Identifying Patterns and Trends in Campus Placement Data using Machine Learning

Project report submitted to

" Madurai kamaraj university"

In partial fulfillment of the requiremts for

The award of the degree of

Bachelor of Science

in

Computer Science

Submitted

By

Team ID :NM2023TMID30946

TEAM LEAD : K.Lavanya

TEAM MEMBER-01 :N.Nandhini devi

TEAM MEMBER-02 :D.Nivetha

TEAM MEMBER-03 : S.Ponnila

Under the Guidance of

Dr.B.UMADEVI M.Sc.,M.Phil.,Ph.D.,

(Assistant Professor,GAC,Melur)

GOVERNMENT ARTS COLLEGE

P.G. DEPARTMENT OF COMPUTER SCIENCE

MELUR-625 106

INDEX

1.INTRODUCTION

Campus placement or campus recruiting is a program conducted within universities or other educational institutions to provide jobs to students nearing completion of their studies. In this type of program, the educational institutions partner with corporations who wish to recruit from the student population.

1.1: 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.

1.2: Overview:

- Identifying patterns and trends in campus placement data using machine learning involves applying various algorithms and techniques to analyze data related to job placements of students from a particular educational institution. The goal is to identify patterns and trends in the data that can help improve future job placement outcomes for students.

- This process typically involves several steps, including data collection, data cleaning, data analysis, and model building. Data collection involves gathering relevant data related to the job placements of students from a particular institution. Data cleaning involves removing any irrelevant or incomplete data and ensuring that the data is in a suitable format for analysis.

- Data analysis involves applying various machine learning algorithms and techniques to the cleaned data to identify patterns and trends. These algorithms and techniques may include clustering, regression, classification, and neural networks, among others. Model building involves developing predictive models based on the analyzed data that can help identify potential job opportunities for students in the future.

2. Collect the Dataset:

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.

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

3. Importing the libraries:

Importing libraries means including pre-written code or functionality into our program, allowing us to use ready-made functions and classes without having to write them from scratch.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

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
```

4. Read the Dataset:

Reading the dataset refers to the process of loading data from an external file or source into memory in a format that can be easily manipulated by a program or script.

```
df = pd.read_csv('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

5. Descriptive State:

Descriptive statistics is a branch of statistics that deals with the summary and presentation of data in a meaningful way. It involves calculating various measures such as mean, median, mode, standard deviation, and range to describe the central tendency, dispersion, and shape of a dataset.

```
df.describe()
```

	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
min	19.000000	0.000000	5.000000	0.000000	0.000000	0.000000
25%	21.000000	0.000000	6.000000	0.000000	0.000000	0.000000
50%	21.000000	1.000000	7.000000	0.000000	0.000000	1.000000
75%	22.000000	1.000000	8.000000	1.000000	0.000000	1.000000
max	30.000000	3.000000	9.000000	1.000000	1.000000	1.000000

6. 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

2.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().any()
```

```
Age                 False
Gender              False
Stream              False
Internships         False
CGPA                False
Hostel              False
HistoryOfBacklogs   False
PlacedOrNot         False
dtype: bool
```

