Dear Students,

Greetings from SmartBridge,

Your team has successfully enrolled for the project.
please find the team details below

**Team ID:** NM2023TMID35448

**Team Size:** 4

**Team Leader:** SURIYA A

**Team Member:** NARMATHA V

**Team Member:** SARATH S

**Team Member:** ARULMURUGAN A

Regards,
Team SmartBridge

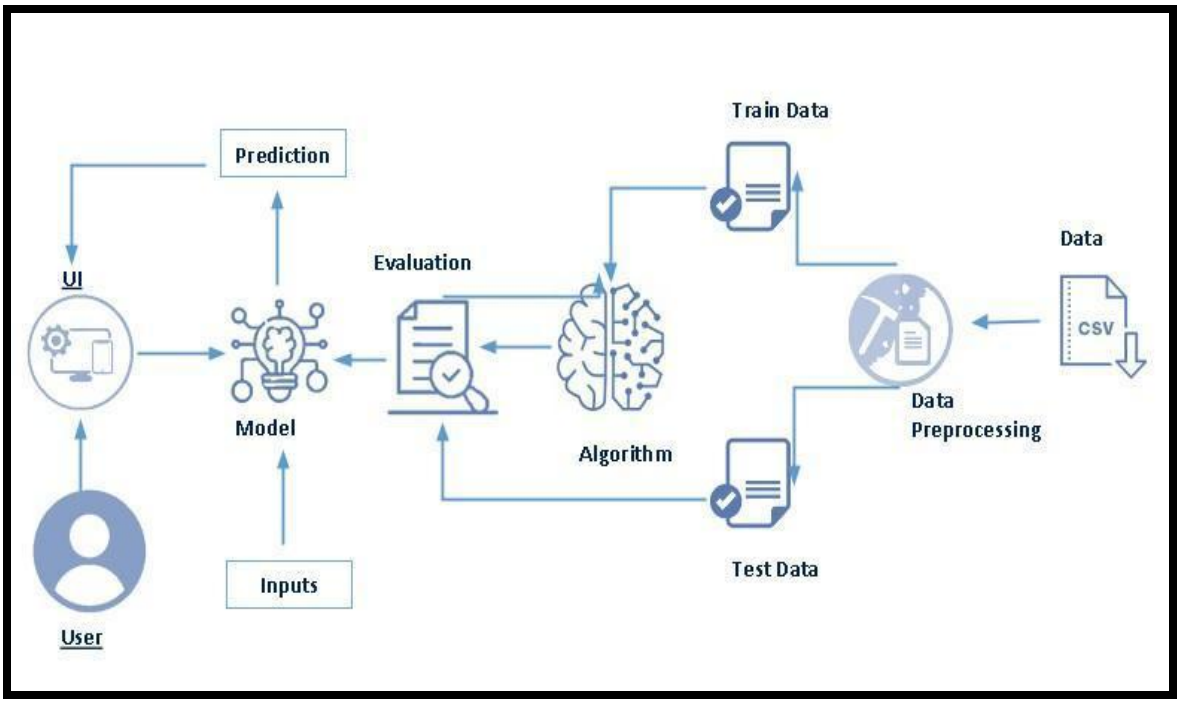
# Identifying Patterns and Trends in Campus Placement Data using Machine Learning

