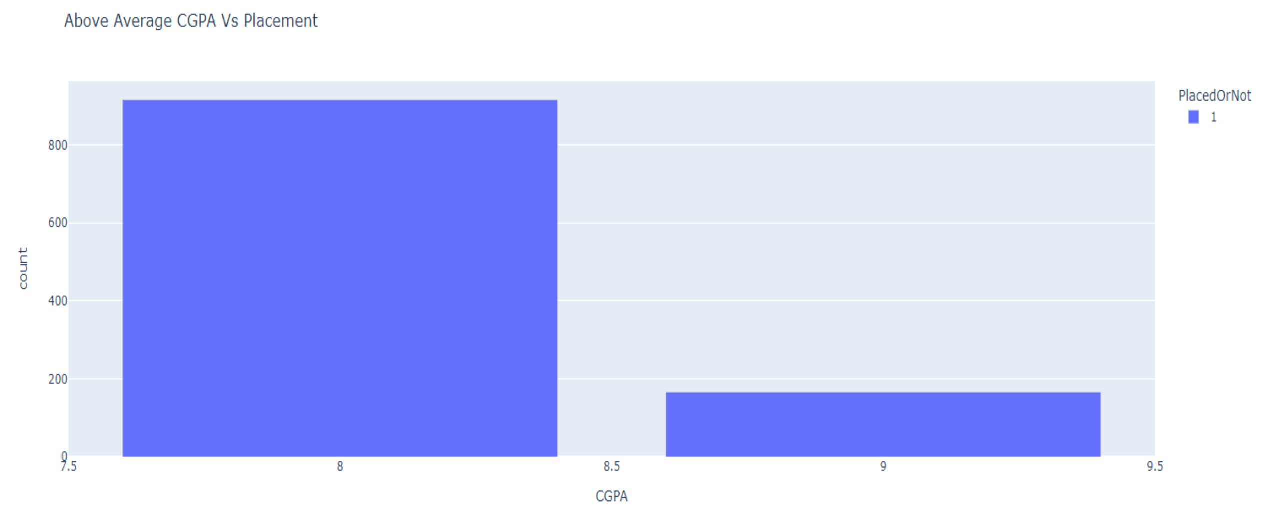
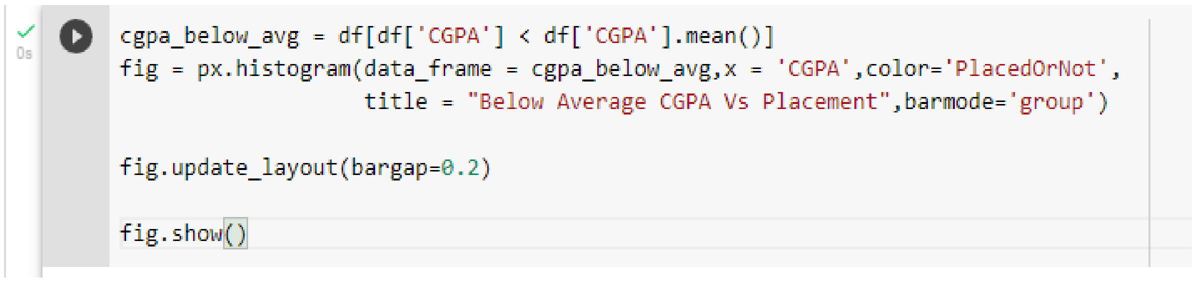


Figure 4: Now we will do the same process to plot a histogram that plots how many students are place who have below average CGPA.



```
cgpa_below_avg = df[df['CGPA'] < df['CGPA'].mean()]
fig = px.histogram(data_frame = cgpa_below_avg,x = 'CGPA',color='PlacedOrNot',
                   title = "Below Average CGPA Vs Placement",barmode='group')

fig.update_layout(bargap=0.2)

fig.show()
```

The graph for the above is as follows:

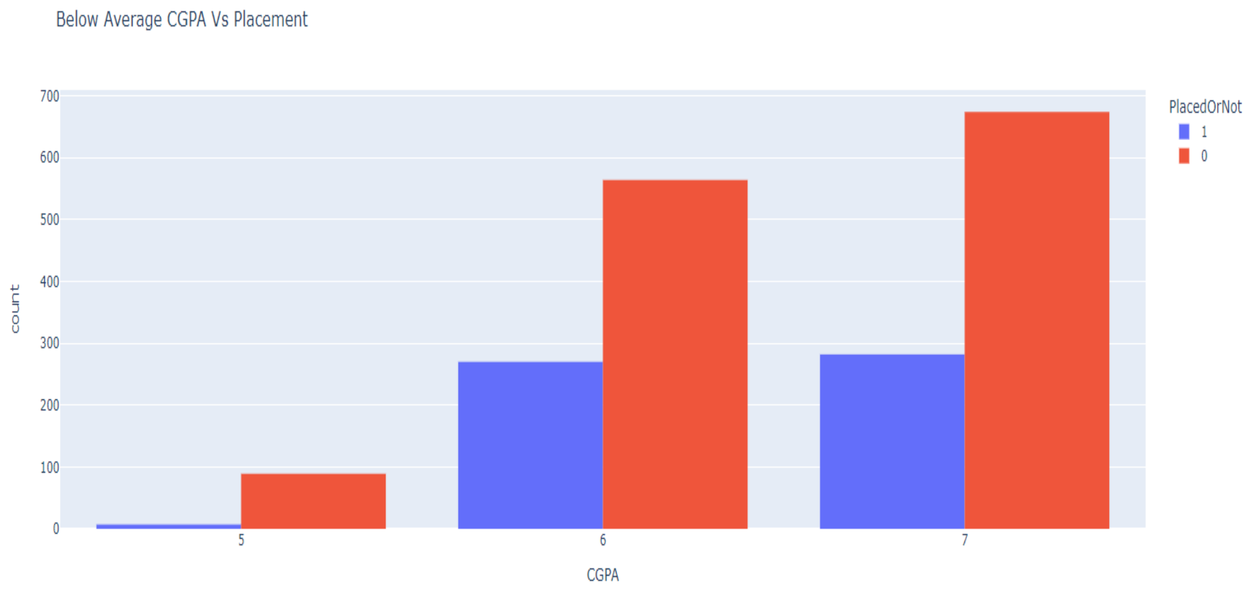
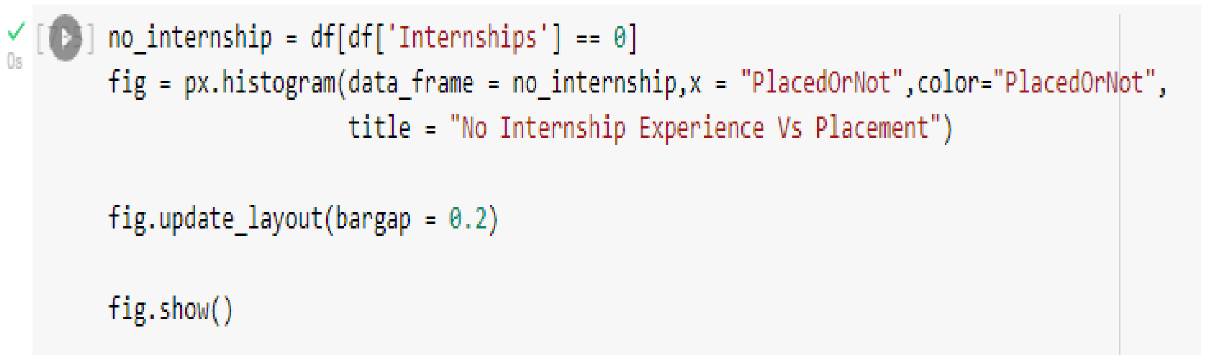


Figure 5: Now we will plot a histogram to count the number of people who have been placed with and without an Internship.

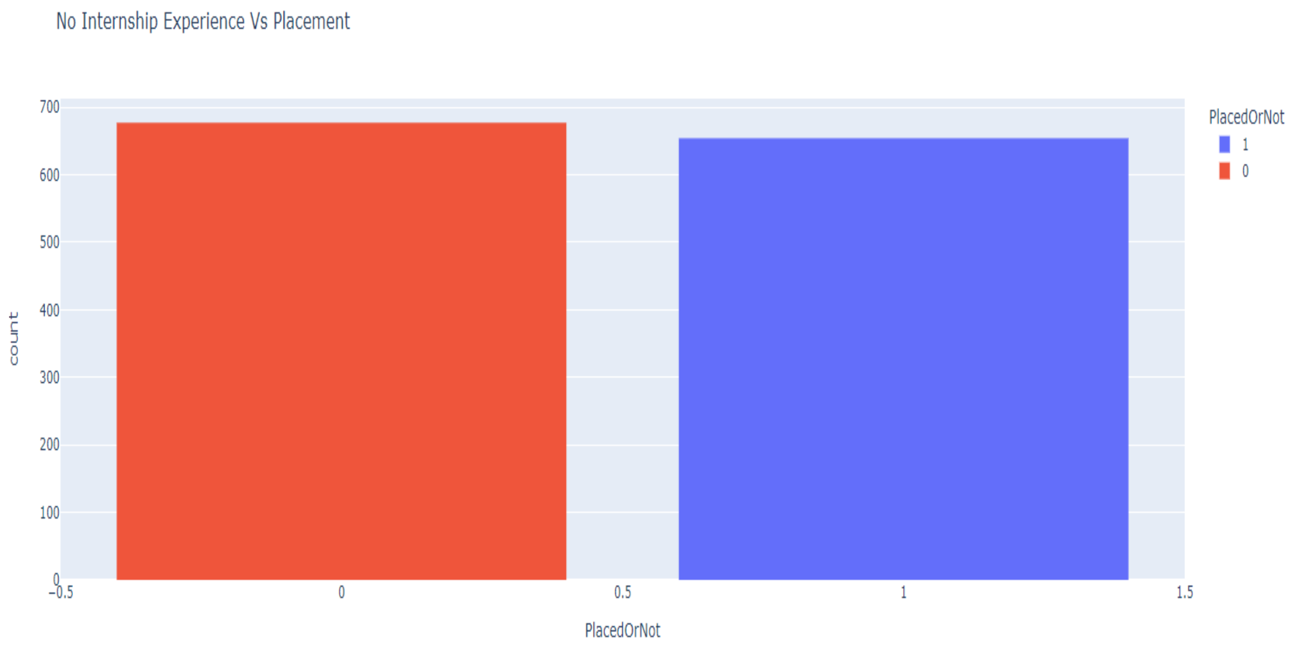


```
no_internship = df[df['Internships'] == 0]
fig = px.histogram(data_frame = no_internship,x = "PlacedOrNot",color="PlacedOrNot",
                   title = "No Internship Experience Vs Placement")

fig.update_layout(bargap = 0.2)

fig.show()
```

The graph for the above is as follows:



Acknowledgements:

We would like to sincerely thank our Machine Learning and Applications Faculty Mrs.Y.Sowjanya madam for guiding ,for constant support and motivating us from our project initiation till its conclusion.

References:

The Dataset we used for the project was attained from Kaggle an online community platform for data scientists and machine learning enthusiasts.

The link for accessing the dataset:

https://www.kaggle.com/datasets/tejashvi14/engineering-placements-prediction