

Identifying Patterns and Trends in Campus Placement Data using Machine Learning

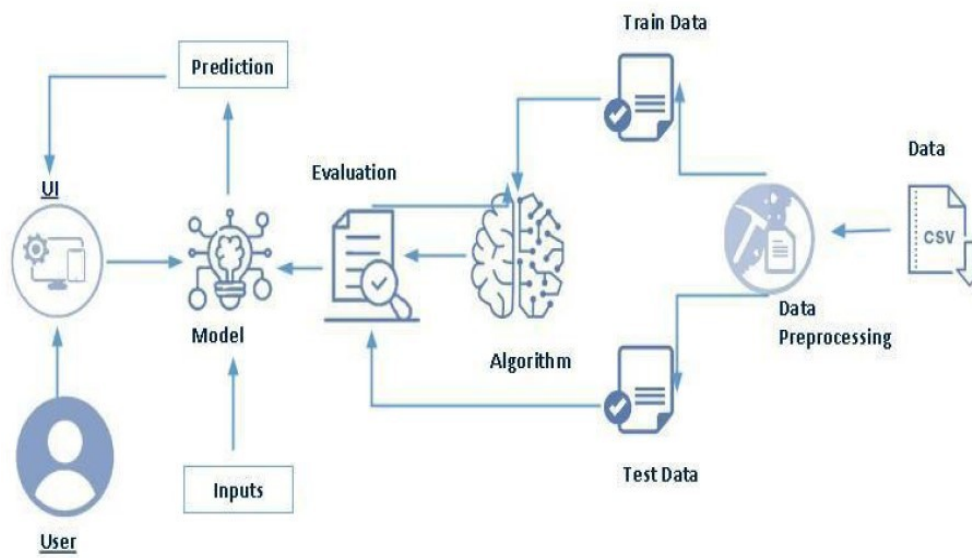
Overview:

Machine learning is a method of data analysis that automates analytical model building. These models help you to make a trend analysis of university placements data, to predict a placement rate for the students of an upcoming year which will help the university to analyze the performance during placements. Many students look at universities as a means of investment which can help them make a great future by getting placed in good companies and which will relieve their stress and unease from their lives before graduating from the university.

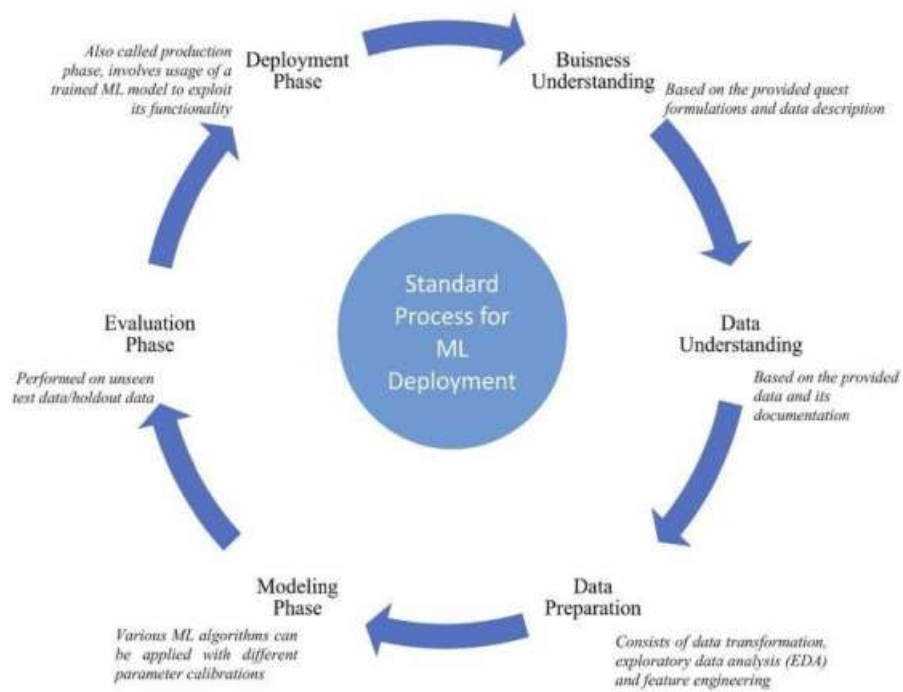
Introduction:

This paper focuses on the trend analysis of university placement. A trend analysis means an aspect of technical analysis that tries to predict future movement on past data. Students are the main stakeholders of universities and their performance plays a significant role in a country's social and economic growth by producing creative graduates, innovators and entrepreneurs. a millions of university and colleges provides the technical education now a days. But few of them land in good jobs as compared to the total. A university placement plays a very important role to get more admission and maintain the reputation in the world. Authors consider some of the important attributes of students like company, package, location etc. a machine learning plays the most important role to find out the performance, define the trend or predict the placement for the upcoming year.

Technical Architecture:



Technical about the project:



Project Description:

Campus recruitment is a strategy for sourcing, engaging and hiring young talent for internship and entry-level positions. College recruiting is typically a tactic for medium- to large-sized companies with high-volume recruiting needs, but can range from small efforts (like working with university career centers to source potential candidates) to large-scale operations (like visiting a wide array of college and attending recruiting events throughout the spring and fall semester). Campus recruitment often involves working with university career services centers and attending career fairs to meet in person with college students and recent graduates. Our solution revolves around the placement season of a Business School in India. Where it has various factors on candidates getting hired such as work experience, exam percentage etc, finally it contains the status of recruitment and remuneration details.

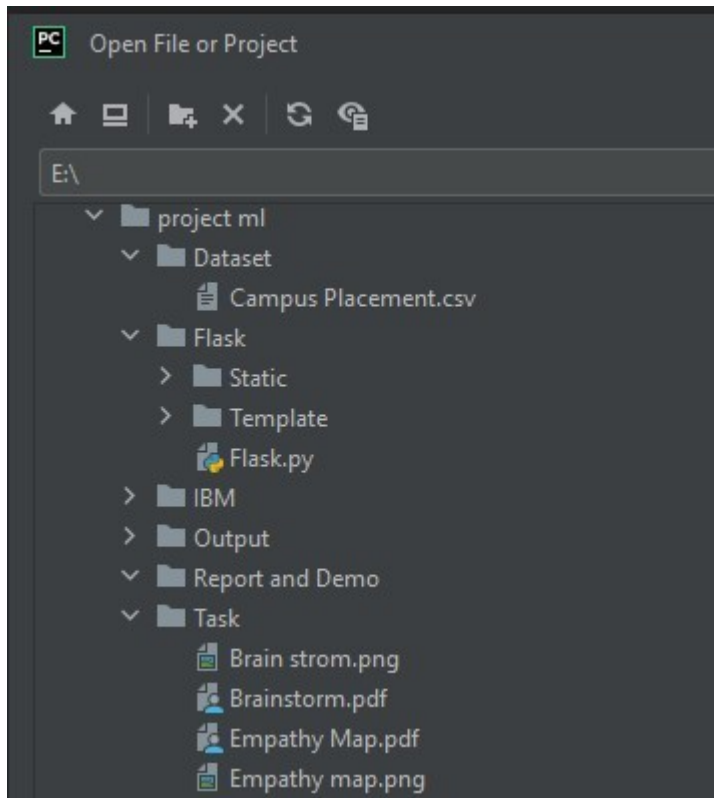
We will be using algorithms such as KNN, SVM and ANN. We will train and test the data with these algorithms. From this the best model is selected and saved in.pkl format. We will be doing flask integration and IBM deployment.

