Contents: Page Numbers

1. Introduction

1.1 Briefing

Our Project aims to predict the employability status of a student, whether he/she is placed or not based on various factors.

1.2 Problem Domain

Machine Learning: Multiple machine learning algorithms, the likes of regression models and classification approaches are applied to design our model.

1.3 Related Studies

- **Logistic Regression:** A statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression[1] (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1"

- **Naive - Bayes' Classification:** A classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods

1.4 Glossary

Acronyms	**Expansion**
CSV	Comma Separated Values
ML	Machine Learning

2. Problem Definition

2.1 Scope

The scope of our project is to determine the campus placement status of a student. Our model has been designed using two classification algorithms namely Logistic Regression and Naive Bayes' approach, comparing the accuracy of the two algorithms based on accuracy_score() .

2.2 Exclusion

Since our model utilizes classification approach to design the model, hence most of the algorithms which are based on linear relationship between the features has been excluded such as:

Linear Regression -> This algorithm is mostly used for continuous variables having linear relationship among features. Our model is a discrete one and lacks linearity, hence this algorithm has been discarded.

Decision Tree/Non-Linear Regression -> As discussed above, our model is a classification one which predicts discrete values, lacks both linearity and non-linearity among the features hence this algorithm is also discarded.

2.3 Assumptions

Some features/attributes do not contribute to the final output/labels for our project. We have considerably dropped those features to prepare a more convenient dataset aiming to get a better prediction and accuracy.

3. Project Planning

3.1 Data Collection

Kaggle is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users. This crowdsourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know beforehand which technique or analyst will be most effective. We have collected convenient data from Kaggle to design our project.

3.2 Data Analysis

Process	Purpose
Fetching the dataset	The desired dataset is fetched from Kaggle
Reading the raw data	The raw data is converted into dataframe
Data Pre-Processing	The raw data is processed to make it meaningful as per our needs
Feature Extraction	Important features are extracted from the data
Data wrangling	The dataset is cleansed and unified

4. Requirement Analysis

4.1 Requirement Matrix, system requirement

SL no	Tools
1	A computer with 8GB RAM
2	Anaconda Navigator/PyCharm – Python and its libraries
3	Spreadsheet

4.2 Requirement Elaboration

4.2.1 Computer Requirement

CPU: 2 x 64-bit 2.8 GHz 8.00 GT/s CPUs.
RAM: 8 GB (or 16 GB)
Storage: 300 GB.
Internet access to download the files from Anaconda Cloud or a USB drive containing all of the files you need with alternate instructions for air gapped installations.

4.2.2 Anaconda Navigator/PyCharm

Anaconda Navigator is an interactive environment which comprises various tools/IDEs like Jupyter Notebook, Spyder etc which is required to carry out multiple machine learning and data science algorithms using Python.
File format: **.ipynb**

PyCharm is an IDE to run various python files.
File format: **.py**

4.2.3 Spreadsheet

A spreadsheet is a computer application for organization, analysis, and storage of data in tabular form. Spreadsheets were developed as computerized analogs of paper accounting worksheets. We have used Microsoft Excel as our spreadsheet.