```
df.isnull().sum()
```

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

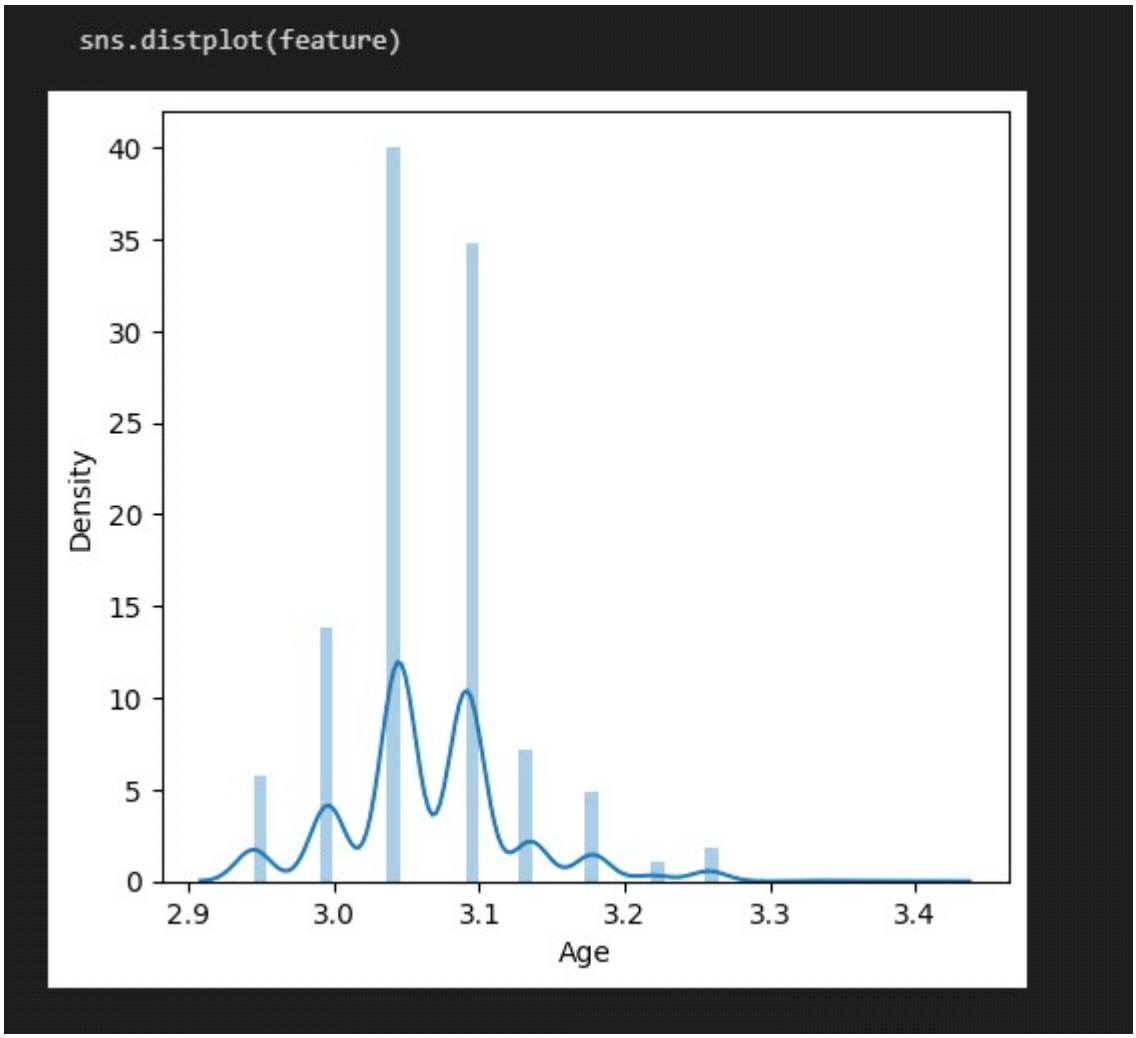
2.1: Handling Outliers:

Handling outliers refers to the process of identifying and dealing with data points that are significantly

different from the rest of the dataset. This can involve removing outliers, transforming them to be more in line with the rest of the data, or treating them as a separate category in analysis.

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

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



2.2: 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['Gender'].replace({"Male":1,"Female":0},inplace=True)
```

```
df['Stream'].replace({"Electronics And Communication":1,"Computer Science":2,"Information Technology":3,"Mechanical":4,"Electrical":5,"Civil":6},inplace=True)
```

```
df
```

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

2966 rows × 7 columns

3. Exploratory Data Analysis:

3.1: Visual analysis:

l analysis is the process of using visual representations, such as arts, 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.