## 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 colleges 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.

## Technical Architecture:




## 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

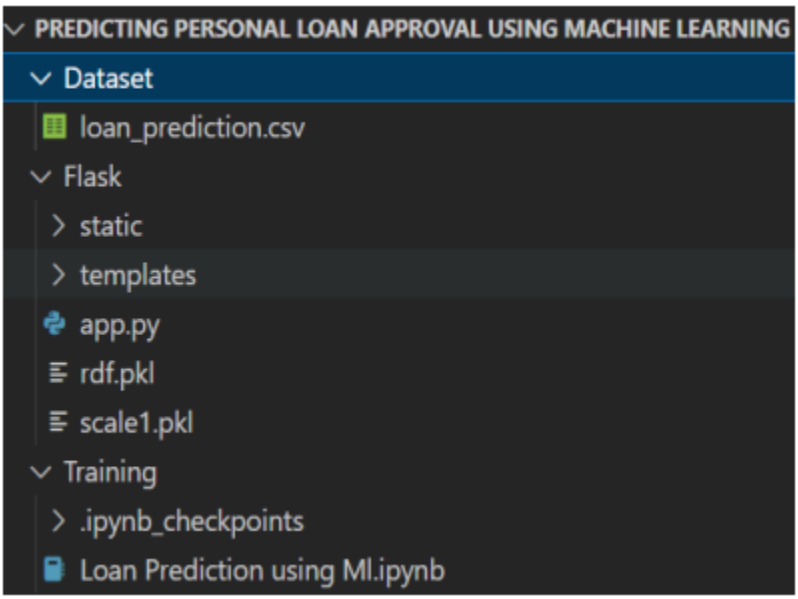
To accomplish this, we have to complete all the activities listed below,

- 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 Flow:**

Create the Project folder which contains files as shown below




- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting
- rdf.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file

**Milestone 1: Define Problem / Problem Understanding**

- **Activity 1: Specify the business problem**

  Refer Project Description

- **Activity 2: Business requirements**

The business requirements for a project aimed at "Identifying Patterns and Trends in Campus Placement Data using Machine Learning" would likely include the following:

- ❖ Access to campus placement data: The project would require access to data on student performance, qualifications, and job placement outcomes. This data would need to be collected, cleaned, and prepared for analysis.

- ❖ Machine learning expertise: The project would require individuals with expertise in machine learning, data science and statistical analysis to develop and implement the algorithms and models needed to analyze the data.

- ❖ Data storage and management: The project would require a robust and secure data storage and management system to store and organize the large amounts of data used in the analysis.

- ❖ Infrastructure for model deployment: The project would require infrastructure for deploying the models and algorithms developed, including hardware, software, and cloud-based resources.

- **Activity 3: Literature Survey (Student Will Write)**
  - ❖ There have been several studies that have used machine learning techniques to identify patterns and trends in campus placement data.
  - ❖ One study by authors P. K. Rajesh and Dr. G. R. Suresh, published in the International Journal of Computer Science and Mobile Computing in 2015, used k-means clustering and decision trees to analyze campus placement data and identify patterns that could be used to predict placement outcomes.
  - ❖ Another study by authors V.V. Kulkarni and K.S. Patil, published in the International Journal of Engineering Research and Technology in 2012, used decision tree and neural network algorithms to analyze campus placement data and identify factors that influence student placement.
  - ❖ A study by authors S.S. Bhosale, S.S. Raut, and D.S. Kulkarni, published in the International Journal of Emerging Research in Management & Technology in 2013, used decision tree and Naive Bayes algorithms to analyze campus placement data and predict student placement outcomes.
  - ❖ A study by authors S.S. Bhosale, S.S. Raut, and D.S. Kulkarni, published in the International Journal of Emerging Research in Management & Technology in 2013, used decision tree and Naive Bayes algorithms to analyze campus placement data and predict student placement outcomes.
  - ❖ In general, these studies found that machine learning techniques were effective at identifying patterns and trends in campus placement data, and could be used to predict student placement outcomes with high accuracy.
  - ❖ It's important to note that all these studies are quite old now and you might find more recent studies and new techniques which can be useful for your project.

- **Activity 4: Social or Business Impact.**

    - ❖ The business impact of a project that uses machine learning to identify patterns and trends in campus placement data could be significant. By analyzing data on factors such as student performance, qualifications, and job placement outcomes, the project could help organizations make more informed decisions about recruiting and hiring new graduates.