5. Design

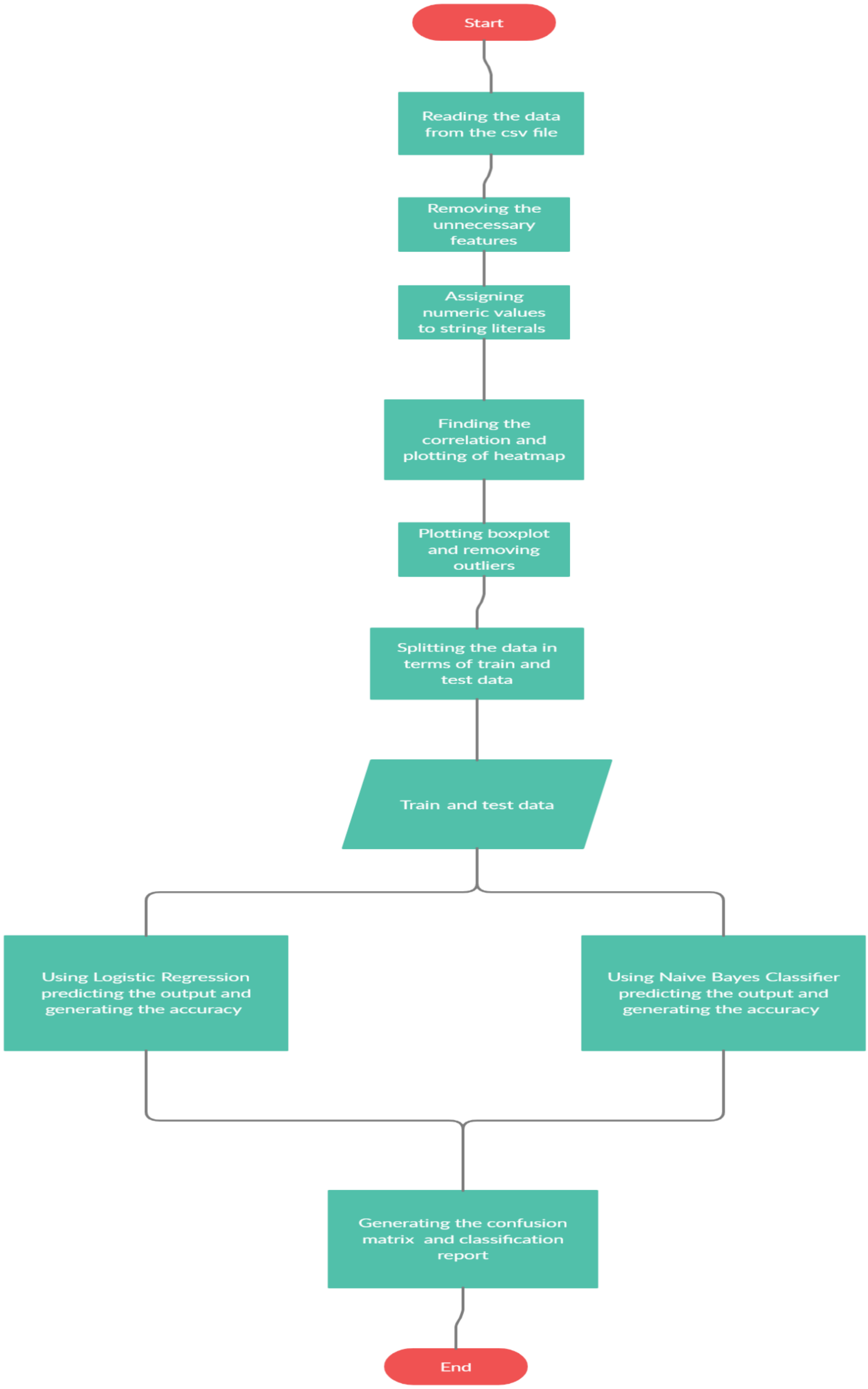
5.1 Technical Environment, software, module requirement

Our Environment

Our Software

Some of the modules used by us in this project have been mentioned below along with a pictorial representation.