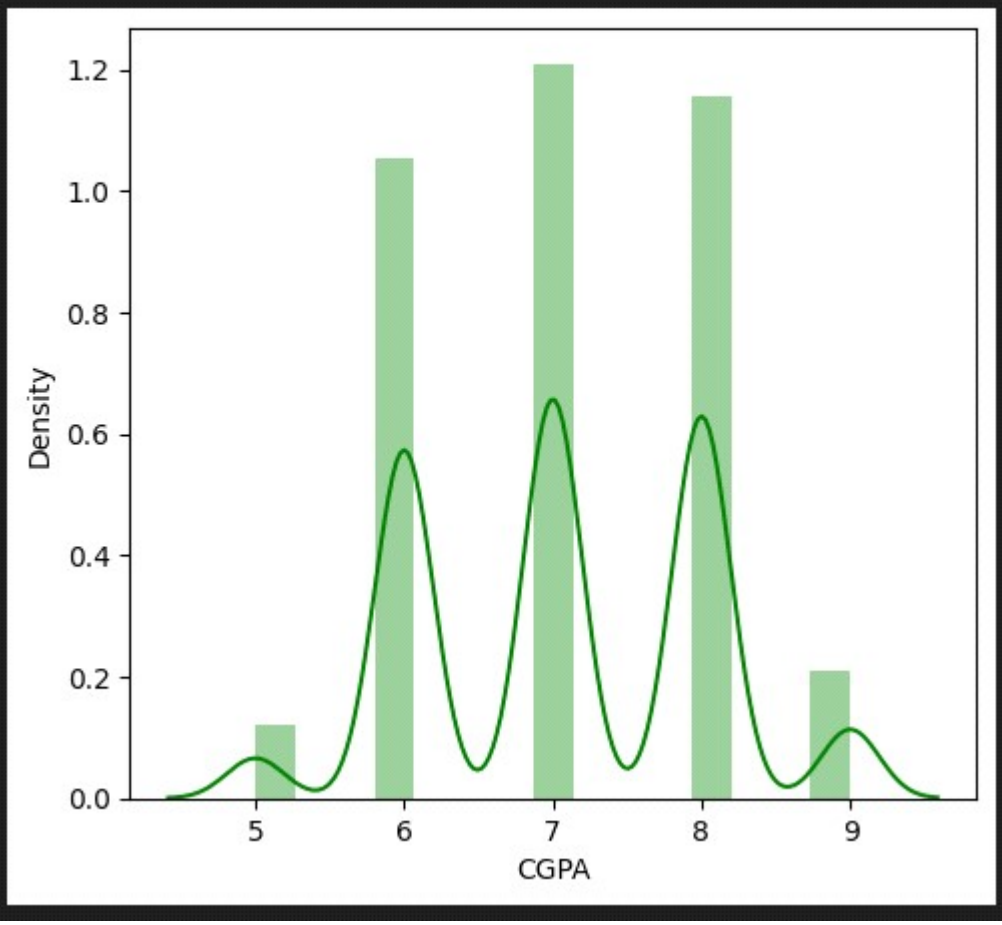
3.2: Univariate analysis:

In simple words, univariate analysis is understanding the data ith 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='g')
```

```
sns.distplot(df['CGPA'], color='g')
```

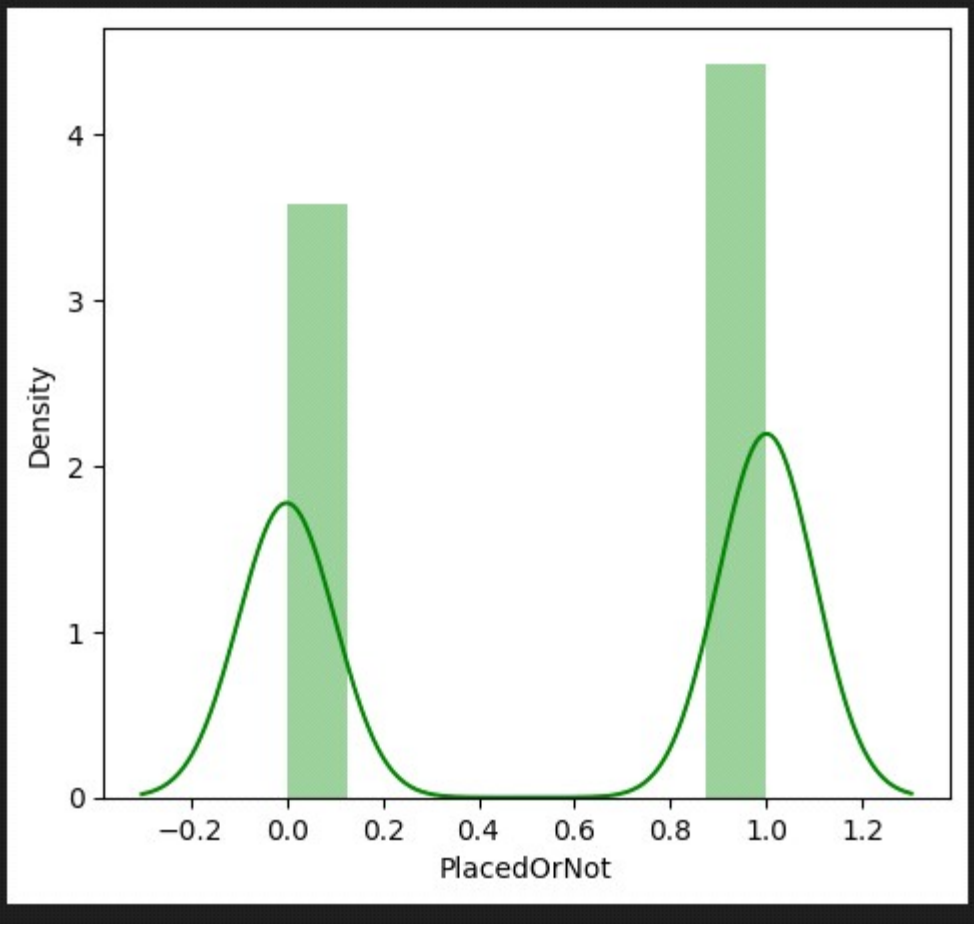
```
<Axes: xlabel='CGPA', ylabel='Density'>
```




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