Project Flow:

- ❖ User interacts with the UI to enter the input.
- ❖ Entered input is analyzed by the model which is integrated.
- ❖ Once model analyzes the input the prediction is showcased on the UI
- ❖ To accomplish this, we have to complete all the activities listed below,
- ❖ Data collection
 - Collect the dataset or create the dataset
- ❖ Visualizing and analyzing data
 - Univariate analysis
 - Bivariate analysis
 - Multivariate analysis
 - Descriptive analysis
- ❖ Data pre-processing
 - Checking for null values
 - Handling outlier
 - Handling categorical data
 - Splitting data into train and test

- ❖ Model building
 - Import the model building libraries
 - Initializing the model
 - Training and testing the model
 - Evaluating performance of model
 - Save the model
- ❖ Application Building
 - Create an HTML file Build python code

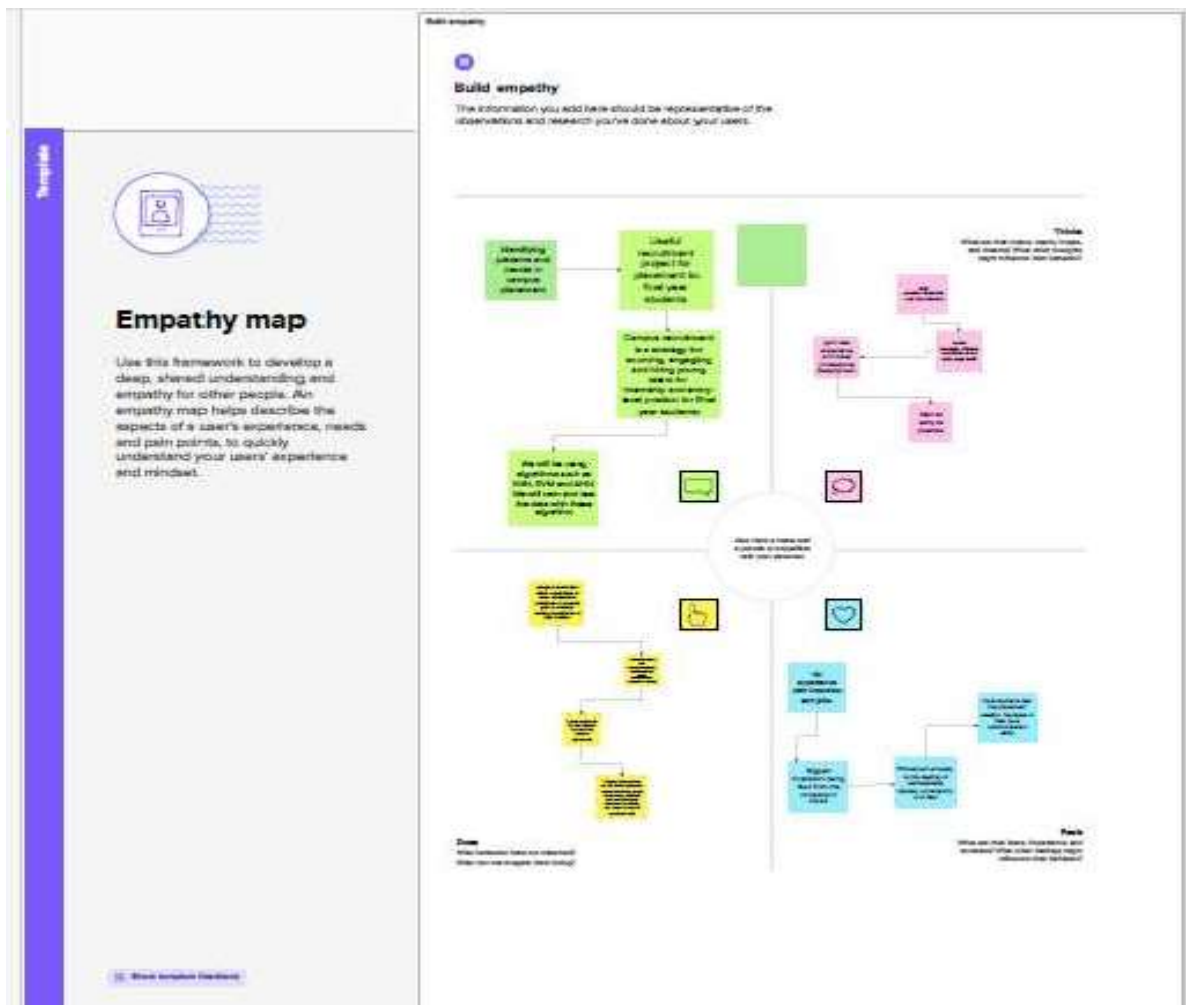
Project Structure:



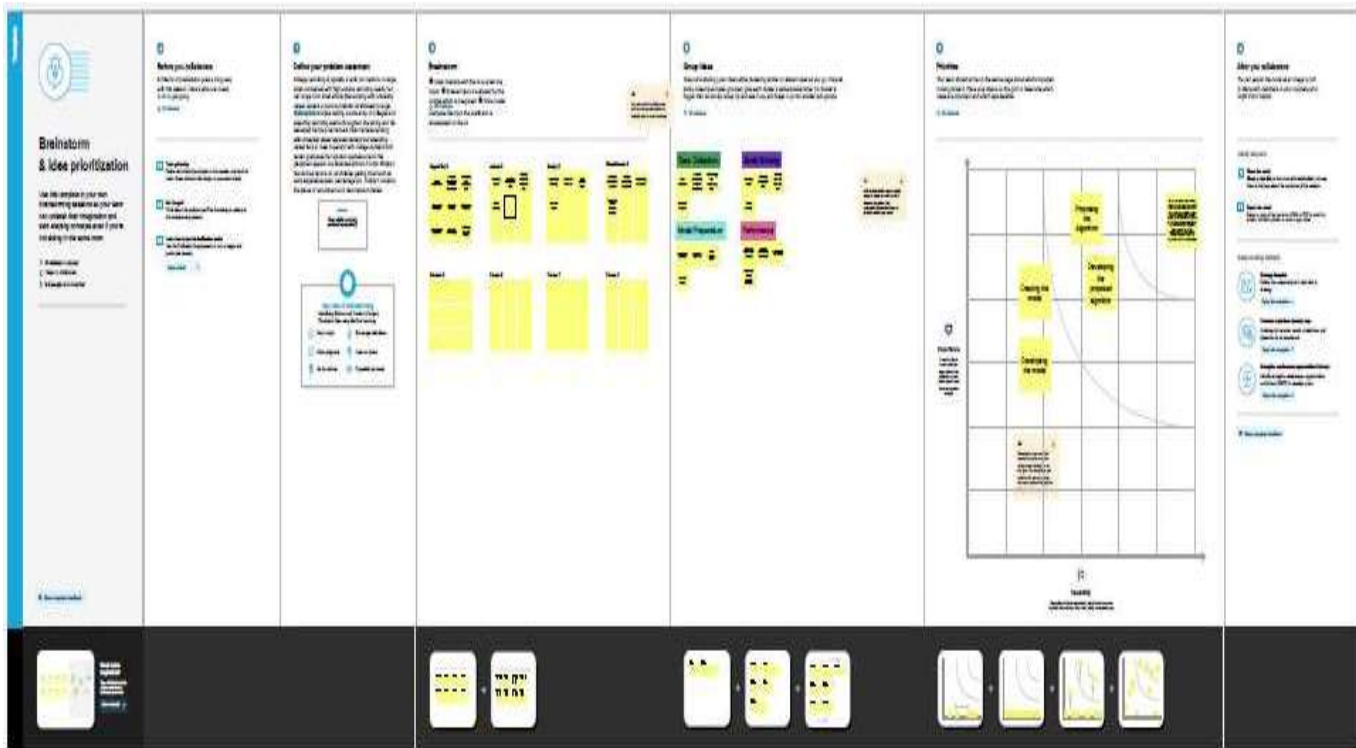
Purpose:

These algorithms independently predict the results and we then compare the efficiency of the algorithms, which is based on the dataset. This model helps the position cell at intervals a corporation to spot the potential students and concentrate to and improve their technical and social skills.