5.2 Detailed Design



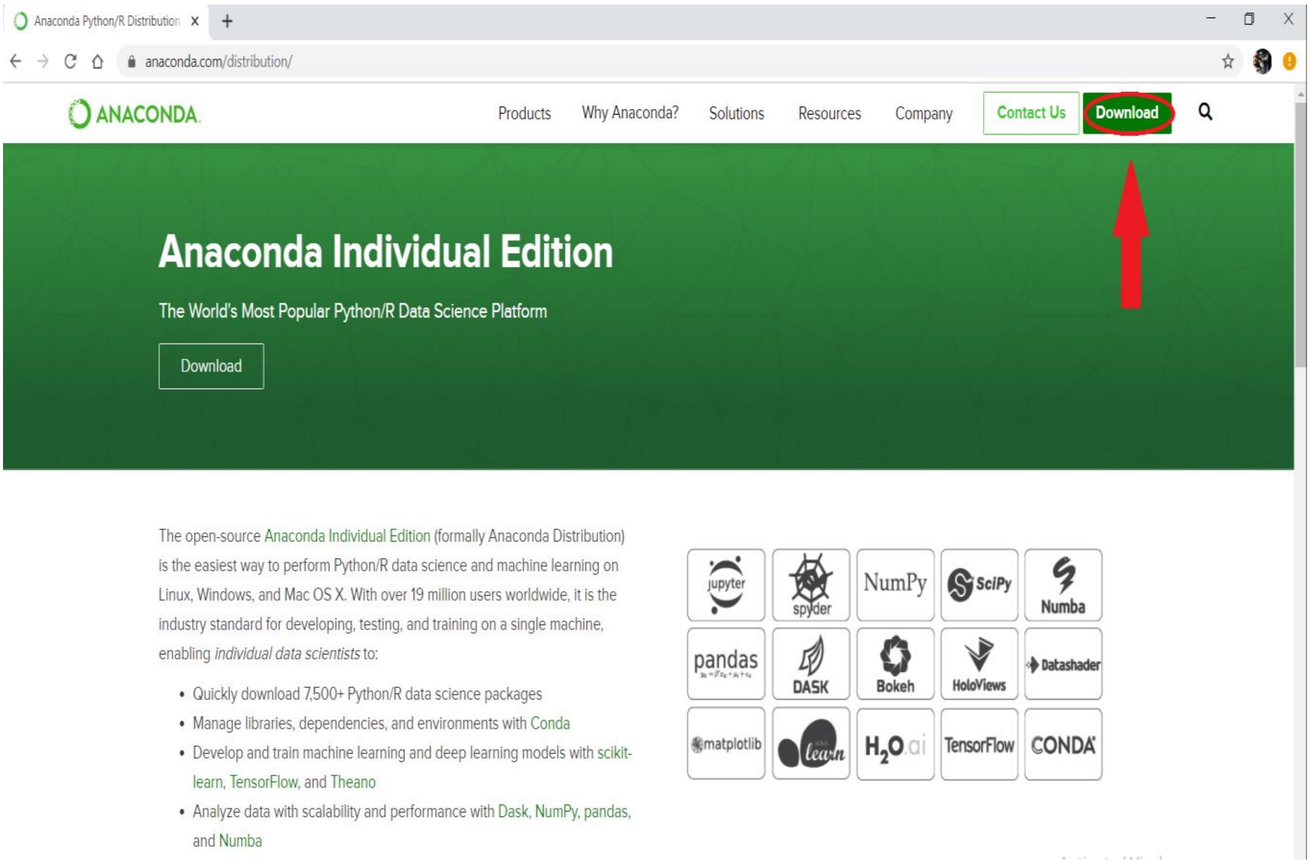
6. Implementation

6.1 System Installation Steps *(System Installation Manual in brief)*

Our code has been written and developed using the **Python 3** programming language along with **Jupyter Notebook** which provided us with an interactive computational environment and has been the primary notebook to develop, run, debug and execute our required unexpurgated code.

