IDENTIFYING PATTERNS AND TRENDS IN CAMPUS PLACEMENT DATA USING MACHINE LEARNING

1.INTRODUCTION

1.1 OVERVIEW:

In Placement Prediction system predicts the probability of an undergraduate students getting placed in a company by applying classification algorithms such as Decision tree and Random forest. The main objective of this model is to predict whether the student he/she gets placed or not in campus recruitment. For this the data consider is the academic history of student like overall percentage, backlogs, credits. The algorithms are applied on the previous years data of the students.

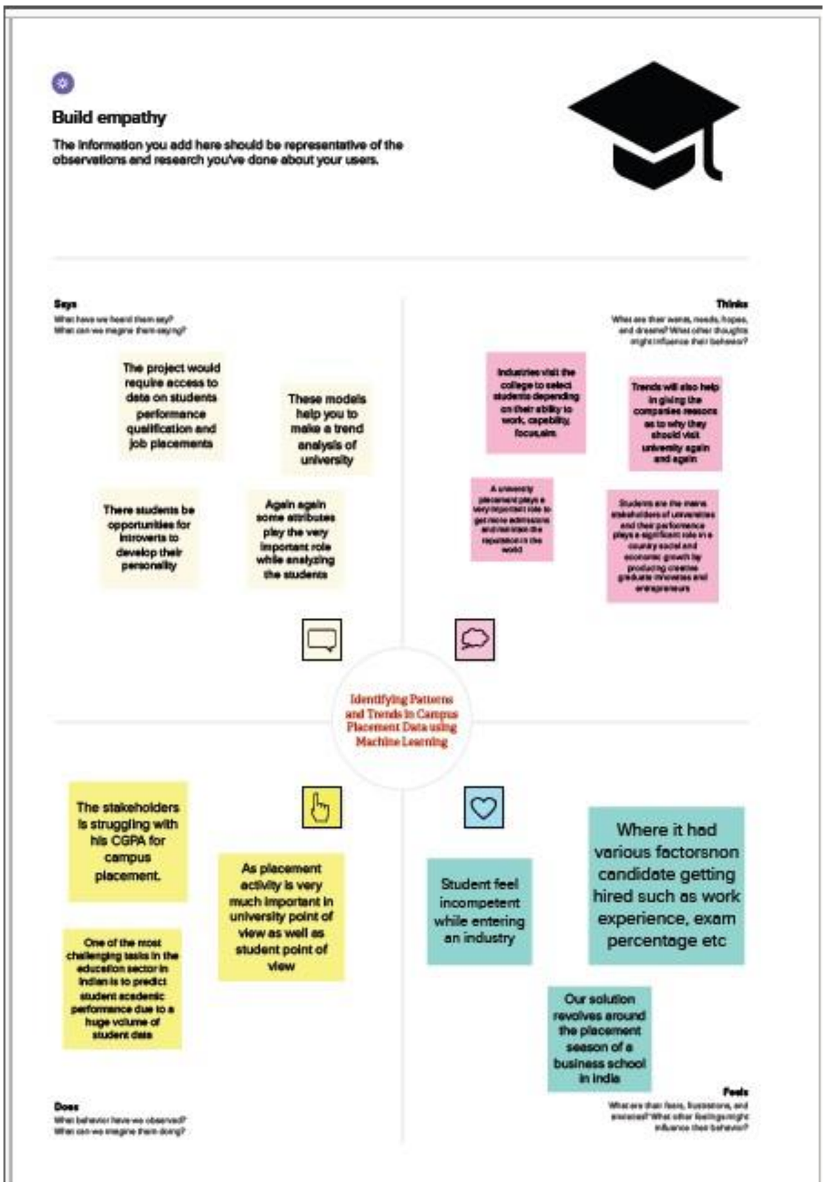
1.2 Purpose:

The objective is to predict the students getting placed for the current year by analyzing the data collected from previous year's students. This model is proposed with an algorithm to predict the same. These models help you to make a trend analysis of university placementsdata, to predict a placement rate for the students of an upcoming year which will help the university to analyze the performance during placements.