Empathy Map:



Brainstorm & idea prioritization:



Result:

Fill the details

Age

Gender M(0), F(1)

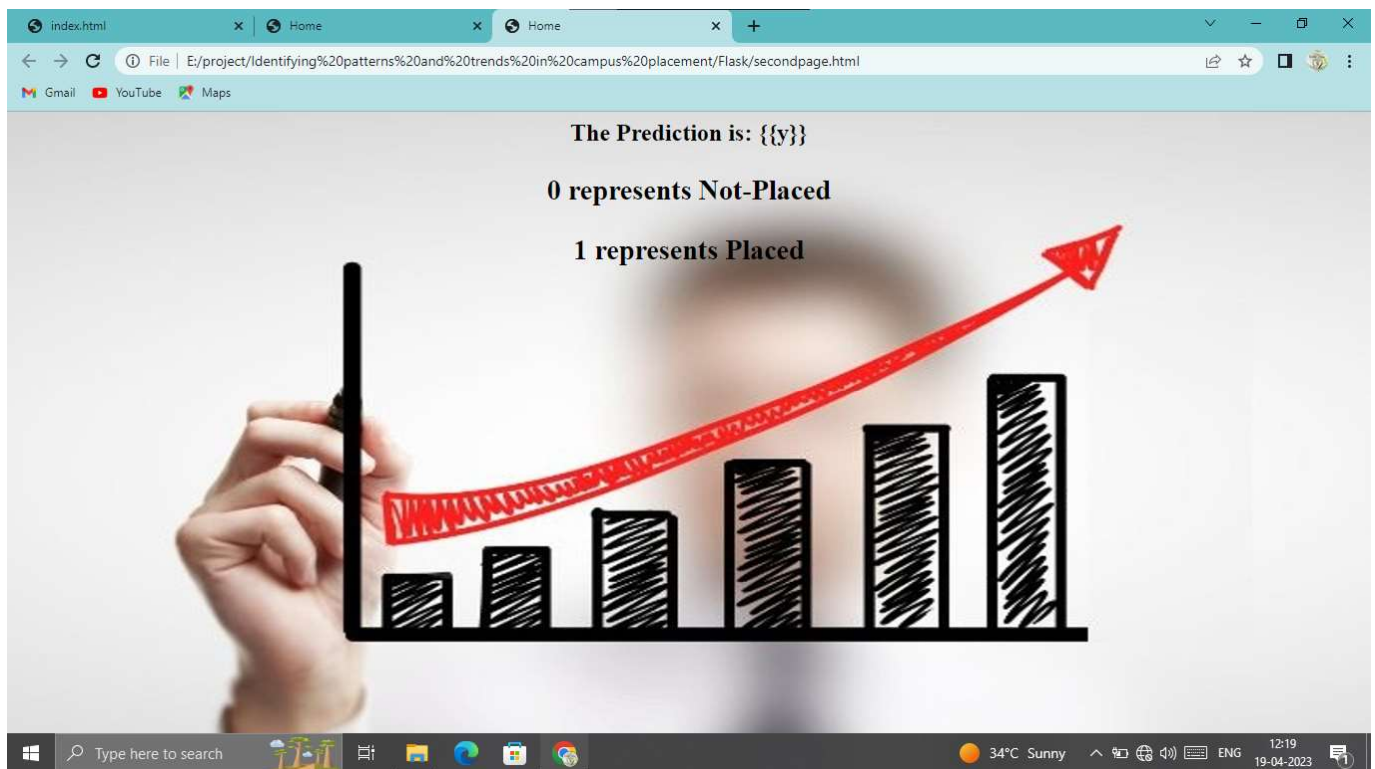
Stream CS(0), IT(1), ECE(2)

Internships

CGPA

Number of backlogs

Placement



Advantages:

- ❖ Students will get complete knowledge of the placement process.
- ❖ Students will get enough time for preparation.
- ❖ Everyone will be able to know their strength and weakness and in which area they need to work.
- ❖ Their route to reach their goals will be much easier as we will be helping them at every step.
- ❖ Even if students lack behind in academics our proposed system will help them to choose different routes according to their skills and also suggest companies which suit their caliber.

Disadvantages:

- ❖ Random forests have been observed to overfit for few datasets with noisy classification/regression tasks.
- ❖ Campus recruitment is an expensive affair for majority of the companies as it adds up costs to the bottom line. Companies incur different expenses related to travel, boarding, training etc while conducting campus selection process.
- ❖ The experienced and skilled candidates having practical job exposures cannot be recruited through campus placements. Fresh candidates selected through campus placements require adequate training for work.
- ❖ This is an additional expense for the company. Also, students can't work with their dream company and will have to remain satisfied with the company that recruits them during campus selection.

APPLICATION

1. Decisions in business operations

Machine Learning algorithms come to the rescue in areas built on a constant flow of heterogeneous data, whether it is several financial reports, payrolls, procurement, the analysis of employee productivity, or predicting further churn rates. Overall, AI, in terms of inner business processes, is able to leverage business intelligence and make a company data-driven in many aspects, including decision making.

2. Complex problem-solving

The potential of AI in decision making is robust, but you can solve multilayer and complex problems. Artificial Intelligence here gathers tons of different data and conducts an interdisciplinary study. Eventually, there's a way to leverage anything from product development stages to digital marketing approaches of product promotion. Also, it's a way to optimize various types of predictions and risk management. For example, you can predict and optimize pricing with the help of AI tools.

3. Strategic changes

AI allows better planning of production, managing all restrictions, reducing shortcomings in operations, and improving manufacturing. It also helps to anticipate and adequately plan product customization, enhance postponement processes, and maintain efficiency with high levels of customer satisfaction.

4. Customer-related decisions

AI can be valuable for customer service management, personalized customer communication, evaluation of customer behavior, predicting consumer trends and patterns. Artificial intelligence enables automatic recognition and profiling of potential customers.

Conclusion:

This system will be helpful to students from the beginning of their Engineering career .This system will reduce the chaos caused at the end of the final year.

Students will start improving themselves from second year itself about their career awareness learning new skills throughout their graduation course. This system will help them to achieve their dream company as well as they will learn how to overcome their weaknesses. Students will be clear about their career growth and what various options are available in the market and how far they can improve themselves .Also with the result generated from the proposed system college placement cell will be well aware of what new skills can introduced by upbringing new training sessions in the college so that maximum student can get benefited out of the training. Also suggestions of new recruitments and the company criteria and requirement of skills to be known will be messaged to the students at very early stage .This system covers all the aspects for increasing the placement in undergraduate students. This system will also helpful for development of college as new project skills will be created by student and percentage of placement will be increased overall.

Future Scope:

With the help of machine learning services like SDKs and APIs, developers are able to include and hone the intelligent capabilities into their applications. This will empower machines to apply the various things they come across ,and accordingly carry out an array of duties like vision recognition, speech detection, and understanding of speech and dialect.