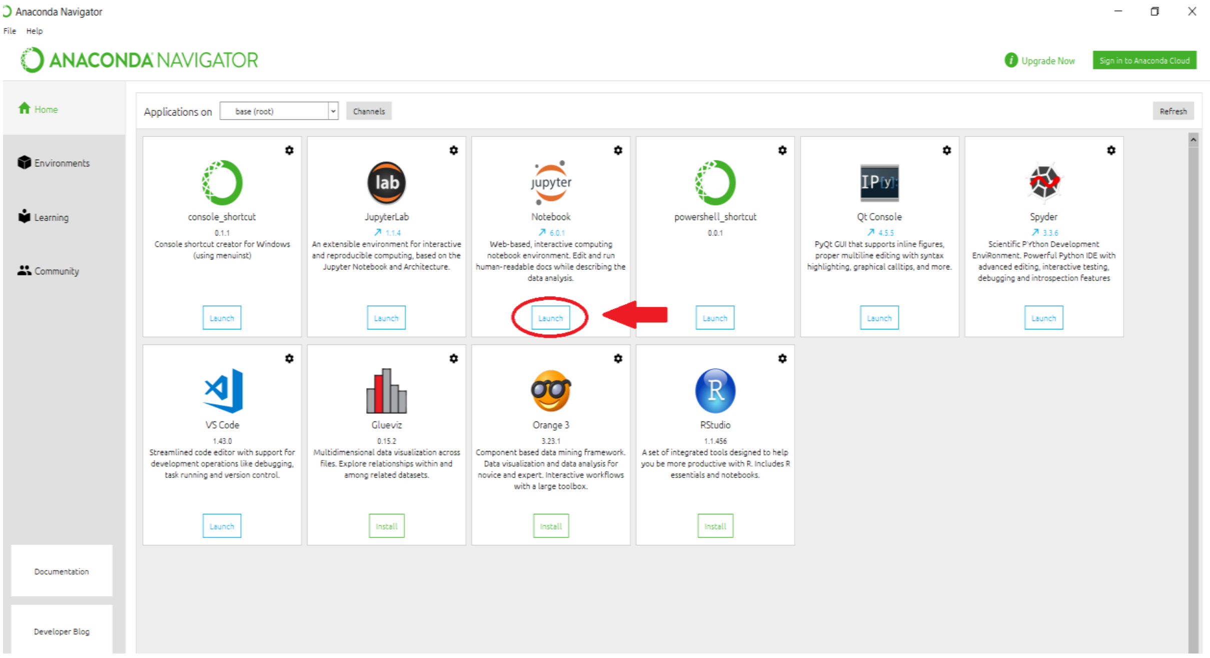
Thus, here providing all the necessary step-by-step guide on how to install Anaconda.

Step 1: Installing the Anaconda Navigator



- https://www.anaconda.com/distribution/ -- go to this link and download the Anaconda Navigator.
- After downloading follow the basic installation process.

<u>Step 2</u>: Running the Jupyter Notebook from your Anaconda Navigator



- Open the Anaconda Navigator from your machine.
- Hit the launch button shown in the image above to run the Jupyter notebook.

6.2 System Usage Instructions *(User Manual in brief)*

- Ensure you can run pip from the command line.
- Ensure pip, setup tools, and wheel are up to date.
- Optionally, create a virtual environment.
- Use pip for installing: The most common usage of pip is to install from the Python Package Index using a requirement specifier.

6.3 Give sample codes and visualization of your analysis

Import Packages

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

Loading the Dataset

url = r'D:\Summer Training TISL ML\Project\Placement_Data_Full_Class.csv' #reading the data from the csv file
plc = pd.read_csv(url)
plc.head()

Out[2]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

Data Pre-processing

Finding the features with null values

plc.isnull().sum()

sl_no	0
gender	0
ssc_p	0
ssc_b	0
hsc_p	0
hsc_b	0
hsc_s	0
degree_p	0
degree_t	0
workex	0
etest_p	0
specialisation	0
mba_p	0
status	0
salary	67
dtype:	int64

Removing the unnecessary columns

plc.drop(['sl_no','ssc_b','specialisation','mba_p','salary'],axis=1,inplace=True)

Out[6]:

	gender	ssc_p	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	status
0	M	67.00	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Placed
1	M	79.33	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Placed
2	M	65.00	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Placed
3	M	56.00	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Not Placed
4	M	85.80	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Placed

Assigning numerical values to the string literals

plc['gender'] = plc['gender'].map({'M':0,'F':1}) #assigning numerical values
plc['hsc_b'] = plc['hsc_b'].map({'Others':0,'Central':1})
plc['hsc_s'] = plc['hsc_s'].map({'Arts':0,'Commerce':1,'Science':2})