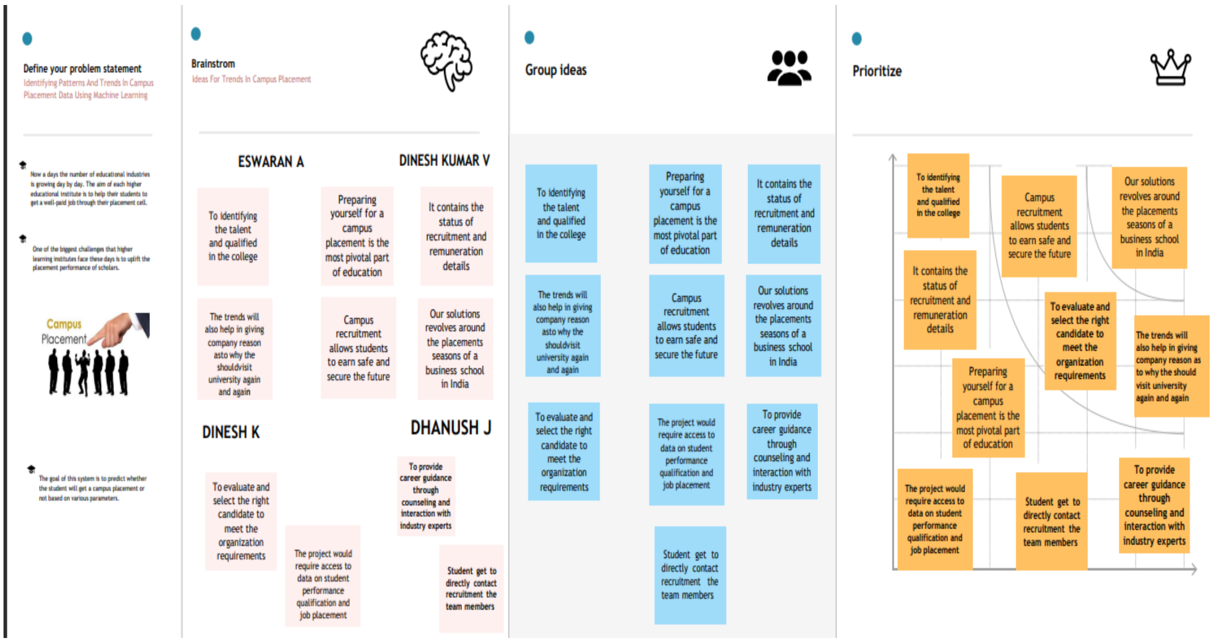
This proposed model is also compared with other traditional classification algorithms such as Decision tree and Random forest with respect to accuracy, precision and recall. From the results obtained it is found that the proposed algorithm performs significantly better in comparison with the other algorithms mentioned.

2. PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Map:



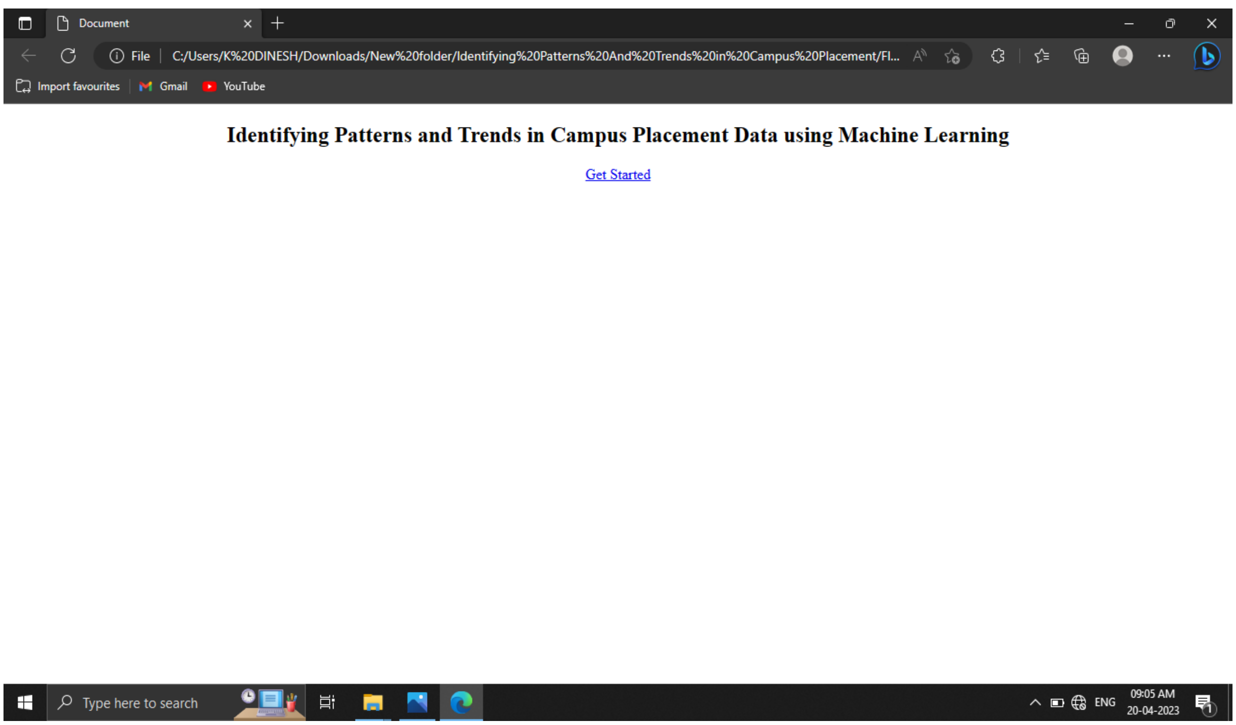
2.2 Ideation & Brainstorming Map:



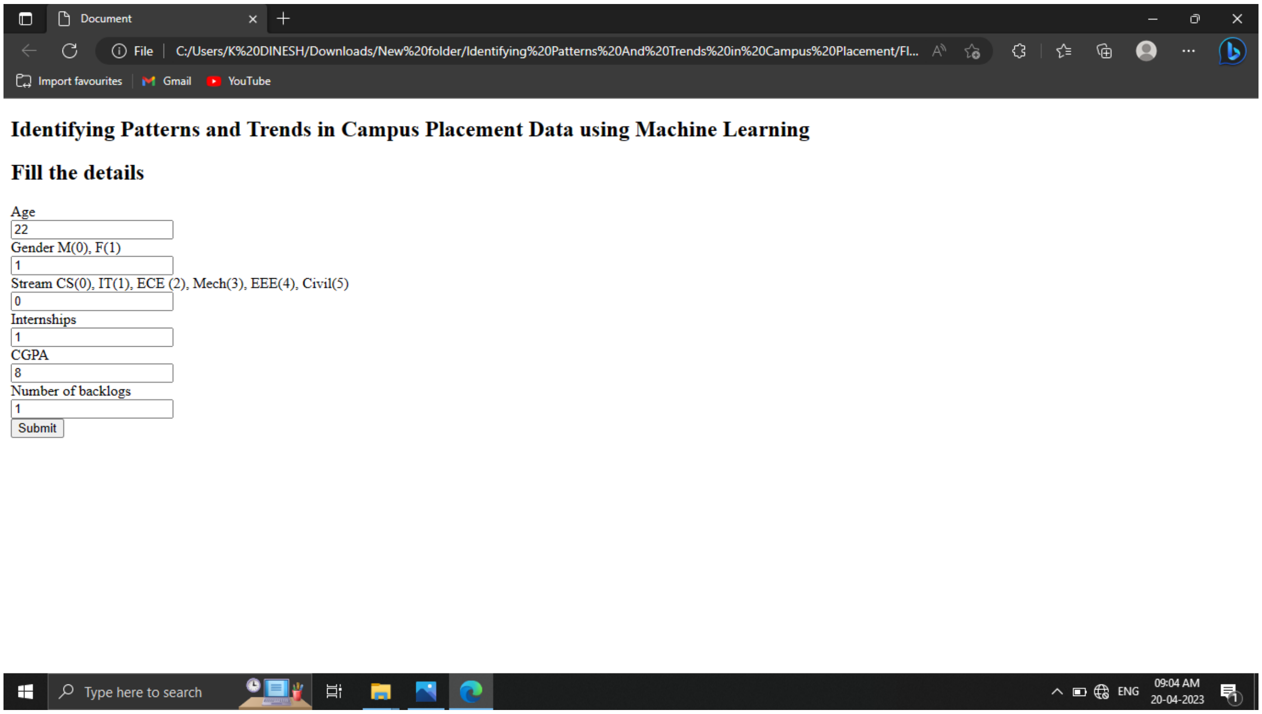
3. RESULT

The algorithms of machine learning we have discussed are can used to find the trend of placement, which will be helpful for university to get more admission in future.