**Milestone 2: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

- **Activity 1: Collect the dataset**

    There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset. Link:

    https://www.kaggle.com/code/neesham/prediction-of-placements/data

    - **Activity 1.1: 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
```

- **Activity 1.2: Read the Dataset**

  Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

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

| | Age | Gender | Stream | Internships | CGPA | Hostel | HistoryOfBacklogs | PlacedOrNot |
|---|---|---|---|---|---|---|---|---|
| 0 | 22 | Male | Electronics And Communication | 1 | 8 | 1 | 1 | 1 |
| 1 | 21 | Female | Computer Science | 0 | 7 | 1 | 1 | 1 |
| 2 | 22 | Female | Information Technology | 1 | 6 | 0 | 0 | 1 |
| 3 | 21 | Male | Information Technology | 0 | 8 | 0 | 1 | 1 |
| 4 | 22 | Male | Mechanical | 0 | 8 | 1 | 0 | 1 |

- **Activity 2: Data Preparation**

  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 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

.

  - **Activity 2.1: Handling missing values**

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

```
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()
```

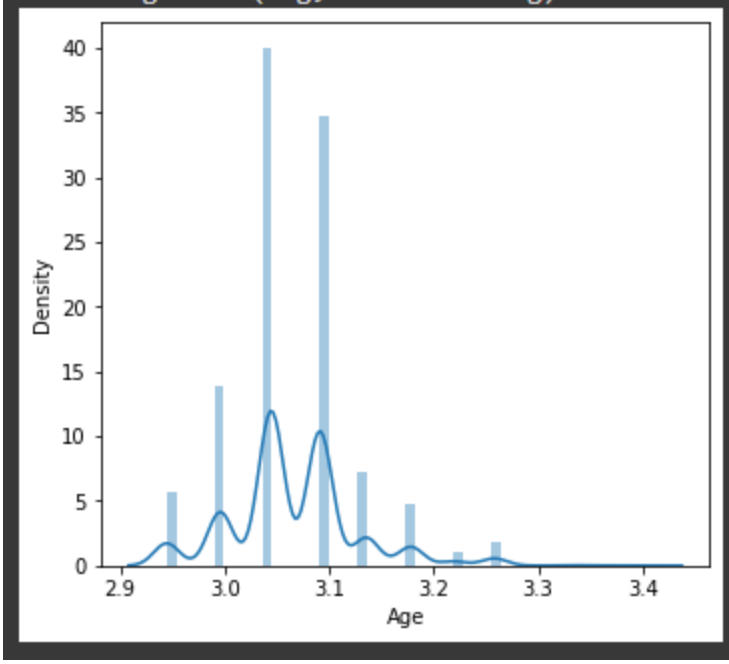
```
Age                0
Gender             0
Stream             0
Internships        0
CGPA               0
Hostel             0
HistoryOfBacklogs  0
PlacedOrNot        0
dtype: int64
```

- **Activity 2.2: Handling outliers**

```
def transformationplot(feature):
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    sns.distplot(feature)

transformationplot(np.log(df['Age']))
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)
```




- **Activity 2.3: Handling Categorical Values**

  As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

  To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using replacements as the distinct values are less.

```
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 = df.drop(['Hostel'], axis=1)
```