Appendix:

Source Code:

- Importing the libraries

```
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib
from sklearn.metrics import accuracy_score
```

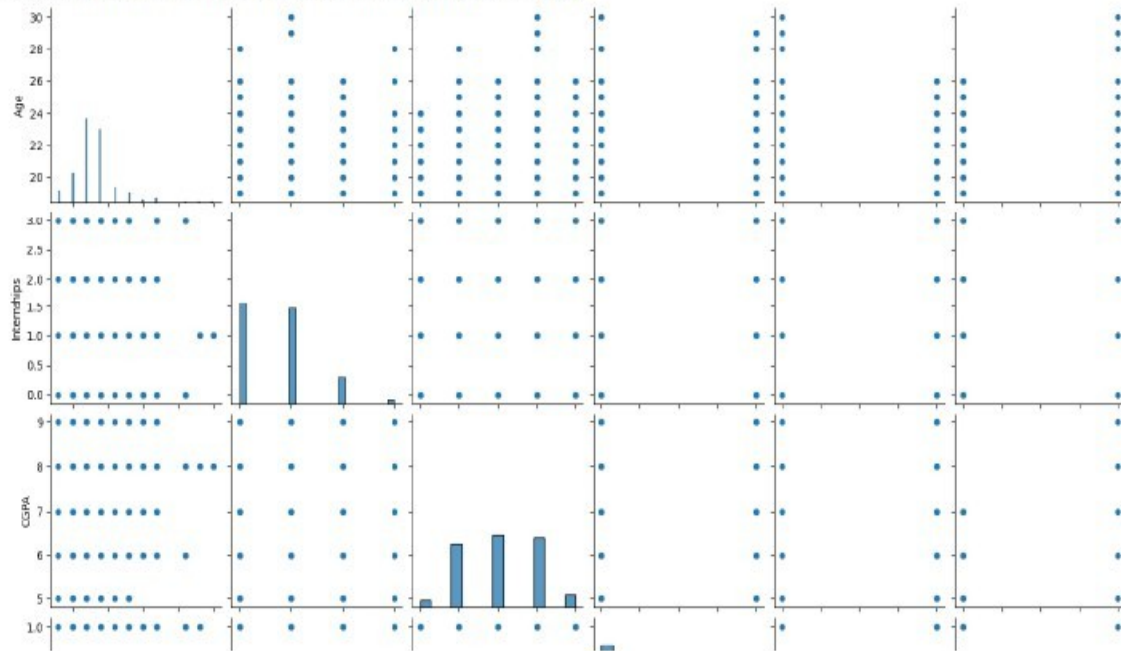
```
df = pd.read_csv("collegePlace.csv")
```

```
df.shape
```

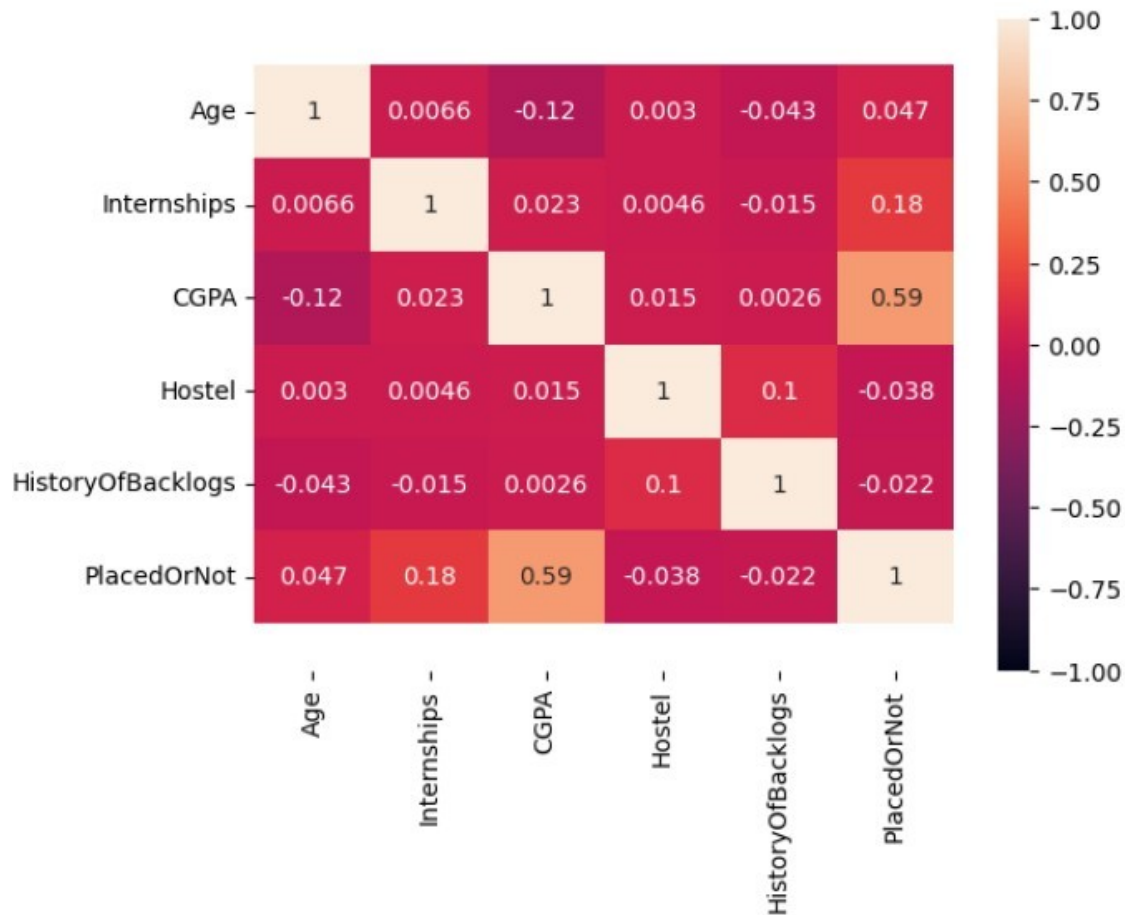
```
(2966, 8)
```

```
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x7fa7fbc4df10>



```
ax = sns.heatmap(corr, vmin = -1, vmax = 1, annot = True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()
corr
```



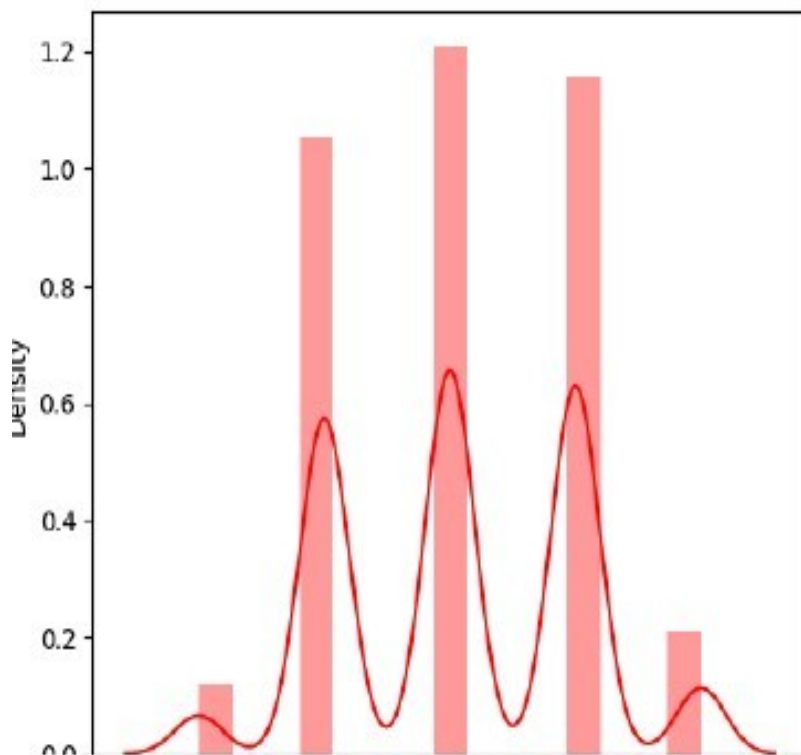
	Age	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
Age	1.000000	0.006552	-0.119787	0.003042	-0.042586	0.046943
Internships	0.006552	1.000000	0.023496	0.004617	-0.015118	0.179334
CGPA	-0.119787	0.023496	1.000000	0.014991	0.002576	0.588648
Hostel	0.003042	0.004617	0.014991	1.000000	0.103506	-0.038182
HistoryOfBacklogs	-0.042586	-0.015118	0.002576	0.103506	1.000000	-0.022337
PlacedOrNot	0.046943	0.179334	0.588648	-0.038182	-0.022337	1.000000

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['CGPA'],color='r')
```