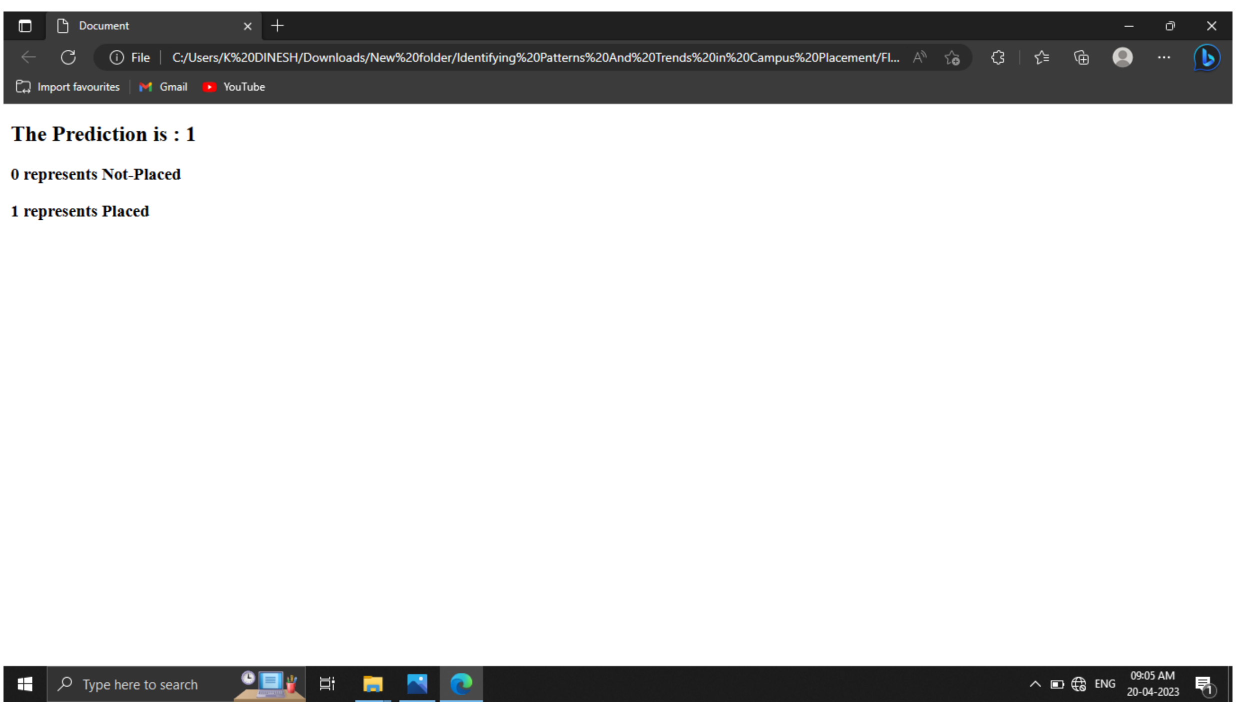
3.1 Home Page/index1:



3.2 Fill the Details/index:



3.3 Predicted page (Final output)/second page:



4. TRAILHEAD PROFILE LINK

Team Lead- https://trailblazer.me/id/eswaran77

Team Member 1- https://trailblazer.me/id/dineshkumarv18

Team Member 2- https://trailblazer.me/id/dineshkd07

Team Member 3- https://trailblazer.me/id/ddhanush23

5. ADVANTAGES AND DISADVANTAGES

Advantages:

- ✓ By analyzing campus placement data, recruiters can identify which universities or program produces the most successful job candidates. This can help recruiters to focus their efforts on these institutions and programs, and to tailor their recruitment strategies accordingly.

- ✓ By examining the skills and qualifications of successful job candidates, employers can identify areas where their current workforce may be lacking. This can help employers to develop targeted training and development programs to address these skill gaps.

- ✓ Campus placement data can also be used to monitor diversity and inclusion efforts.
- ✓ Examining campus placement data over time can provide insights into industry trends and changes in the job market.
- ✓ By analyzing the success rates of graduates from different educational programs, employers can evaluate the effectiveness of those programs in preparing students for the job market.

Disadvantages:

- ✓ Campus placement may only represent a small sample of the overall job market.
- ✓ Campus placement data may be influenced by various biases, such as the preferences of recruiters or the characteristics of the universities or program being studied.
- ✓ Campus placement data may be incomplete or inaccurate due to factors such as incomplete reporting or data entry errors.
- ✓ Campus placement data may not provide enough context to fully understand the factors that contribute to job placement success.

- ✓ The collection and analysis of campus placement data may raise ethical concerns related to privacy and confidentially. Organizations must ensure that they are collecting and analyzing the data in a responsible and ethical manner.

6. APPLICATIONS

- ✓ Campus placement data can help organizations to develop more effective recruitment strategies.

- ✓ BY identifying which universities, programs, or majors produce the most successful job candidates, organizations can focus their recruitment efforts on these areas.

- ✓ Campus placement data can be used to identify skills gaps in the workforce.

- ✓ Campus placement data can be used to monitor diversity and inclusion efforts.

- ✓ Campus placement data can provide insights into industry trends and changes in the job market.

- ✓ Campus placement data can be used to identify potential future leaders within the organizations.

7. CONCLUSION

The campus placement activity is incredibly a lot of vital as institution point of view as well as student point of view. In this regard to improve the student's performance, a work has been analyzed and predicted using the classification algorithms Decision Tree and the Random forest algorithm to validate the approaches. The algorithms are applied on the data set and attributes used to build the model. The accuracy obtained after analysis for Decision tree is 84% and for the Random Forest is 86%. Hence, from the above said analysis and prediction it's better if the Random Forest algorithm is used to predict the placement results.

8. FUTURE SCOPE

Moreover from the study, the researcher concludes that most of the institutions campus placement is not taken seriously. The awareness of such campus hiring should be taught to the students. The students also should consider the importance of campus placement as a part of academics and should have the ability to create academic balance. The significance of Off-campus drives should be taught to the students to crack the campus efficiently. The students should also develop the habit of reading books and journals to prepare themselves for campus placement. The organization should find ways to develop their process of recruitment which in turn helps in an increase in return on investment of campus hiring.