```
df
```

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

2966 rows × 7 columns

## Milestone 3: Exploratory Data Analysis

- **Activity 1: Visual analysis**

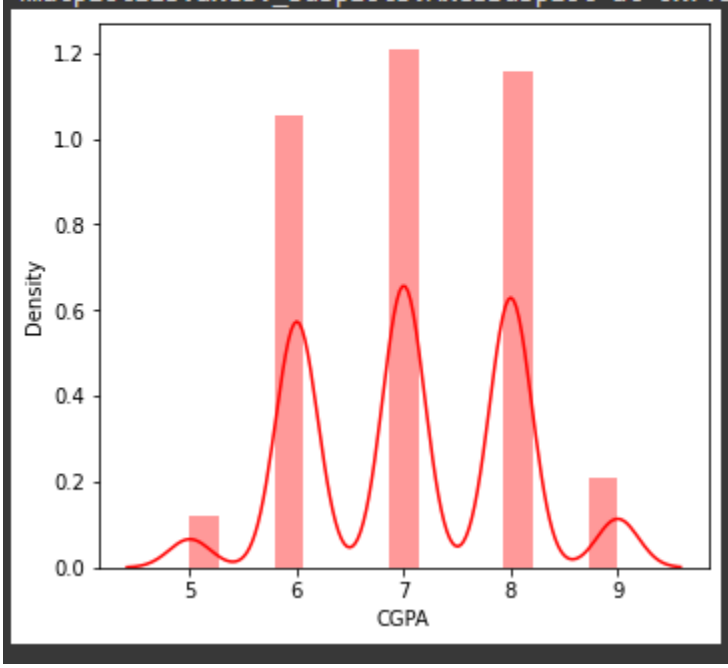
  Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

  - **Activity 1.1 : Univariate analysis**

    In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

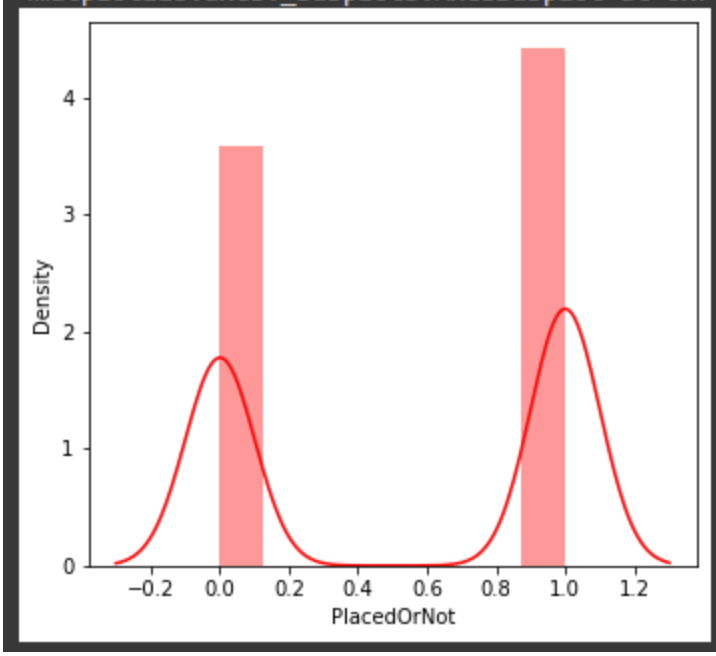
```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['CGPA'],color='r')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:261
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f5463e50d00>
```



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

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: Futu
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f5463d95790>
```