Generating the range using the "cut" function

plc['hsc_pBand'] = pd.cut(plc['hsc_p'],5)

Out[16]:

	gender	ssc_p	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	status	hsc_pBand
0	0	67.00	91.00	0	1	58.00	1	0	55.0	1	(85.56, 97.7]
1	0	79.33	78.33	0	2	77.48	1	1	86.5	1	(73.42, 85.56]
2	0	65.00	68.00	1	0	64.00	0	0	75.0	1	(61.28, 73.42]
3	0	56.00	52.00	1	2	52.00	1	0	66.0	0	(49.14, 61.28]
4	0	85.80	73.60	1	1	73.30	0	0	96.8	1	(73.42, 85.56]

Checking the dependency of "hsc_pBand" on the placement status

plc[['hsc_pBand','status']].groupby(['hsc_pBand'],as_index=False).mean()

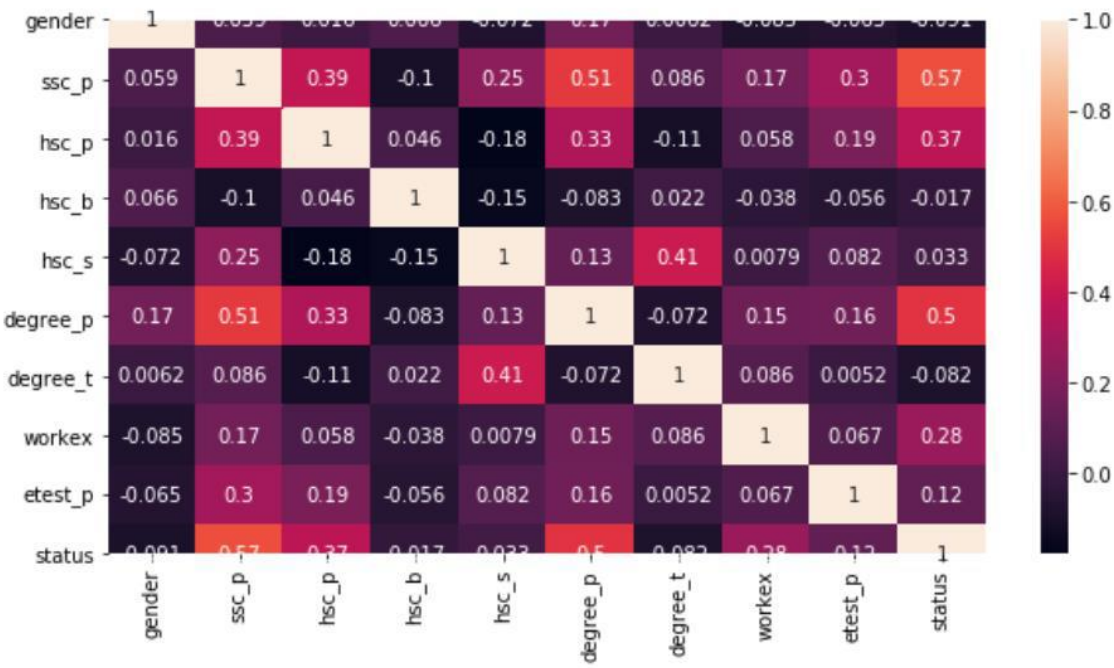
	hsc_pBand	**status**
0	(36.939, 49.14]	0.000000
1	(49.14, 61.28]	0.530612
2	(61.28, 73.42]	0.733333
3	(73.42, 85.56]	0.888889
4	(85.56, 97.7]	1.000000

Dividing the values of "hsc_p" into various ranges and assigning numerical values

plc.loc[plc['hsc_p'] <= 40, 'hsc_p'] = 0
plc.loc[(plc['hsc_p'] > 40) & (plc['hsc_p'] <= 65), 'hsc_p'] = 1
plc.loc[(plc['hsc_p'] > 65) & (plc['hsc_p'] <= 80), 'hsc_p'] = 2
plc.loc[(plc['hsc_p'] > 80) & (plc['hsc_p'] <= 90), 'hsc_p'] = 3
plc.loc[plc['hsc_p'] > 90, 'hsc_p'] = 4