❖ Data Preparation

1. As we have understood how the data is, let's pre-process the collected data.
The download data set is not suitable for training the machine
2. Learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

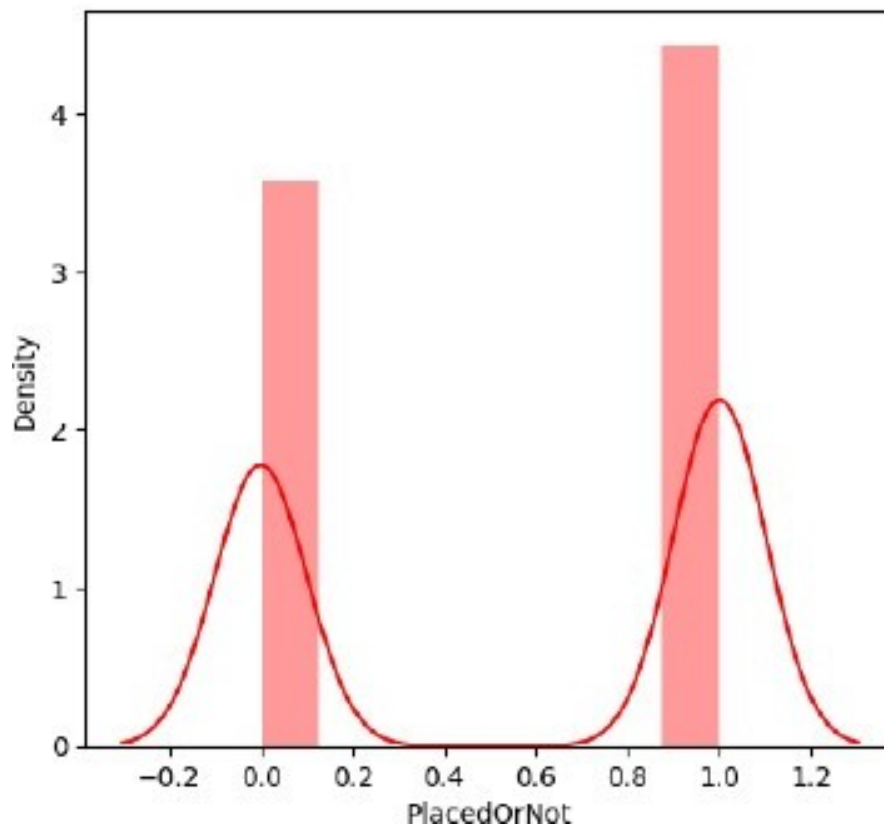
- ❖ Handling Missing data
- ❖ Handling Categorical data
- ❖ Handling missing data



❖ Handling missing values

Let's find the shape of our dataset first. To find the shape of our df.shape method is used. To find the data type, df.info() function

data, the is used.



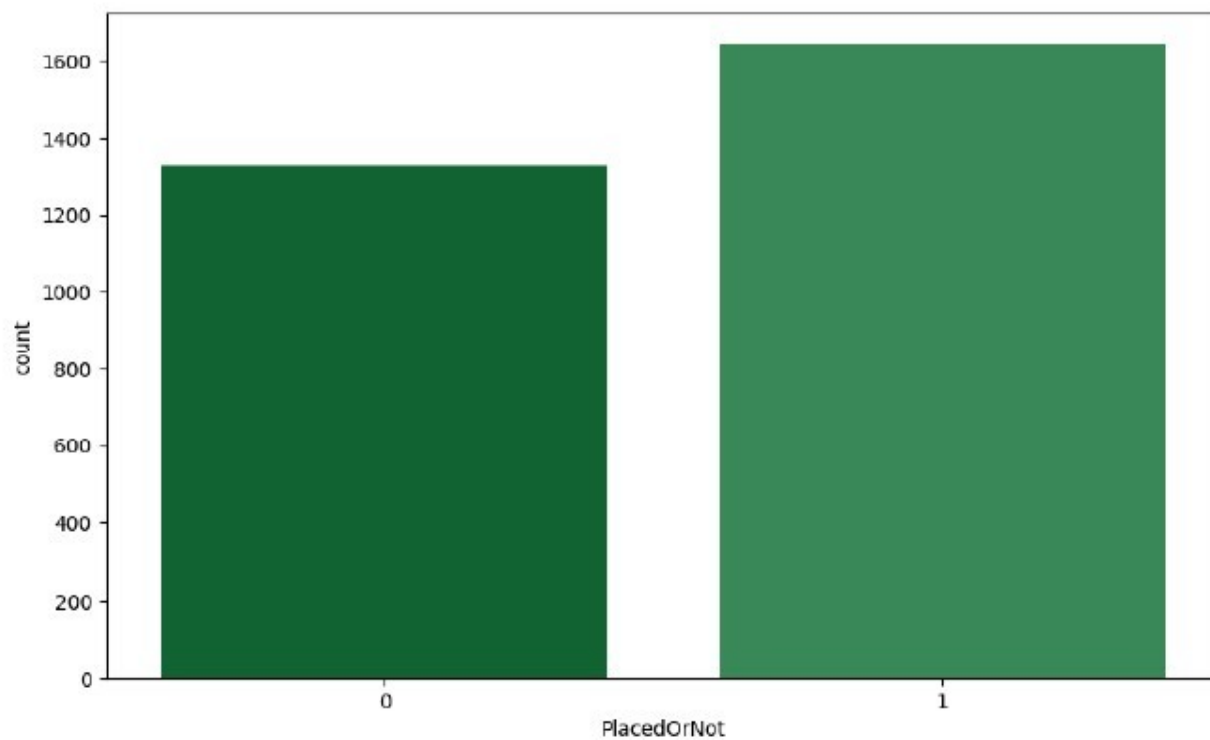
```
# how many placed?  
plt.figure(figsize = (10,6),dpi = 100)
```



```
# setting the different color palette
color_palette = sns.color_palette("BuGn_r")
sns.set_palette(color_palette)

sns.countplot(x = "PlacedOrNot", data = df)

plt.show()
```



df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2966 entries, 0 to 2965
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                   2966 non-null   int64
1   Gender                2966 non-null   object
2   Stream                2966 non-null   object
3   Internships           2966 non-null   int64
4   CGPA                  2966 non-null   int64
5   Hostel                2966 non-null   int64
6   HistoryOfBacklogs     2966 non-null   int64
7   PlacedOrNot           2966 non-null   int64
dtypes: int64(6), object(2)
memory usage: 185.5+ KB
```

df.isnull().sum()

```
Age                0
Gender              0
Stream             0
Internships        0
CGPA               0
Hostel             0
HistoryOfBacklogs  0
PlacedOrNot        0
dtype: int64
```

	Age	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
count	2966.000000	2966.000000	2966.000000	2966.000000	2966.000000	2966.000000
mean	21.485840	0.703641	7.073837	0.269049	0.192178	0.552596
std	1.324933	0.740197	0.967748	0.443540	0.394079	0.497310