```
sns.distplot(df['PlacedOrNot'],color='g')
```

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

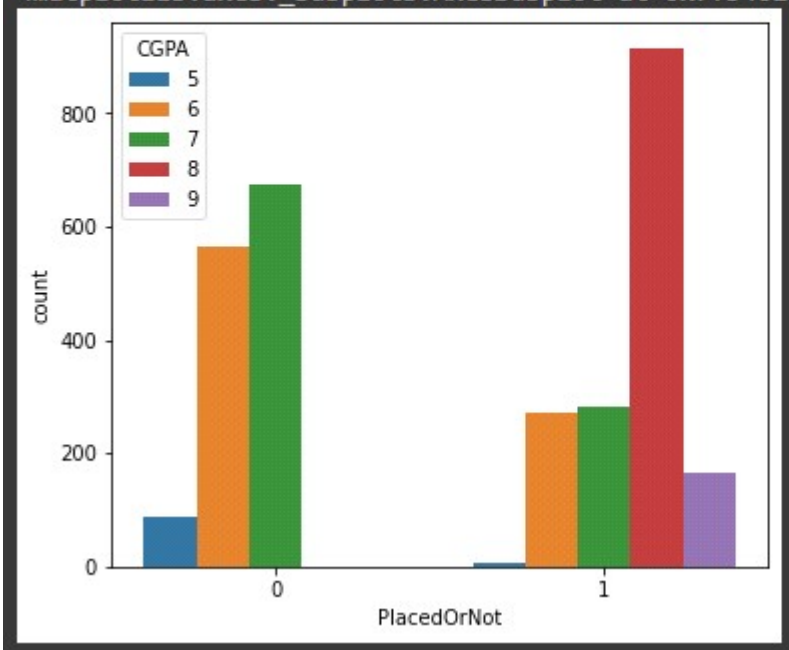


3.3 : Multivariate analysis:

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used swarmplot from the seaborn package.

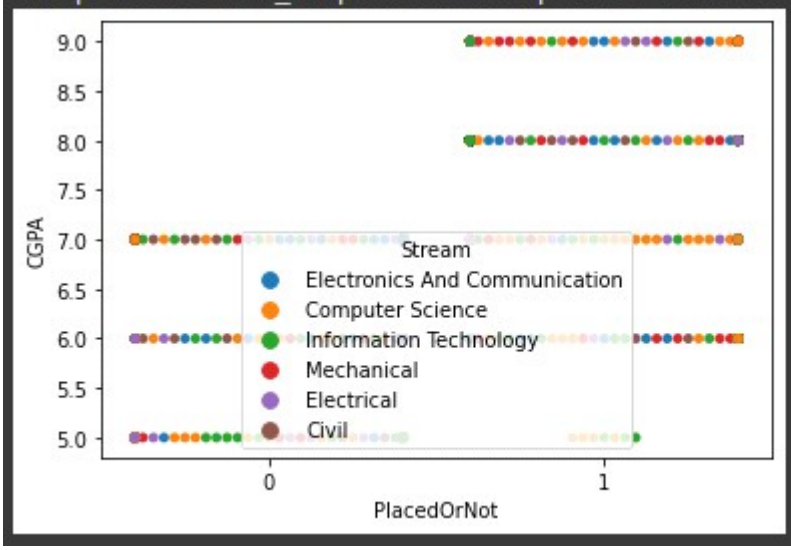
```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(df["PlacedOrNot"],hue=df['CGPA'])
```

```
/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>
```



4. 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 = df.drop('PlacedOrNot', axis=1)
y = df['PlacedOrNot']
```

```
x
```

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

2966 rows × 6 columns

```
y
```

```
0       1
1       1
2       1
3       1
4       1
       ..
2961    0
2962    0
2963    0
2964    0
2965    1
Name: PlacedOrNot, Length: 2966, dtype: int64
```

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=15)
```

5. 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.

5.1: 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.

5.2: : 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
```

5.3: 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}
```

5.4: Logistic Regression Model:

Logistic regression is a statistical method used to analyze and model the relationship between a categorical dependent variable (such as "yes" or "no") and one or more independent variables (also known as predictor variables). It is a type of regression analysis that uses a logistic function to model the probability of the dependent variable. Logistic regression is commonly used in fields such as medicine, finance, marketing, and social sciences to predict the likelihood of an event occurring based on a set of predictor variables.