9. APPENDIX

Source Code:

Milestone 1:

Data Collection & Preparation

Importing the libraries:

```
[1] #TASK 2 [DATA COLLECTION & PREPARATION]
```

```
#importing the libraries

import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn import svm
from sklearn.svm import SVC
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
```

Read the Dataset:

```
#Read the Dataset

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

Handling missing values:

```
#DATA PREPARATION
# 1) Handling Missing data

df.info()
```

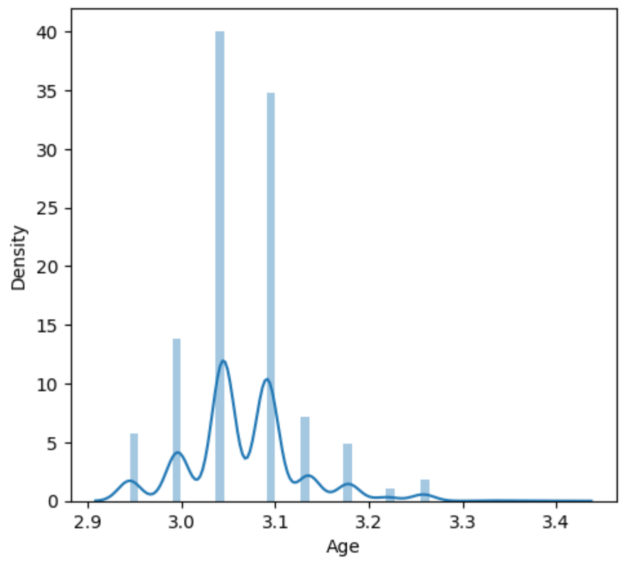
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2966 entries, 0 to 2965
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Age                2966 non-null   int64
 1   Gender             2966 non-null   object
 2   Stream             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
```

Handling outliers:

```
def transformationplot(feature):
  plt.figure(figsize=(12,5))
  plt.subplot(1,2,1)
  sns.distplot(feature)
transformationplot(np.log(df['Age']))
```



Handling Categorical Values:

```
# Handling Categorical Values

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


df = df.replace(['Computer Science','Information Technology','Electronics And Communication','Mechanical','Electrical','C:
```

```
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

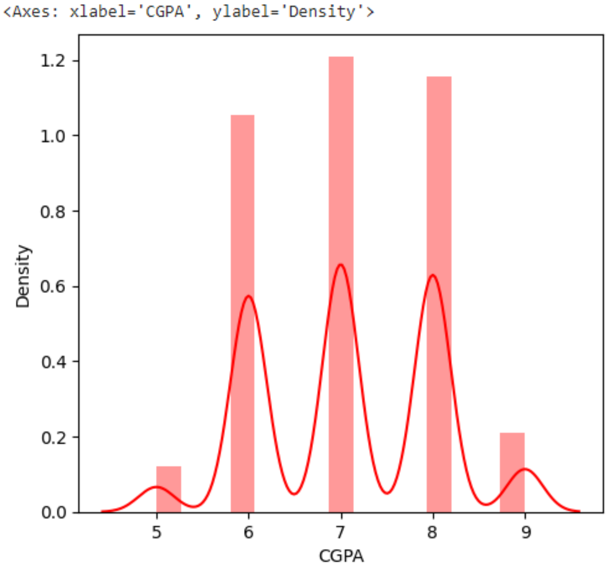
Milestone 2:

Exploratory Data Analysis

Uni-variate analysis:

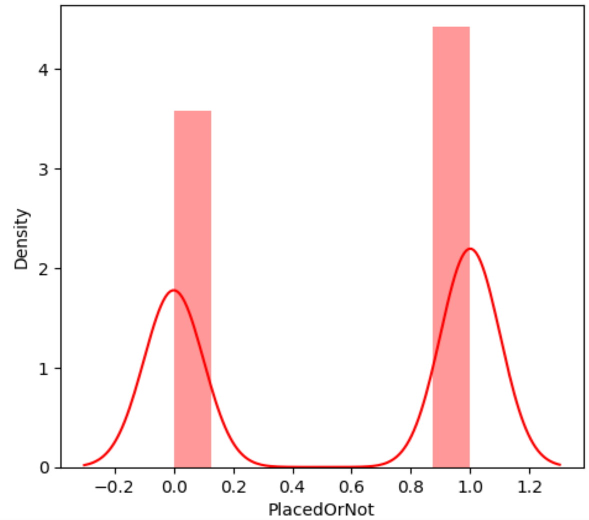
```
#univariate analysis
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['CGPA'],color='r')
```

<Axes: xlabel='CGPA', ylabel='Density'>



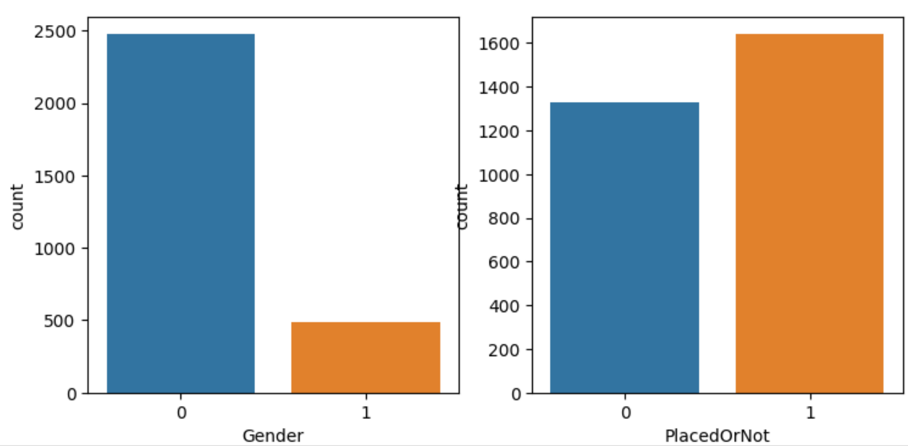
```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['PlacedOrNot'],color='r')
```