```
df = df.replace(['Male'],[0])
df = df.replace(['Female'],[1])
```

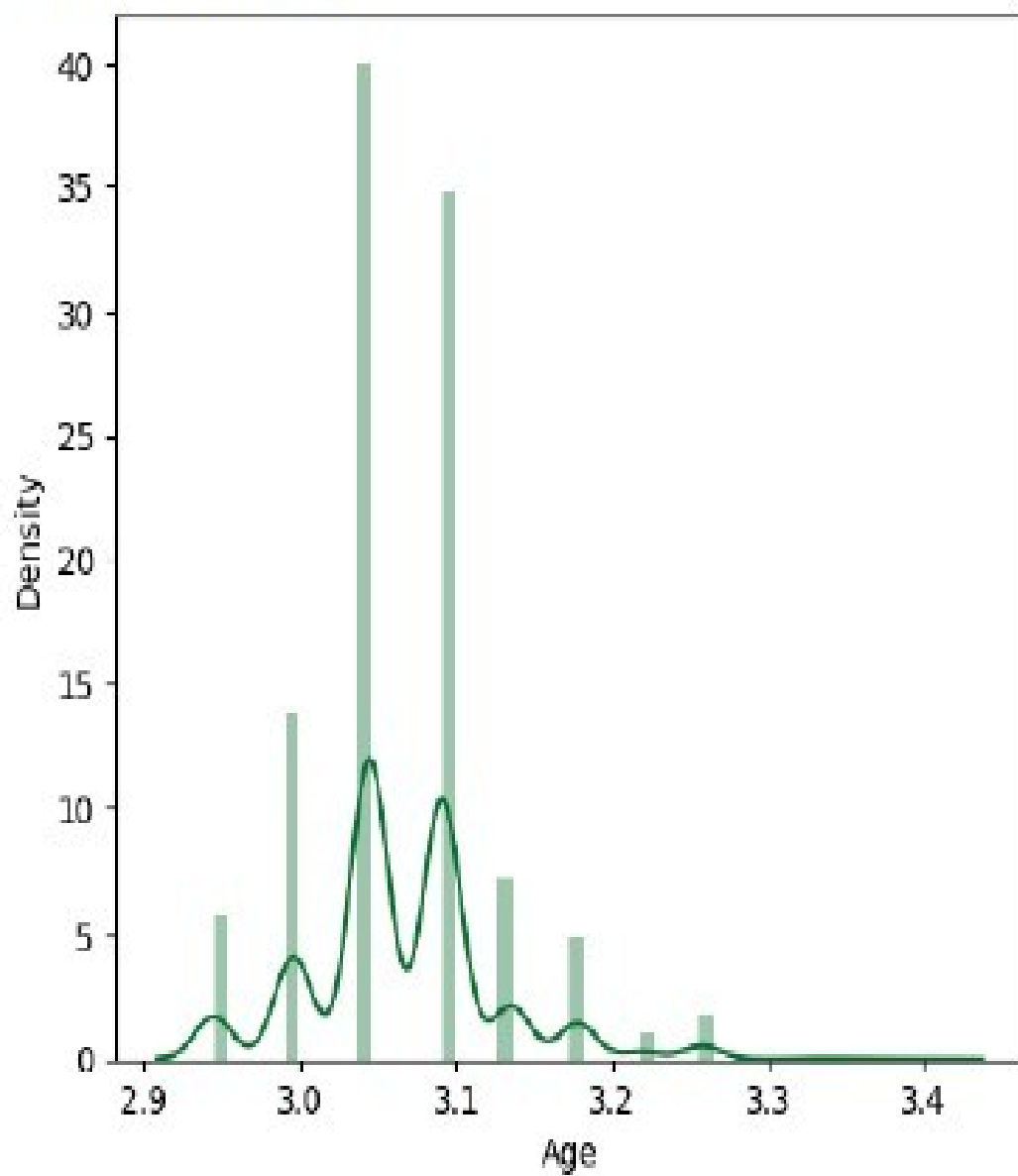
```
df = df.replace(['Computer Science','Information Technology','Electronics And Communication','Mechanical','Electrical','Civil'],[0,1,2,3,4,5])
```

df

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

2966 rows x 8 columns

```
sns.distplot(feature)
```



```
scaler = StandardScaler()
```

```
scaler.fit(X)
```

```
StandardScaler
StandardScaler()
```

```
standardized_data = scaler.transform(X)
```

```
print(standardized_data)
```

```
[[ 0.38813058 -0.44540301  0.04008175  0.40044544  0.95719068  2.05024603]
 [-0.36675158  2.24515772 -1.14874288 -0.95077319 -0.07631043  2.05024603]
 [ 0.38813058  2.24515772 -0.55433057  0.40044544 -1.10981154 -0.48774634]
 ...
 [ 0.38813058 -0.44540301 -0.55433057  0.40044544 -0.07631043 -0.48774634]
 [ 0.38813058 -0.44540301 -1.14874288  0.40044544 -0.07631043 -0.48774634]
 [ 1.14301273 -0.44540301  1.82331869 -0.95077319  0.95719068 -0.48774634]]
```

```
X = standardized_data
```

```
Y = df['PlacedOrNot']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify=Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(2966, 6) (2372, 6) (594, 6)
```

```
classifier = svm.SVC(kernel='linear')
```

```
classifier.fit(X_train, Y_train)
```

```
SVC
SVC(kernel='linear')
```

- **Training the model in multiple algorithms**

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

- **SVM model**

A function named Support vector machine is created and train and test data are passed as the parameters. Inside the function, SVM Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with . predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
#testing accuracy
X_test_prediction = classifier.predict(X_test)
y_pred= accuracy_score(X_test_prediction, Y_test)
y_pred

0.7794612794612794

X_test

array([[ -0.36675158,  2.24515772,  0.04008175,  1.75166407, -0.07631043,
        -0.48774634],
       [ 0.38813058, -0.44540301,  0.63449406, -0.95077319, -0.07631043,
        -0.48774634],
       [ 1.89789488, -0.44540301, -0.55433057,  0.40044544, -0.07631043,
        -0.48774634],
       ...,
       [-1.12163373, -0.44540301,  0.63449406,  0.40044544,  1.99069179,
        -0.48774634],
       [ 0.38813058, -0.44540301, -1.14874288,  0.40044544, -1.10981154,
        -0.48774634],
       [ 0.38813058,  2.24515772, -0.55433057,  0.40044544,  0.95719068,
        -0.48774634]])

#training accuracy
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy score of the training data : ', training_data_accuracy)

Accuracy score of the training data :  0.7685497470489039
```

```

print("---Results---\nk: {}\nScore: {}".format(best_k, best_score))
## Instantiate the models
knn = KNeighborsClassifier(n_neighbors=best_k["Regular"])
## Fit the model to the training set
knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
testd = accuracy_score(knn_pred, Y_test)

```

```

---Results---
K: {'Regular': 7}
Score: {'Regular': 86.19528619528619}

```

knn_pred

```

array([[1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
        1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
        1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
        0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
        1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0,
        1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
        0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
        0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
        1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,

        1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
        1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
        0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
        0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
        0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
        0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
        0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1,
        1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
        1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
        1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1])

```



```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from tensorflow.keras import layers
```

```
classifier = Sequential()
```

```
#add input Layer and first hidden Layer
classifier.add(keras.layers.Dense(6,activation = 'relu', input_dim = 6))
classifier.add(keras.layers.Dropout(0.50))
#add 2nd hidden Layer
classifier.add(keras.layers.Dense(6,activation = 'relu'))
classifier.add(keras.layers.Dropout(0.50))
```

```
#final or output Layer
classifier.add(keras.layers.Dense(1, activation = 'sigmoid'))
```

```
#fitting the model
classifier.fit(X_train, Y_train, batch_size = 20, epochs = 100)
```

```
Epoch 1/100
```

```
119/119 [=====] - 4s 4ms/step - loss: 0.8081 - accuracy: 0.4810
```

```
Epoch 2/100
```

```
119/119 [=====] - 0s 2ms/step - loss: 0.7411 - accuracy: 0.5287
```

```
Epoch 3/100
```

```
119/119 [=====] - 0s 2ms/step - loss: 0.7025 - accuracy: 0.5519
```