```
log_r = LogisticRegression()
```

```
log_r.fit(xtrain,ytrain)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
ypred = log_r.predict(xtest)
```

```
ypred
```

```
array([1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],
      dtype=int64)
```

```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.78      0.78       271
           1       0.82      0.81      0.82       323

    accuracy                           0.80       594
   macro avg       0.80      0.80      0.80       594
weighted avg       0.80      0.80      0.80       594
```

```
confusion_matrix(ytest,ypred)
```

```
array([[212,  59],
       [ 60, 263]], dtype=int64)
```

```
log_r.predict([[22,1,2,1,8,1]])
```

```
array([1], dtype=int64)
```

6. Model Deployment:

6.1: 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'))
```

6.2: 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

6.3: Building Html Pages:

For this project create one HTML file namely

1)index.html

2)predict.html

3)output.html

and save it in templates folder

1) index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
</head>
<body>
 <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">
                   <h2>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h2>
                    <form action="\">
                        <input type="submit" value="Home">
                    </form>
                    <form action="/predict">
                        <input type="submit" value="Predict">
                    </form>
            </div>
        </div>
    </div>
</section><!-- End Hero -->
</body>
</html>
```

2) predict.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction Page</title>

</head>
<body>
    <form action="/Register" method="POST" autocomplete="off">
        <div class="col-md-offset-3 col-md-4">
            <h3 class="page-header text-primary">Registration</h3>
            <div class="form-group">
                <label>Age:</label>
                <input type="number" name="age" placeholder="Enter Your Age" class="form-control" required>
            </div><br>
            <div class="form-group">
                <label>Gender:</label>
                <input type="number" name="gender" placeholder="Male(1), Female(0)" class="form-control" required>
            </div><br>
            <div class="form-group">
                <label>Stream:</label>
                <input type="number" name="stream" placeholder="ECE(1), CS(2), IT(3), Mech(4), Elec(5), Civil(6)" class="form-control" required>
            </div><br>
            <div class="form-group">
                <label>Internships:</label>
                <input type="number" name="internships" placeholder="Internships" class="form-control" required>
            </div><br>
```

```
        <div class="form-group">
            <label>Internships:</label>
            <input type="number" name="internships" placeholder="Internships" class="form-control" required>
        </div><br>
        <div class="form-group">
            <label>CGPA:</label>
            <input type="number" name="cgpa" placeholder="CGPA" class="form-control" required>
        </div><br>
        <div class="form-group">
            <label>No_of_Backlog:</label>
            <input type="number" name="no_of_backlog" placeholder="No of Backlog" class="form-control" required>
        </div><br><br>
        <div class="form-group">
            <input type="submit" name="submit" value="Submit" class="btn-success">
            <input type="reset"  value="Clear" class="btn-danger">
        </div>
    </div>
</form>
</body>
</html>
```

3) output.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Output Page</title>
    <link rel="stylesheet" href="index.css">
</head>
<body>
    <h1>The Prediction is: {{y}}</h1>
    <h3>0 represents Not-Placed </h3>
    <h3> 1 represents Placed</h3>
    <form action="\">
        <input type="submit" value="Home">
    </form>
    <form action="/predict">
        <input type="submit" value="Predict">
    </form>
</body>
</html>
```

6.4: Build Python code:

Import the libraries:

Load the saved model. Importing the flask module in the project is andatory. An object of Flask class is our WSGI application. Flask onstructor takes the name of the current module (__name__) as agument.

```python
from flask import Flask, render_template, request
import pickle
import sklearn
import joblib


app = Flask(__name__)

model = pickle.load(open('placement.pkl', 'rb'))


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


1 usage (1 dynamic)
@app.route('/predict')
def predict():
    return render_template("predict.html")
```

```python
@app.route('/Register', methods=["POST", "GET"])
def result():
    if request.method == "POST":
        a = request.form.get('age')
        g = request.form.get('gender')
        s = request.form.get('stream')
        i = request.form.get('internships')
        c = request.form.get('cgpa')
        n = request.form.get('no_of_backlog')
        prediction = [[int(a), int(g), int(s), int(i), int(c), int(n)]]

        print(prediction)

        prediction = model.predict[[int(a), int(g), int(s), int(i), int(c), int(n)]]

        if prediction == 0:
            output = "Not-Placed"
        else:
            output = "Placed"

        return render_template("output.html", y=output)


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

Main Function:

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

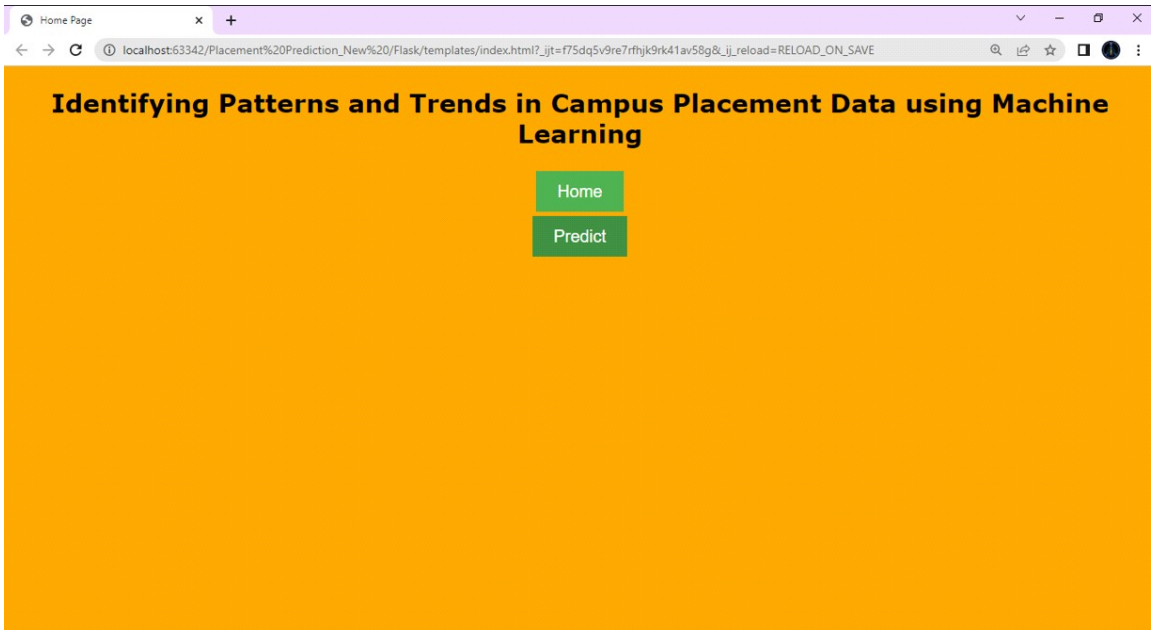
4)Navigate to the localhost where you can view your web page.