Generating the “heatmap”

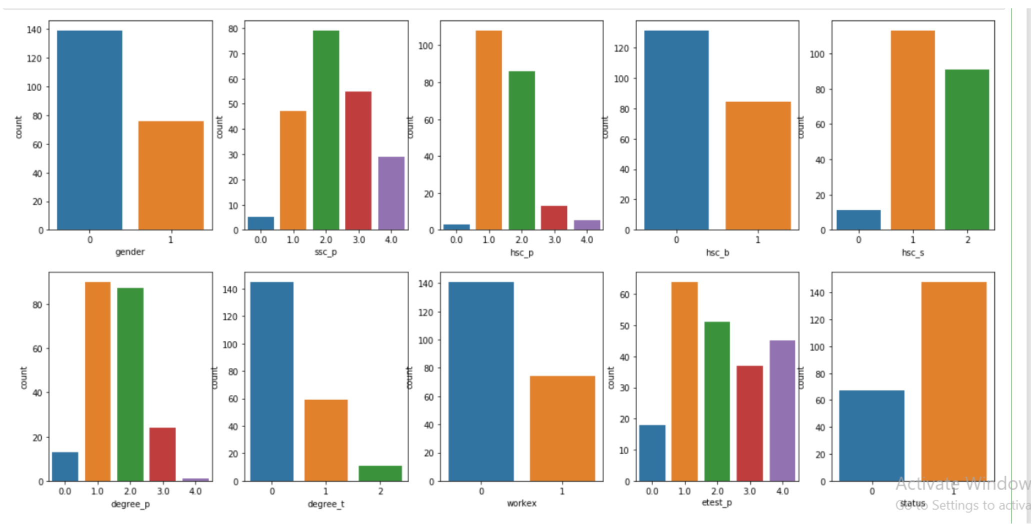
```
fig, corr = plt.subplots(figsize=(10,5))
sb.heatmap(plc.corr(), annot=True)
plt.show()
```



Data Visualization

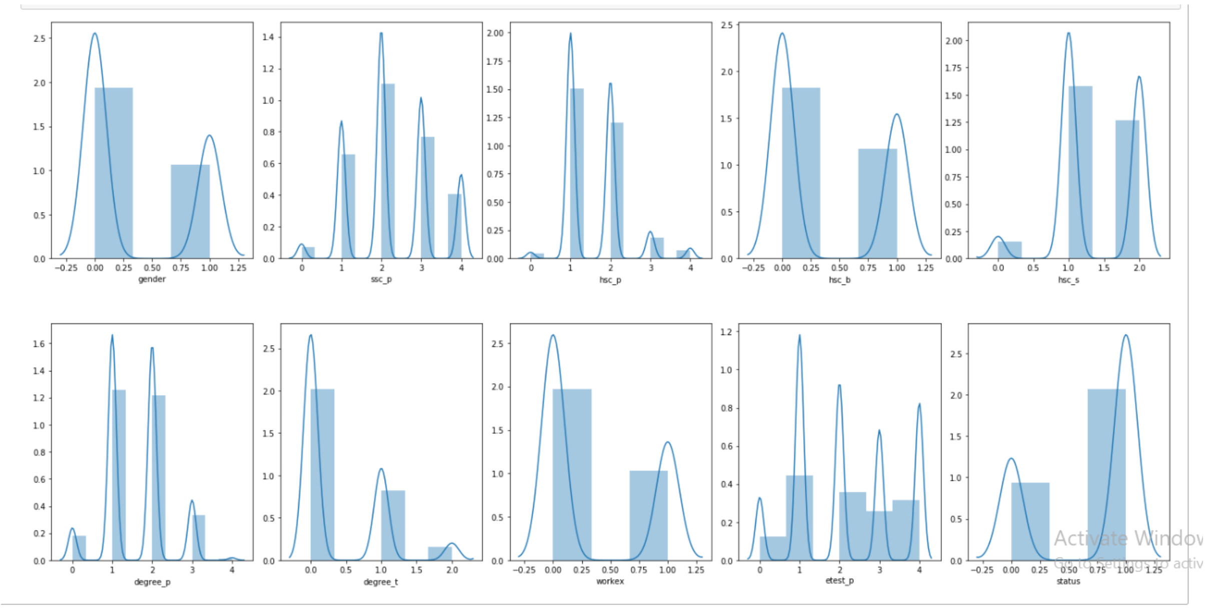
a) Count plot

```
fig2, axs = plt.subplots(ncols=5, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten() # to flaten to 1d
for k,v in plc.items():
sb.countplot(x=v, data=plc, ax=axs[index])
index += 1
```