Epoch 5/100
119/119 [=====] - 0s 2ms/step - loss: 0.6692 - accuracy: 0.5877
Epoch 6/100
119/119 [=====] - 0s 2ms/step - loss: 0.6402 - accuracy: 0.6033
Epoch 7/100
119/119 [=====] - 0s 2ms/step - loss: 0.6386 - accuracy: 0.5847
Epoch 8/100
119/119 [=====] - 0s 2ms/step - loss: 0.6177 - accuracy: 0.5995
Epoch 9/100
119/119 [=====] - 0s 2ms/step - loss: 0.6097 - accuracy: 0.5974
Epoch 10/100
119/119 [=====] - 0s 2ms/step - loss: 0.5972 - accuracy: 0.6193
Epoch 11/100
119/119 [=====] - 0s 2ms/step - loss: 0.5790 - accuracy: 0.6265
Epoch 12/100
119/119 [=====] - 0s 2ms/step - loss: 0.5680 - accuracy: 0.6320
Epoch 13/100
119/119 [=====] - 0s 2ms/step - loss: 0.5626 - accuracy: 0.6505
Epoch 14/100
119/119 [=====] - 0s 2ms/step - loss: 0.5468 - accuracy: 0.6476
Epoch 15/100
119/119 [=====] - 0s 2ms/step - loss: 0.5271 - accuracy: 0.6678
Epoch 16/100
119/119 [=====] - 0s 2ms/step - loss: 0.5187 - accuracy: 0.6906
Epoch 17/100
119/119 [=====] - 0s 2ms/step - loss: 0.5196 - accuracy: 0.6754
Epoch 18/100
119/119 [=====] - 0s 2ms/step - loss: 0.5177 - accuracy: 0.6703
Epoch 19/100
119/119 [=====] - 0s 2ms/step - loss: 0.5132 - accuracy: 0.6728
Epoch 20/100
119/119 [=====] - 0s 2ms/step - loss: 0.4933 - accuracy: 0.6939
Epoch 21/100
119/119 [=====] - 0s 2ms/step - loss: 0.4936 - accuracy: 0.6960
Epoch 22/100
119/119 [=====] - 0s 2ms/step - loss: 0.4856 - accuracy: 0.7049
Epoch 23/100
119/119 [=====] - 0s 2ms/step - loss: 0.4887 - accuracy: 0.6994
Epoch 24/100
119/119 [=====] - 0s 2ms/step - loss: 0.4937 - accuracy: 0.6825
Epoch 25/100
119/119 [=====] - 0s 2ms/step - loss: 0.4788 - accuracy: 0.6914
Epoch 26/100
119/119 [=====] - 0s 2ms/step - loss: 0.4783 - accuracy: 0.7032
Epoch 27/100
119/119 [=====] - 0s 2ms/step - loss: 0.4869 - accuracy: 0.6884
Epoch 28/100
119/119 [=====] - 0s 2ms/step - loss: 0.4929 - accuracy: 0.6762

pred

[False],

```
input_data = [[22,0,2,1,8,1]]
```

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script
- Run the web application