❖ 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)
```

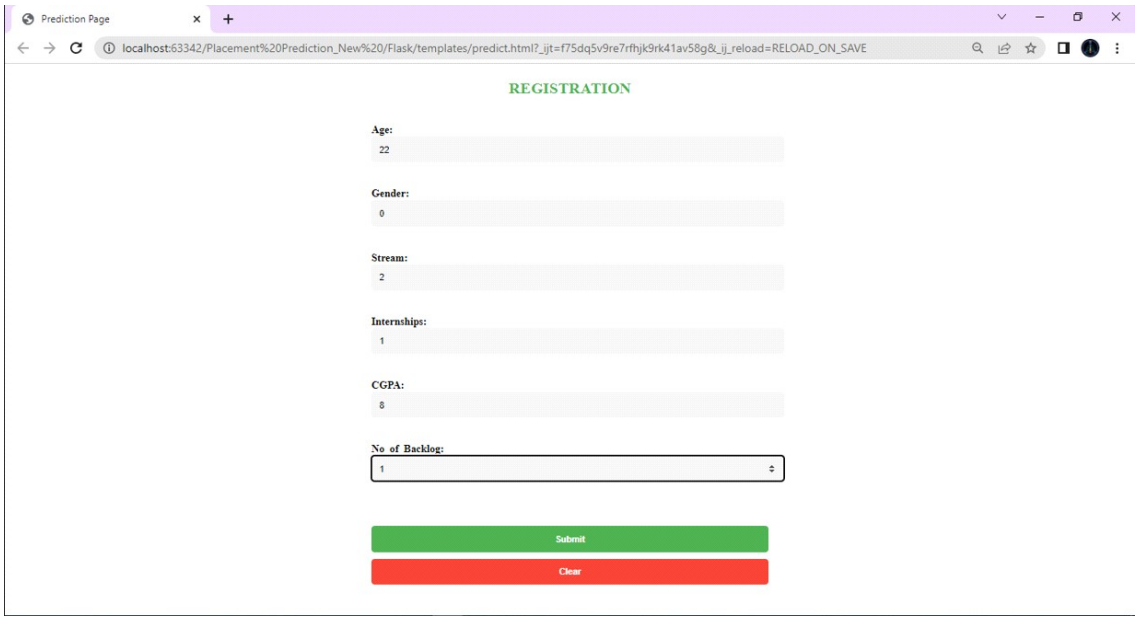
❖ Copy the link from the prompt and paste it in chrome. A web page opens up as described below

❖ Let’s see how our page looks like :



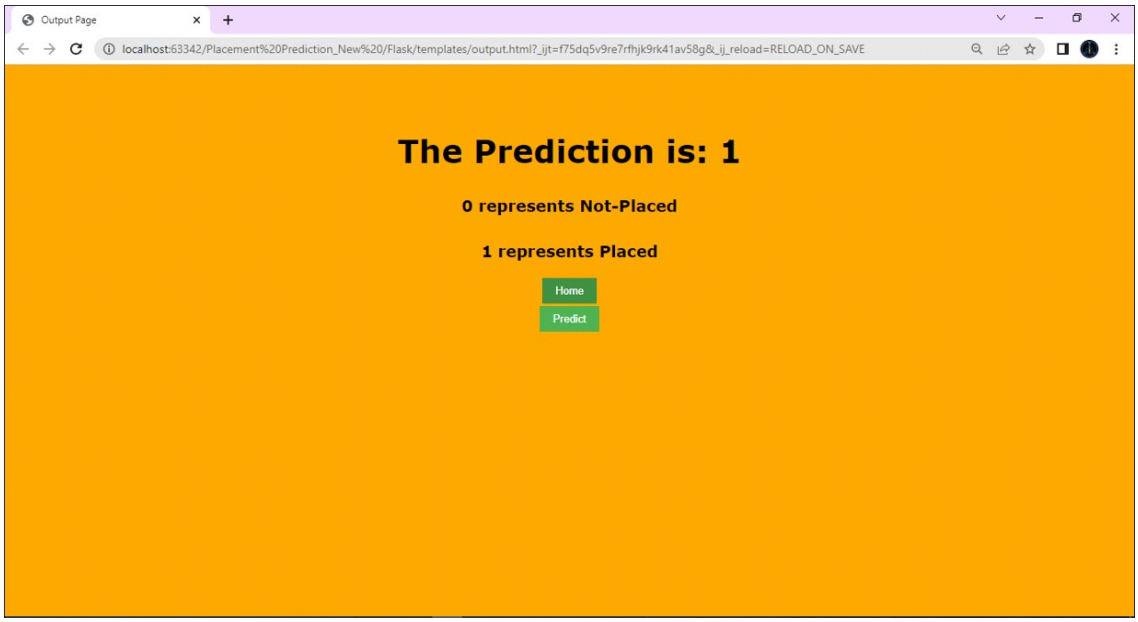
❖ We need to enter the values into the fields provided to get our prediction

❖ 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:



7. Purpose:

The purpose of identifying patterns and trends in campus placement data using machine learning is to gain insights and actionable information that can be used to improve the job placement outcomes for students from educational institutions. By analyzing historical

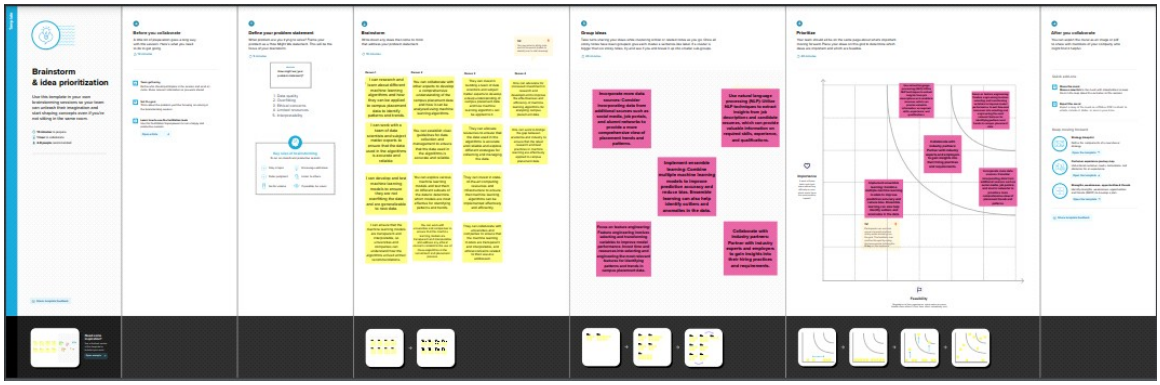
placement data, machine learning models can identify patterns and trends in job placements, including factors that contribute to successful job placements and potential areas for improvement.

Overall, the purpose of identifying patterns and trends in campus placement data using machine learning is to leverage data-driven insights to improve the job placement outcomes of students and better prepare them for success in their chosen career paths.

8. Problem Definition & Design Thinking:

Empathy Map:

Ideation & Brainstorming Map:



9. ADVANTAGES & DISADVANTAGES:

Advantages:

Scalability: Machine learning algorithms can handle large datasets and provide accurate results, which is especially useful in the case of campus placement data that involves a large number of students and companies.

Efficiency: Machine learning algorithms can process and analyze data much faster than humans, which allows for quick identification of patterns and trends.

Accurate predictions: Machine learning algorithms can identify complex patterns and relationships between variables that may not be apparent to humans, leading to more accurate predictions of future placement trends.

Personalized insights: Machine learning algorithms can provide personalized insights and recommendations for individual students based on their unique characteristics and placement history.

Continuous improvement: Machine learning algorithms can learn and improve over time as new data becomes available, allowing for continuous optimization of placement strategies.

Disadvantages:

Data quality: The accuracy and reliability of machine learning algorithms heavily rely on the quality of the data used to train them. Inaccurate or incomplete data can lead to inaccurate results and predictions.

Bias: Machine learning algorithms can be biased based on the data they are trained on, leading to biased predictions and recommendations.

Lack of interpretability: Some machine learning algorithms are considered "black boxes" because they are difficult to interpret and understand. This can be a disadvantage when it comes to explaining the results of the analysis to stakeholders.

Cost: Implementing machine learning algorithms can be costly, especially for smaller organizations or institutions.

Ethical concerns: The use of machine learning algorithms in the hiring and placement process can raise ethical concerns regarding fairness, discrimination, and privacy.

10. APPLICATIONS:

Optimizing career services and counseling programs:

By analyzing historical placement data, educational institutions can identify patterns and trends in

successful job placements and use this information to optimize their career services and counseling programs. This can help students better understand the job market, identify potential job opportunities, and develop the skills and qualifications needed for success in their chosen career paths.