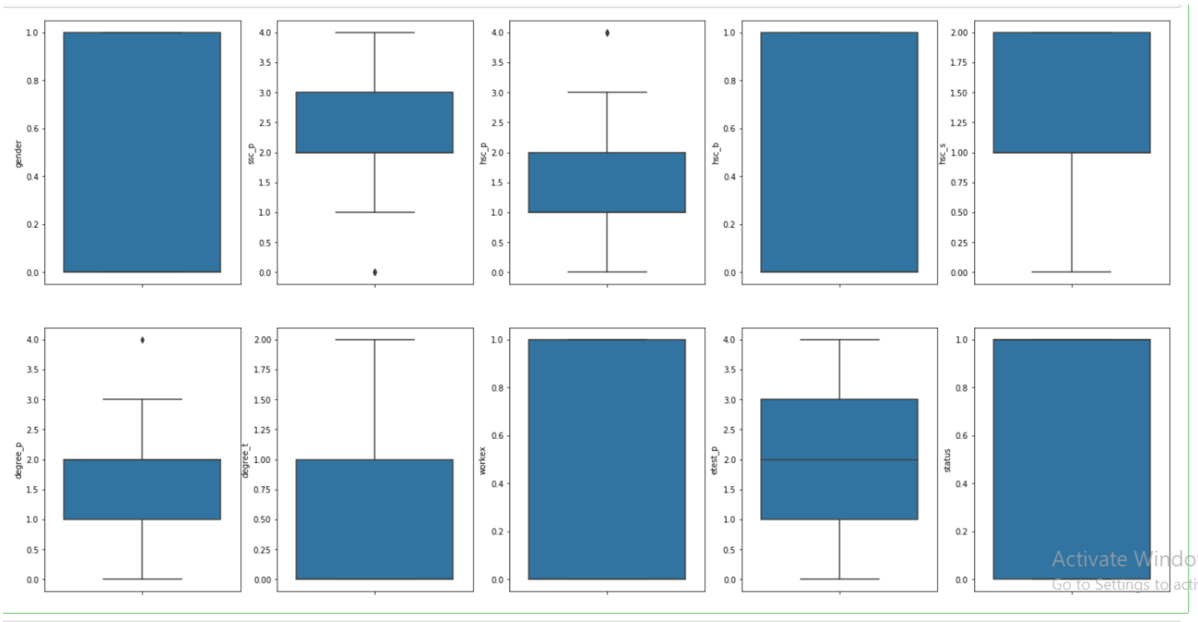
b) Distribution graph

```
fig3, axs = plt.subplots(ncols=5, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in plc.items():
sb.distplot(v, ax=axs[index],kde_kws={'bw': 0.1})
index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
plt.show()
```



c) Boxplot

```
fig1, axs = plt.subplots(ncols=5, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in plc.items():
sb.boxplot(y=v, data=plc, ax=axs[index])
index += 1
plt.tight_layout(pad=0.4, w_pad=0.1, h_pad=5.0)
plt.show()
```



Extracting and splitting of data

Extraction

```
X=plc.iloc[:,:9]
Y=plc.iloc[:,9]
```

Splitting

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=.1,random_state=1)
```

Logistic Regression Analysis