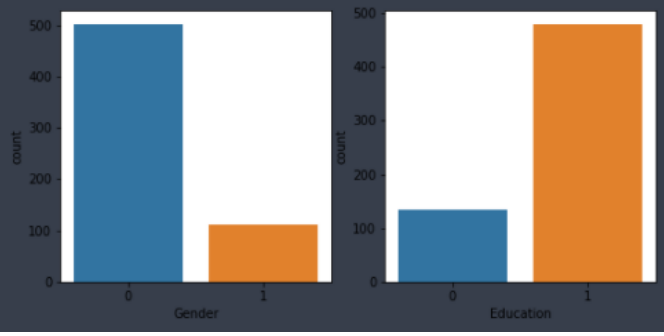



- **Activity 1.2: Bivariate analysis**

  Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```
#plotting the count plot
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(data['Gender'])
plt.subplot(1,4,2)
sns.countplot(data['Education'])
plt.show()
```

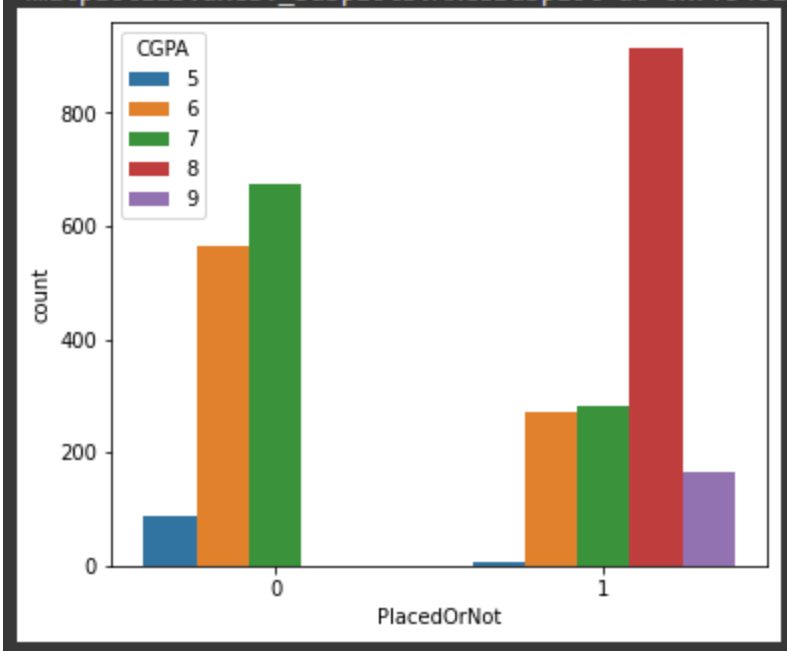
```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
```




- **Activity 1.3: Multivariate analysis**
  In simple words, multivariate analysis is to find the relation between multiple features. Here we have used swarmplot from the seaborn package.

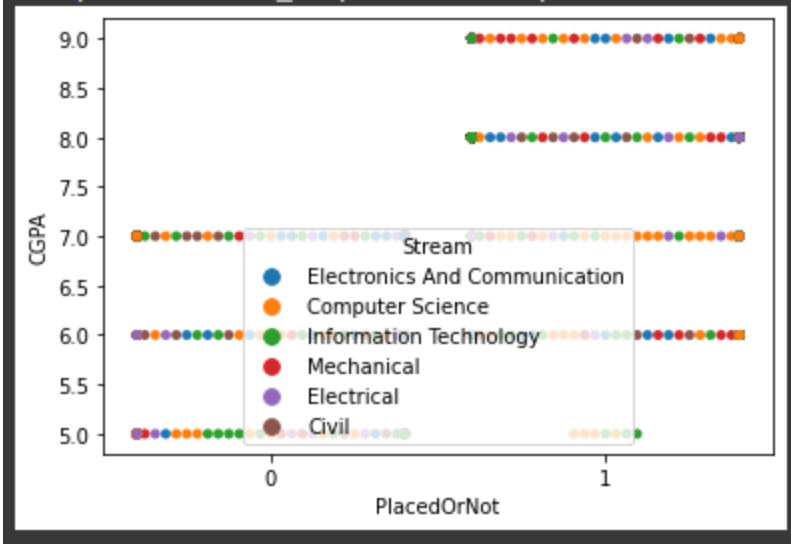
```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(df["PlacedOrNot"],hue=df['CGPA'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: Futur
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f5461cf85b0>
```



```
sns.swarmplot(df['PlacedOrNot'],df['CGPA'],hue=df['Stream'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36:
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:129
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:129
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f5463d06df0>
```




- **Activity 2: Scaling the data**

  Scaling is one the important processes we have to perform on the dataset, because data measures in different ranges can lead to mislead in prediction Models such as KNN, Logistic regression needs scaled data, as they follow distance based method and Gradient Descent concept.

```
# perfroming feature Scaling operation using standard scaller on X part of the dataset becau
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)

x_bal = pd.DataFrame(x_bal,columns=names)
```

We will perform scaling only on the input values.Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

- **Activity 3: Splitting the data into train and test**

  Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing.data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
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)
```

## Milestone 4: Model Building

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 a few  algorithms. The best model is saved based on its performance.