Improving student outcomes:

Machine learning models can be used to predict potential job opportunities for students based on their academic performance, skills, and interests. This information can be used to provide targeted guidance to students on job search strategies and help them make informed decisions about their career paths.

Industry-specific insights:

By analyzing placement data, machine learning models can identify patterns and trends specific to particular industries or job roles. This information can be used by educational institutions to provide targeted guidance to students on the skills and qualifications needed for success in these industries or job roles.

Employer partnerships:

Educational institutions can use machine learning to identify potential employer partnerships based on historical placement data. This can help institutions develop relationships with employers and provide targeted career services to students.

Overall, the applications of identifying patterns and trends in campus placement data using machine learning are vast and can help educational institutions and students make more informed decisions about career paths, optimize career services, and improve job placement outcomes.

11. FUTURE SCOPE:

The future scope of identifying patterns and trends in campus placement data using machine learning is promising. Some potential areas of growth and development in this field include:

Incorporating more data sources:

Machine learning models can be trained on a variety of data sources beyond campus placement data, such as job postings, salary data, and industry trends. Incorporating more data sources can help provide a more comprehensive picture of the job market and improve the accuracy of predictive models.

Personalization:

Machine learning models can be used to develop personalized career guidance and job search strategies for students. This can help students make more informed decisions about their career paths and increase the likelihood of successful job placements.

Integration with other technologies:

Machine learning models can be integrated with other emerging technologies, such as natural language processing and chatbots, to provide personalized career guidance to students in real-time.

Continual improvement:

Machine learning models can continually learn from new data, allowing them to improve over time. This can help educational institutions and students stay up-to-date with the latest job market trends and improve job placement outcomes.

Overall, the future scope of identifying patterns and trends in campus placement data using machine learning is exciting, with many potential opportunities for growth and development. As data collection and analysis technologies continue to advance, machine learning will likely play an increasingly important role in helping students navigate the job market and achieve successful job placements.

12. APPENDIX:

Source Code & Link:

Link with source code -
https://github.com/raji797u/Identifying-Patterns-and-Trends-in-Campus-Placement-Data-using-Machine-Learning_1/blob/main/NmProjectPlacementPrediction_GC.ipynb

Video Demonstration Link - https://youtu.be/8cDahynOBZg

13. Conclusion:

Using machine learning to analyze campus placement data can provide valuable insights and benefits such as improved placement outcomes, enhanced decision-making, and personalized career guidance. However, it is important to ensure accurate and representative data and address potential biases. Overall, this is a promising area for education and workforce development.