```
<Axes: xlabel='PlacedOrNot', ylabel='Density'>
```



Bi-variate analysis:

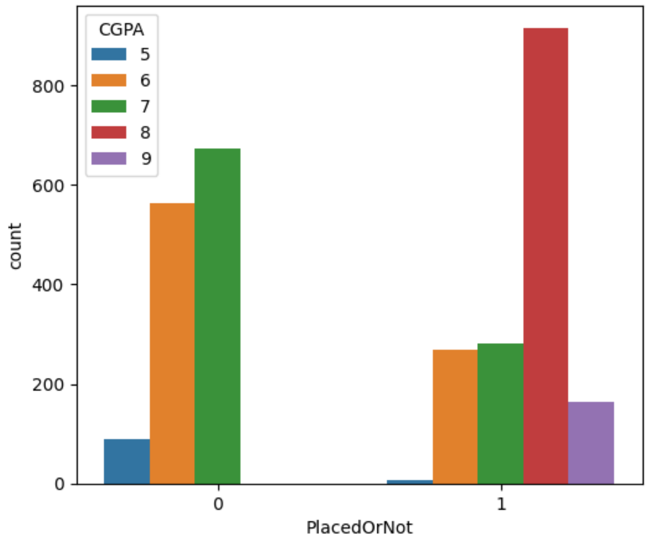
```
#Bivariate analysis
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(data=df,x=df['Gender'])
plt.subplot(1,4,2)
sns.countplot(data=df,x=df['PlacedOrNot'])
plt.show()
```



Multi-variate analysis:

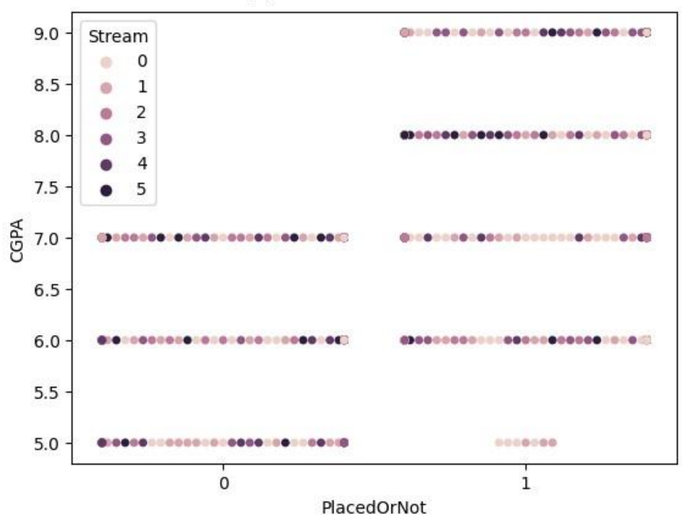
```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(data=df,x=df["PlacedOrNot"],hue=df['CGPA'])
```

```
<Axes: xlabel='PlacedOrNot', ylabel='count'>
```




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

```
<Axes: xlabel='PlacedOrNot', ylabel='CGPA'>
```



Scaling the data:

```
[16] #Scaling the data

     sc=StandardScaler()
     x_bal=sc.fit_transform(df)
     x_bal=pd.DataFrame(x_bal)
```

Splitting the data into train and test:

```
[17] #Splitting the data into train and test

     x=x_bal
     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 3:

Model Building

SVM model:

```
#TASK 4 [MODEL BUILDING]
#Training the model in multiple algorithms
```

```
[15] #SVM Model
     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
```

KNN model:

```
[23] #KNN Model
best_k = {"Regular":0}
best_score = {"Regular":0}
for k in range(3,50,2):
  knn_temp = KNeighborsClassifier(n_neighbors=k)
  knn_temp.fit(X_train, Y_train)
  knn_temp_pred = knn_temp.predict(X_test)
  score = metrics.accuracy_score(Y_test, knn_temp_pred) * 100
  if score >= best_score["Regular"] and score < 100:
    best_score["Regular"] = score
    best_k["Regular"] = k
  else:
    best_score["Regular"] = score
    best_k["Regular"] = k
    break
```

```
print("---Results---\nK: {}\nScore: {}".format(best_k, best_score))

knn = KNeighborsClassifier(n_neighbors=best_k["Regular"])

knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
testd = accuracy_score(knn_pred,Y_test)
print(testd)
```

```
---Results---
K: {'Regular': 7}
Score: {'Regular': 87.20538720538721}
0.8720538720538721
```

Artificial neural network model:

```
#Artificial neural network model

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

```
[23] classifier = Sequential()

classifier.add(keras.layers.Dense(6,activation = 'relu',input_dim=6))
classifier.add(keras.layers.Dropout(0.50))

classifier.add(keras.layers.Dense(6,activation = 'relu'))
classifier.add(keras.layers.Dropout(0.50))

classifier.add(keras.layers.Dense(1,activation = 'sigmoid'))
```

```
loss_1 = tf.keras.losses.BinaryCrossentropy()

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

```
classifier.fit(X_train, Y_train, batch_size=20, epochs=100)

Epoch 1/100
119/119 [==============================] - 1s 2ms/step - loss: 1.2003 - accuracy: 0.4831
Epoch 2/100
119/119 [==============================] - 0s 2ms/step - loss: 0.7836 - accuracy: 0.4798
Epoch 3/100
119/119 [==============================] - 0s 2ms/step - loss: 0.7189 - accuracy: 0.4802
Epoch 4/100
119/119 [==============================] - 0s 2ms/step - loss: 0.7029 - accuracy: 0.4751
Epoch 5/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6952 - accuracy: 0.4857
Epoch 6/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6925 - accuracy: 0.5451
Epoch 7/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5405
Epoch 8/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6914 - accuracy: 0.5527
Epoch 9/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6905 - accuracy: 0.5527
Epoch 10/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6884 - accuracy: 0.5527
Epoch 11/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6903 - accuracy: 0.5527
Epoch 12/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6910 - accuracy: 0.5527
Epoch 13/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
Epoch 14/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6882 - accuracy: 0.5527
Epoch 15/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6887 - accuracy: 0.5527
Epoch 16/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6875 - accuracy: 0.5527
Epoch 17/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6883 - accuracy: 0.5527
Epoch 18/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6886 - accuracy: 0.5527
Epoch 19/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 20/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6870 - accuracy: 0.5527
```

```
Epoch 21/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6882 - accuracy: 0.5527
Epoch 22/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6881 - accuracy: 0.5527
Epoch 23/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6881 - accuracy: 0.5527
Epoch 24/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6878 - accuracy: 0.5527
Epoch 25/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6880 - accuracy: 0.5527
Epoch 26/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6875 - accuracy: 0.5527
Epoch 27/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 28/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 29/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6878 - accuracy: 0.5527
Epoch 30/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6878 - accuracy: 0.5527
```

```
Epoch 31/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 32/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 33/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6878 - accuracy: 0.5527
Epoch 34/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 35/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 36/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 37/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 38/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 39/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 40/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6875 - accuracy: 0.5527
```

```
Epoch 41/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 42/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 43/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 44/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6875 - accuracy: 0.5527
Epoch 45/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6874 - accuracy: 0.5527
Epoch 46/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
Epoch 47/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6874 - accuracy: 0.5527
Epoch 48/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 49/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 50/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
```

```
Epoch 51/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 52/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6879 - accuracy: 0.5527
Epoch 53/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6874 - accuracy: 0.5527
Epoch 54/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
Epoch 55/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6876 - accuracy: 0.5527
Epoch 56/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6872 - accuracy: 0.5527
Epoch 57/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 58/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
Epoch 59/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6870 - accuracy: 0.5527
Epoch 60/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6879 - accuracy: 0.5527
Epoch 61/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6875 - accuracy: 0.5527
Epoch 62/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6874 - accuracy: 0.5527
Epoch 63/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6870 - accuracy: 0.5527
Epoch 64/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 65/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6864 - accuracy: 0.5527
Epoch 66/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
Epoch 67/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6866 - accuracy: 0.5527
Epoch 68/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6864 - accuracy: 0.5527
Epoch 69/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6871 - accuracy: 0.5527
Epoch 70/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6863 - accuracy: 0.5527
Epoch 71/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6877 - accuracy: 0.5527
Epoch 72/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6870 - accuracy: 0.5527
Epoch 73/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6859 - accuracy: 0.5527
Epoch 74/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6873 - accuracy: 0.5527
Epoch 75/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6863 - accuracy: 0.5527
```

```
Epoch 76/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6854 - accuracy: 0.5527
Epoch 77/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6862 - accuracy: 0.5527
Epoch 78/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6848 - accuracy: 0.5527
Epoch 79/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6828 - accuracy: 0.5527
Epoch 80/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6842 - accuracy: 0.5527
Epoch 81/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6841 - accuracy: 0.5527
Epoch 82/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6841 - accuracy: 0.5527
Epoch 83/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6841 - accuracy: 0.5527
Epoch 84/100
119/119 [==============================] - 0s 3ms/step - loss: 0.6845 - accuracy: 0.5527
Epoch 85/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6865 - accuracy: 0.5527
Epoch 86/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6845 - accuracy: 0.5527
Epoch 87/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6846 - accuracy: 0.5527
Epoch 88/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6832 - accuracy: 0.5527
Epoch 89/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6867 - accuracy: 0.5527
Epoch 90/100
119/119 [==============================] - 0s 3ms/step - loss: 0.6826 - accuracy: 0.5527
Epoch 91/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6832 - accuracy: 0.5527
Epoch 92/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6835 - accuracy: 0.5527
Epoch 93/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6819 - accuracy: 0.5527
Epoch 94/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6813 - accuracy: 0.5527
Epoch 95/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6828 - accuracy: 0.5527
Epoch 96/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6804 - accuracy: 0.5527
Epoch 97/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6826 - accuracy: 0.5527
Epoch 98/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6798 - accuracy: 0.5527
Epoch 99/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6809 - accuracy: 0.5527
Epoch 100/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6832 - accuracy: 0.5527
<keras.callbacks.History at 0x7f655d4cf490>
```