- **Activity 1: 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.

  - **Activity 1.1: SVM model**

    A function named Support vector machine is created and train and test data are passed as the parameters. Inside the function, SVMClassifier 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.

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

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

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

```
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
```

- **Activity 1.2: KNN model**

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

```
best_k = {"Regular":0}
best_score = {"Regular":0}
for k in range(3, 50, 2):

    ## Using Regular training set
    knn_temp = KNeighborsClassifier(n_neighbors=k)              # Instantiate the model
    knn_temp.fit(X_train, Y_train)                              # Fit the model to the training set
    knn_temp_pred = knn_temp.predict(X_test)                    # Predict on the test set
    score = metrics.accuracy_score(Y_test, knn_temp_pred) * 100 # Get accuracy
    if score >= best_score["Regular"] and score < 100:          # Store best params
        best_score["Regular"] = score
        best_k["Regular"] = k
```

```
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}
```

- **Activity 1.3: Artificial neural network model**

  We will also be using a neural network to train the model.

```
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'))
```

```
[ ] #Compiling the model

    loss_1 = tf. keras.losses.BinaryCrossentropy()

    classifier.compile(optimizer = 'Adam', loss = loss_1 , metrics = ['accuracy'])
```

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

## Milestone 5: Model Deployment

- **Activity 1:Save the best model**

  Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future

```
[ ] import pickle

    pickle.dump(knn,open("placement.pkl",'wb'))
    model = pickle.load(open('placement.pkl', 'rb'))
```

- **Activity 2: Integrate with Web Framework**

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

- **Activity 2.1: Building Html Pages**

  For this project create one HTML file namely

  1)index.html

  2)index1.html

  3)secondpage.html

  and save it in templates folder

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

- **Activity 2.2: Building Html Pages(part-2)**

❖ 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.

```
<section id="about" class="about">
  <div class="container">

    <div class="section-title">
      <h2>Fill the details</h2>

    </div>
    <div class="row content">
      <div class="first">
        <form action="{{ url_for('y_predict')}}" method="POST">
            <input type="number" id="sen1" name="sen1" placeholder="Age">
            <input type="number" id="sen2" name="sen2" placeholder="Gender M(0),F(0)">
            <input type="number" id="sen3" name="sen3" placeholder="Stream CS(0),IT(1),ECE(2),Mech(3),EEE(4),Civil
            <input type="number" id="sen4" name="sen4" placeholder="Internships">
            <input type="number" id="sen5" name="sen5" placeholder="CGPA">
            <input type="number" id="sen6" name="sen6" placeholder="Number of backlogs">
            <input type="submit" value="Submit">

          </form>
      </div>
    </div>

  </div>
</section><!-- End About Us Section -->
```

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

```
<section id="hero" class="d-flex flex-column justify-content-center">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-xl-8">
        <h1>The Prediction is : {{y}}</h1>
        <h3> 0 represents Not-Placed </h3>
        <h3> 1 represents Placed<h2>


      </div>
    </div>
  </div>
</section><!-- End Hero -->
```

**Activity 3: Build Python code**

- **Activity 3.1: 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. Flask

constructor takes the name of the current module (__name__) as argument.

```
from flask import Flask, render_template , request
app=Flask(__name__)
import pickle
import joblib
model=pickle.load(open("placement123.pkl",'rb'))
ct=joblib.load('placement')
```

- **Activity 3.2 : Render HTML page**

```
@app.route('/')
def hello():
    return render_template("index.html")
```

❖ Here we will be using a declared constructor to route to the HTML page which we have created earlier.