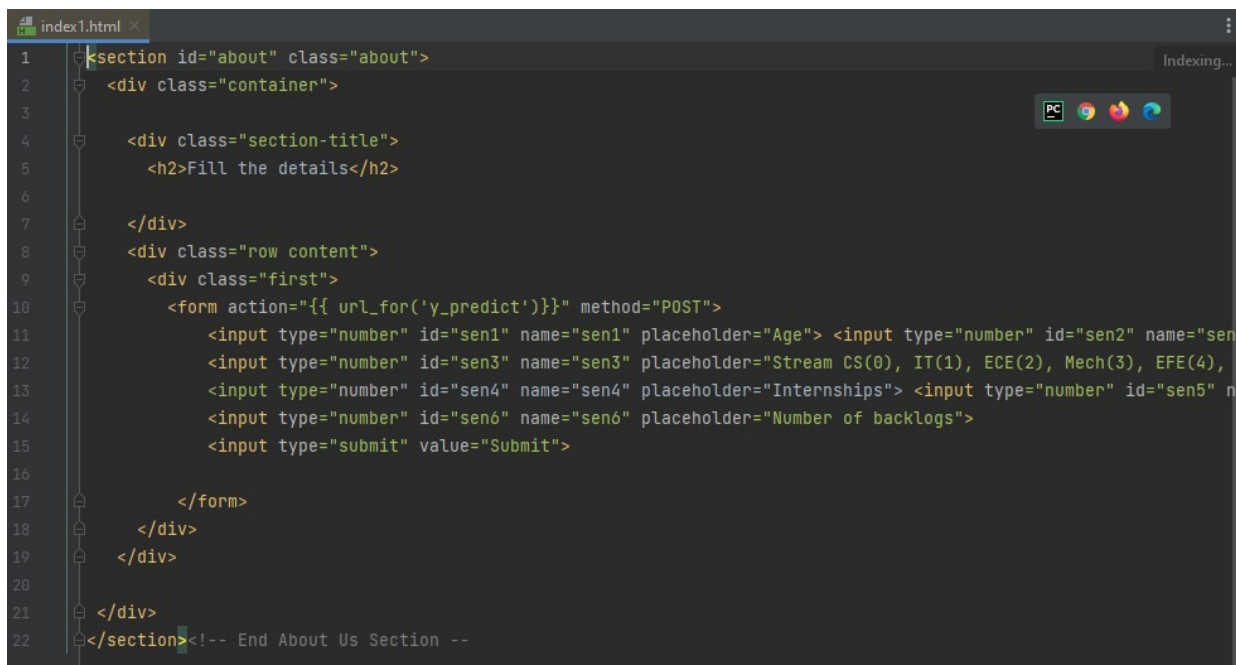
● Building Html Pages

project create HTML files

1)index.html 2)index1.html

3)secondpage.html

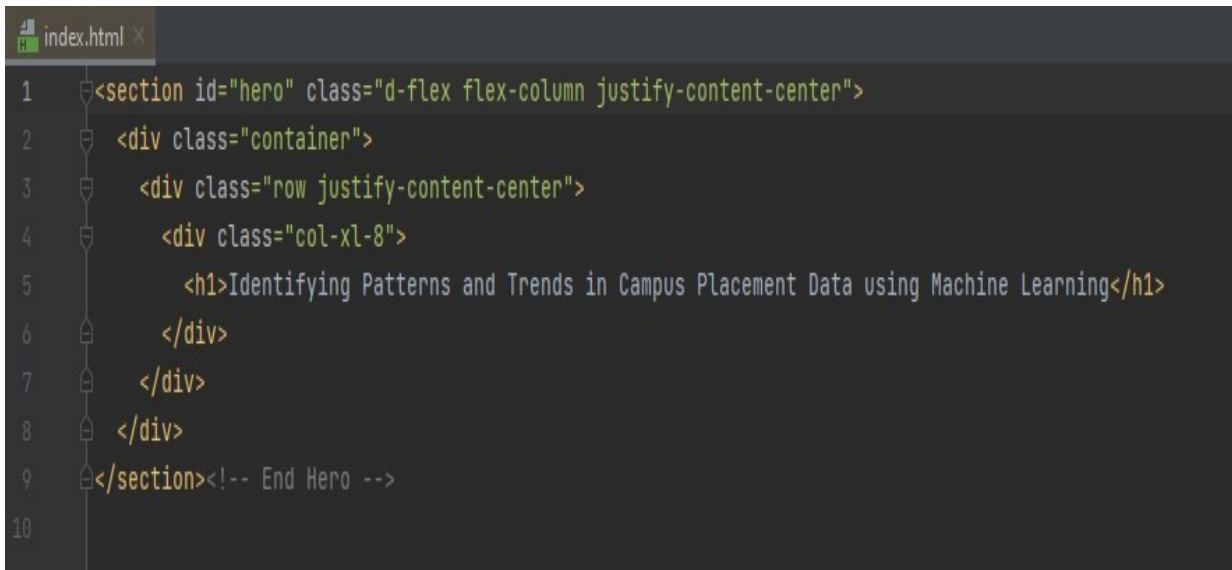
and save it in templates folder



```
1 <section id="about" class="about">
2   <div class="container">
3
4     <div class="section-title">
5       <h2>Fill the details</h2>
6
7     </div>
8     <div class="row content">
9       <div class="first">
10        <form action="{{ url_for('y_predict')}}" method="POST">
11          <input type="number" id="sen1" name="sen1" placeholder="Age"> <input type="number" id="sen2" name="sen2" placeholder=" " >
12          <input type="number" id="sen3" name="sen3" placeholder="Stream CS(0), IT(1), ECE(2), Mech(3), EFE(4), " >
13          <input type="number" id="sen4" name="sen4" placeholder="Internships"> <input type="number" id="sen5" name="sen5" placeholder=" " >
14          <input type="number" id="sen6" name="sen6" placeholder="Number of backlogs">
15          <input type="submit" value="Submit">
16
17        </form>
18      </div>
19    </div>
20  </div>
21 </section><!-- End About Us Section --
22
```

● Building Html Pages

❖ The below code is written to fetch the details from the user using `<form>` tag. A submit button is provided at the end which navigates to the prediction page upon clicking.

A screenshot of a code editor with a dark background. The editor has a tab at the top labeled 'index.html' with a small icon to its left. On the left side, there is a vertical line with markers for each line of code, numbered 1 through 10. The code is as follows:

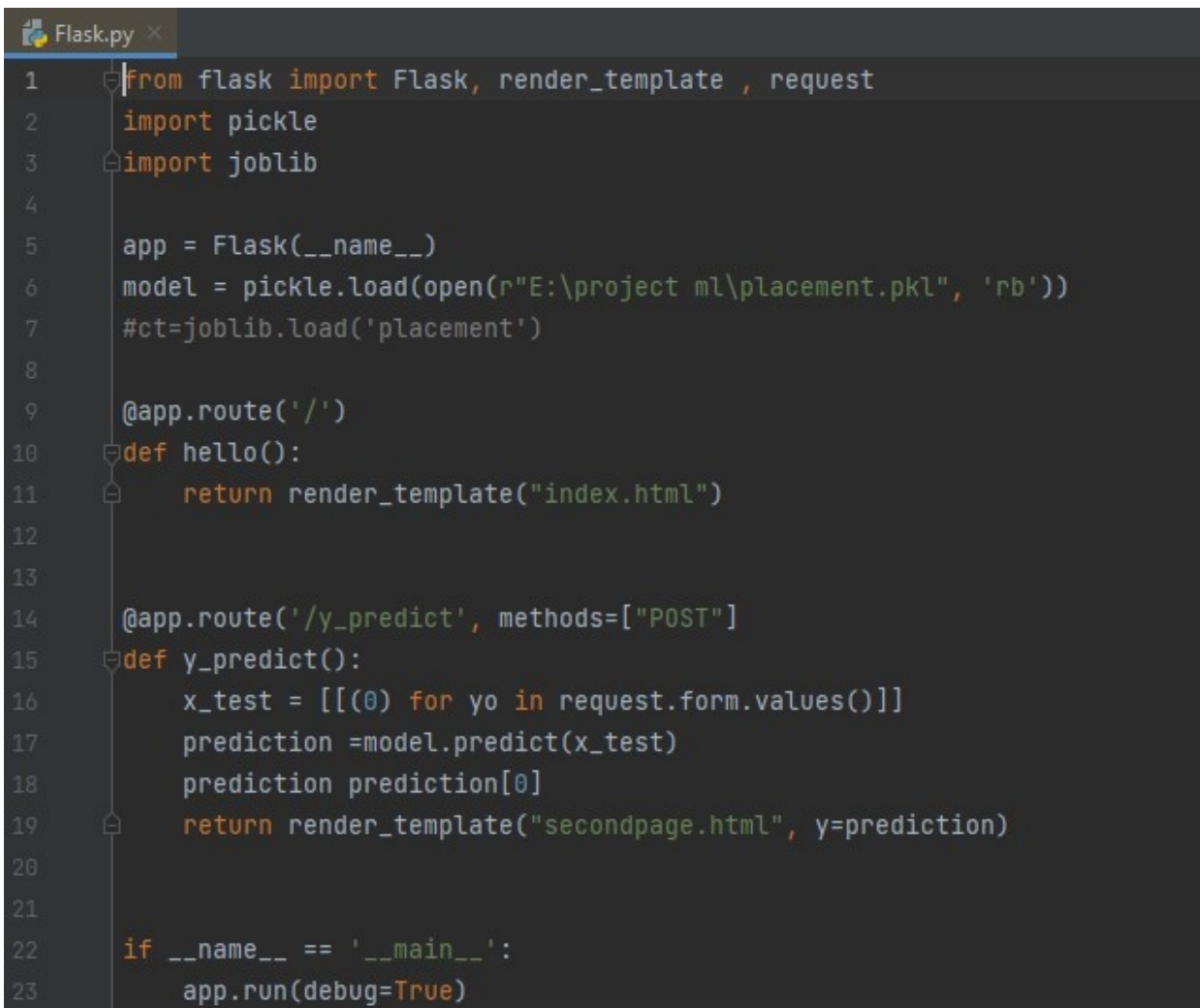
```
1 <section id="hero" class="d-flex flex-column justify-content-center">
2   <div class="container">
3     <div class="row justify-content-center">
4       <div class="col-xl-8">
5         <h1>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h1>
6       </div>
7     </div>
8   </div>
9 </section><!-- End Hero -->
```

❖ The code for secondpage.html is as shown below. The code includes a section that provides the output on screen

Build Python code

- Import the libraries

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application.

A screenshot of a code editor window titled 'Flask.py'. The code is written in Python and implements a Flask web application. It includes imports for Flask, render_template, request, pickle, and joblib. The application has two routes: a root route '/' that renders 'index.html', and a POST route '/y_predict' that takes form data, uses a loaded model to make a prediction, and renders 'secondpage.html' with the prediction. The code also includes a standard Flask main block to run the application in debug mode.

```
1  from flask import Flask, render_template , request
2      import pickle
3      import joblib
4
5      app = Flask(__name__)
6      model = pickle.load(open(r"E:\project ml\placement.pkl", 'rb'))
7      #ct=joblib.load('placement')
8
9      @app.route('/')
10     def hello():
11         return render_template("index.html")
12
13
14     @app.route('/y_predict', methods=["POST"])
15     def y_predict():
16         x_test = [[() for yo in request.form.values()]]
17         prediction =model.predict(x_test)
18         prediction prediction[0]
19         return render_template("secondpage.html", y=prediction)
20
21
22     if __name__ == '__main__':
23         app.run(debug=True)
```

- ❖ In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.
 - ❖ This below code retrieves the value from UI
 - ❖ Here in the above code we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model. Predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier
- 1) Navigate to the folder where your python script is.
 - 2) Now type "python app.py" command
 - 3) Navigate to the local host where you can view your web page.
- ❖ Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 146-359-021
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