Milestone 4:
Model Deployment

Save the best model:

```python
[26] #TASK 5 [MODEL DEPLOYMENT]
```

```python
#Save the best model

import pickle

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

Integrate with Web Framework

index.html app.py index1.html secondpage.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body bgcolor="Aqua">
    <section id="hero">
        <div>
            <div>
                <div>
                    <h1>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h1>
                </div>
            </div>
        </div>
    </section>
    <section>
        <div>
            <div>
                <h2>Fill the details</h2>
            </div>
```

html › body

index.html × app.py index1.html secondpage.html

```html
                <div>
                    <div>
                        <form action="{{url_for('y_predict')}}" method="POST">
                            <label>Age</label><br>
                            <input type="number" id="sen1" name="sen1"><br>
                            <label>Gender M(0), F(1)</label><br>
                            <input type="number" id="sen2" name="sen2"><br>
                            <label>Stream CS(0), IT(1), ECE (2), Mech(3), EEE(4), Civil(5)</label><br>
                            <input type="number" id="sen3" name="sen3"><br>
                                <label>Internships</label><br>
                            <input type=" number" id="sen4" name="sen4"><br>
                            <label>CGPA</label><br>
                            <input type="number" id="sen5" name="sen5"><br>
                            <label>Number of backlogs</label><br>
                            <input type="number" id="sen6" name="sen6"><br>
                            <input type="submit" value="Submit">
                        </form>
                    </div>
                </div>
            </div>
        </section><!-- End About Us Section -->
</body>

</html>
```

html › body

index.html app.py index1.html × secondpage.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body bgcolor="Gray">
    <section id="hero">
        <center>
            <div>
                <div>
                    <div>
                        <h1>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h1>
                        <a href="/guest">Get Started</a>
                    </div>
                </div>
            </div>
        </center>
    </section>
</body>
</html>
```

html

onProject3 › Placement › templates › index1.html 25:1 CRLF UTF-8 4 spaces Python 3.10 (pytho index.html app.py index1.html secondpage.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body bgcolor="slateBlue">
<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 -->
</body>
</html>
```

html › body

index.html app.py index1.html secondpage.html

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

@app.route("/")
def hello():
    return render_template("index1.html")

@app.route('/guest')
def Guest():
    return render_template("index.html")

@app.route('/y_predict', methods = ["POST"])
def y_predict():
    x_test = [[int(yo) for yo in request.form.values()]]
    prediction = model.predict(x_test)
    prediction = prediction[0]
    return render_template("secondpage.html",y=prediction)

if __name__ == "__main__":
    app.run(debug=True)
```

y_predict()