❖ 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

```
@app.route('/guest' , methods = ["POST"])
def Guest():

    sen1=request.form["sen1"]
    sen2=request.form["sen2"]
    sen3=request.form["sen3"]
    sen4=request.form["sen4"]
    sen5=request.form["sen5"]
    sen6=request.form["sen6"]



@app.route('/y_predict' , methods = ["POST"])
def y_predict():
    x_test = [[(yo) for yo in request.form.values()]]

    prediction =model.predict(x_test)


    prediction = prediction[0]


    return render_template("secondpage.html",y=prediction)
```

- 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.

- **Activity 3.3: Main Function**

```
app.run(debug=True)
```

1)Open anaconda prompt from the start menu

2)Navigate to the folder where your python script is.

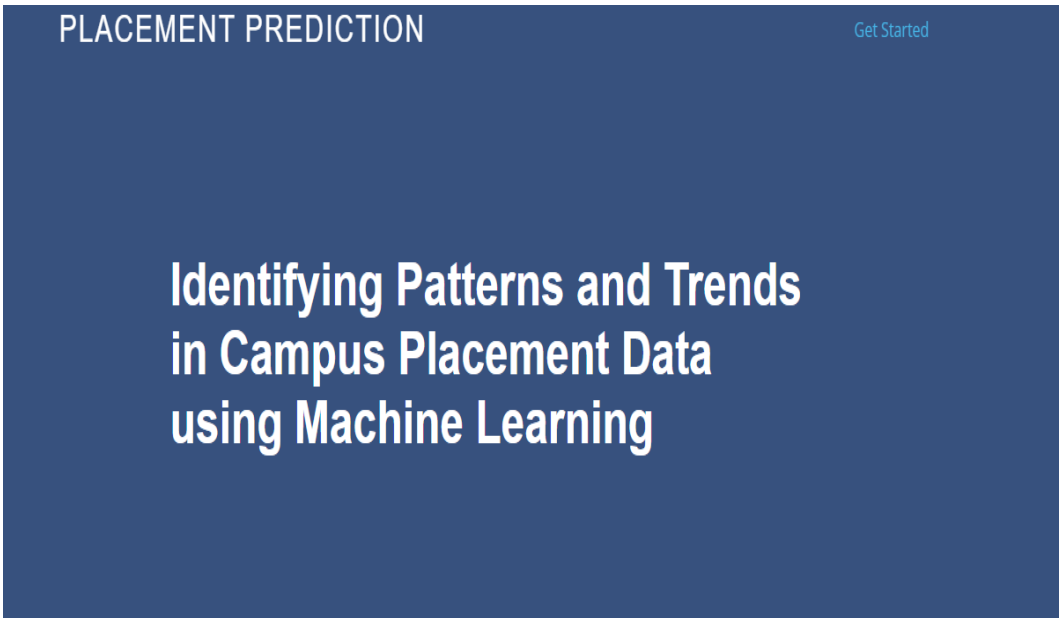
3)Now type “python app.py” command

4)Navigate to the localhost where you can view your web page.