```
log=LogisticRegression()
log.fit(x_train,y_train)
y_pred=log.predict(x_test)
accuracy_score(y_pred,y_test)
```

0.8181818181818182

```
log.score(x_train,y_train)
```

0.8549222797927462

Naïve Bayes Classification

```
from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(x_train, y_train)
y_prednb = nvclassifier.predict(x_test)
accuracy_score(y_prednb,y_test)
```

0.7727272727272727

nvclassifier.score(x_train,y_train)
0.8393782383419689

Generating the "confusion matrix" and "the classification report"

a) Logistic Regression

confusion_matrix(y_pred,y_test)
array ([[5, 2], [2, 13]], dtype=int64)
print(classification_report(y_pred,y_test))

	precision	recall	f1-score	support
0	0.71	0.71	0.71	7
1	0.87	0.87	0.87	15
accuracy			0.82	22
macro avg	0.79	0.79	0.79	22
weighted avg	0.82	0.82	0.82	22

b) Naïve Bayes Classification

confusion_matrix(y_prednb,y_test)

array([[5, 4], [1, 12]], dtype=int64)
print(classification_report(y_prednb,y_test))

	precision	recall	f1-score	support
0	0.83	0.56	0.67	9
1	0.75	0.92	0.83	13
accuracy			0.77	22
macro avg	0.79	0.74	0.75	22
weighted avg	0.78	0.77	0.76	22

Result

The accuracy achieved using **Logistic Regression** is **82%** (accuracy score)

The accuracy achieved using **Naïve Bayes Classification** is **77%** (accuracy score)

7. Conclusion

7.1. Project Benefits

The campus placement prediction aims to project a detail study on the campus recruitment of a handful of 200 students. The project achieves an accuracy of **82%** in detecting the placement status of a student . The model uses Logistic Regression approach in establishing the accuracy since the label is binary.

Not placed	0
Placed	1

Our data set consists of Placement data of students in our campus. It includes secondary and higher secondary school percentage and specialization. It also includes degree specialization, type and Work experience and salary offers to the placed students.

Apart from Logistic Regression, we have applied Naïve Bayes Classification in order to compare the accuracy percentage of the two algorithms. It is a classification algorithm which is mostly used for multiple classification but also used for binary classification. We have got a **77%** accuracy.

In the data wrangling part, we have assigned a numerical form to the string input for the complementary analysis since Machine Learning algorithms cannot work on string inputs.

We have discarded the Linear Regression algorithm since very few of the features have a linear relationship with the label.

7.2. Future Scope for Improvements

This project can be easily implemented under various situations. We can add features as and when required. Reusability and flexibility can be exhibited in the modules. This project is extendable in ways that its original developers may not expect. The model enhances extensibility like Supervised Learning and binary classification. With the advent of upgraded version of the
programming language we might be to reduce the code and simplify our understanding.

Machine Learning models are created with the aim of increasing accuracy over manual analysis. In most of the cases we find the models being biased towards the prediction of output as the models do not take into account the class distribution/parents of classes. Data imbalance usually reflects an unequal distribution of classes with in a dataset. Dealing with an imbalance dataset entails strategists such as improving classification algorithms or balancing classes in the data pre-processing stage before the model is trained with the data. Data imbalancing can be handled primarily using two methods:-

i) Re-sampling
- Over sampling
- Under sampling

ii) Ensembling methods

In conclusion, without handling the data imbalancing section the overall performance of our ML model will be constrained by its ability to predict rare and minority points. Thus, identifying and resolving the imbalance of those points is critical to the quality and performance of the generated model.

8. References

We have referred to the following links in the making of our project:

- https://www.kaggle.com/benroshan/factors-affecting-campus-placement
- https://seaborn.pydata.org/introduction.html#:~:text=Seaborn%20is%20a%20library%20for,examining%20relationships%20between%20multiple%20variables
- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- Video lectures on Youtube

We would also like to express our gratitude to Prof. Mousita Dhar for guiding us in this project.