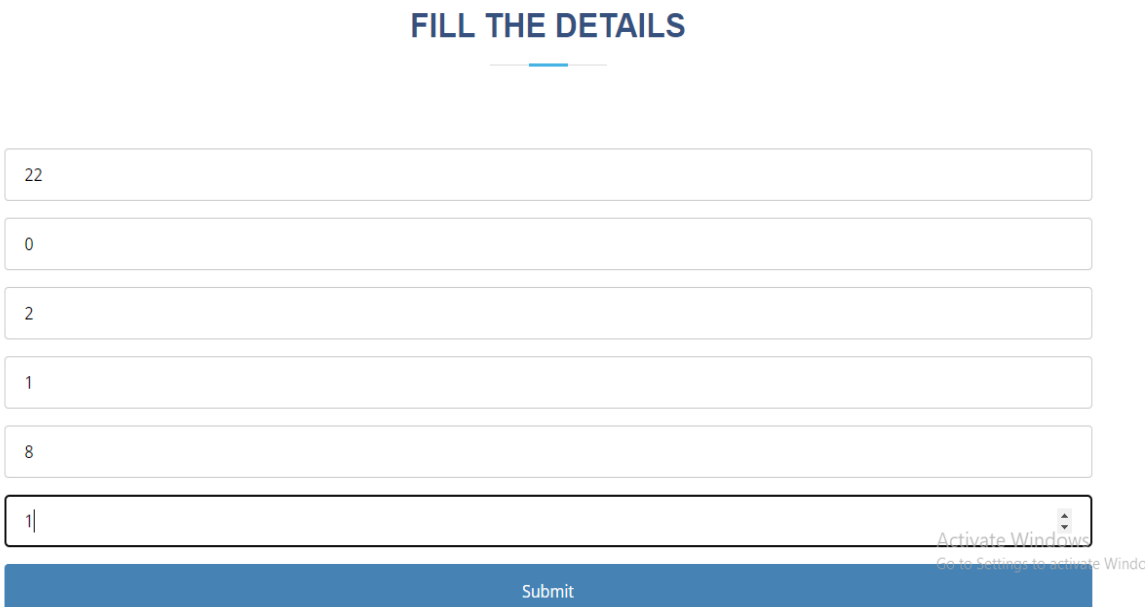
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
* 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)
```

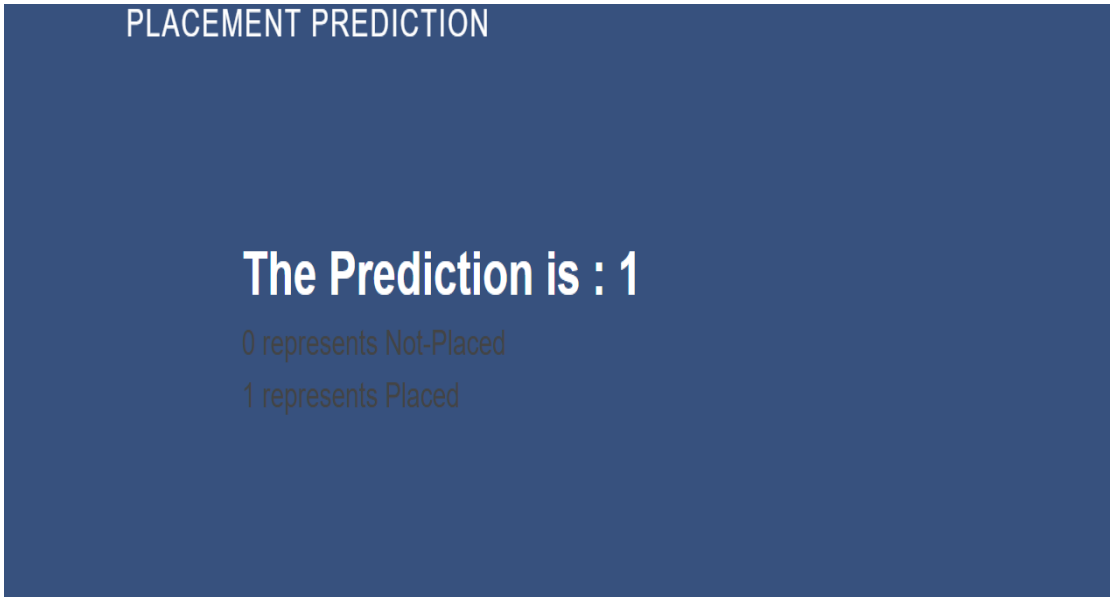
- Copy the link from the prompt and paste it in chrome. A web page opens up as described below
- Let’s see how our page looks like :



- We need to  enter the values into the fields provided to get our prediction
- Let's look how our html file looks like after entering values:




- Now when you click on submit button to see the prediction
- Let’s look how our prediction looks like:



## Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

1) **Activity 1**:  Record explanation Video for project end to end solution
2) **Activity 2**: Project Documentation-Step by step project development procedure