Identifying Patterns And Trends In Campus Placement Data Using Machine Learning

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE

TEAM MEMBERS

JENITHA. S	(Reg. No. 10320U18020)
JAYALAKSHMI. M	(Reg. No. 10320U18019)
KOWSALYA. E	(Reg. No. 10320U18024)
MANIMEGALAI.S	(Reg. No. 10320U18019)

Under the Guidance of

Dr. S. MANIKANDAN M.Sc.,M.C.A.,M.Phil,B.Ed.,P.G.Dip.Yoga, Ph.D.,

ASSISTANT PROFESSOR

UG DEPARTMENT OF COMPUTER SCIENCE

UG DEPARTMENT OF COMPUTER SCIENCE

. GOVT.ARTS COLLEGE, CHIDAMBARAM – 608 102

(Affiliated to Thiruvalluvar University, Annamalai Nagar)

Project Report

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 Overview

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.

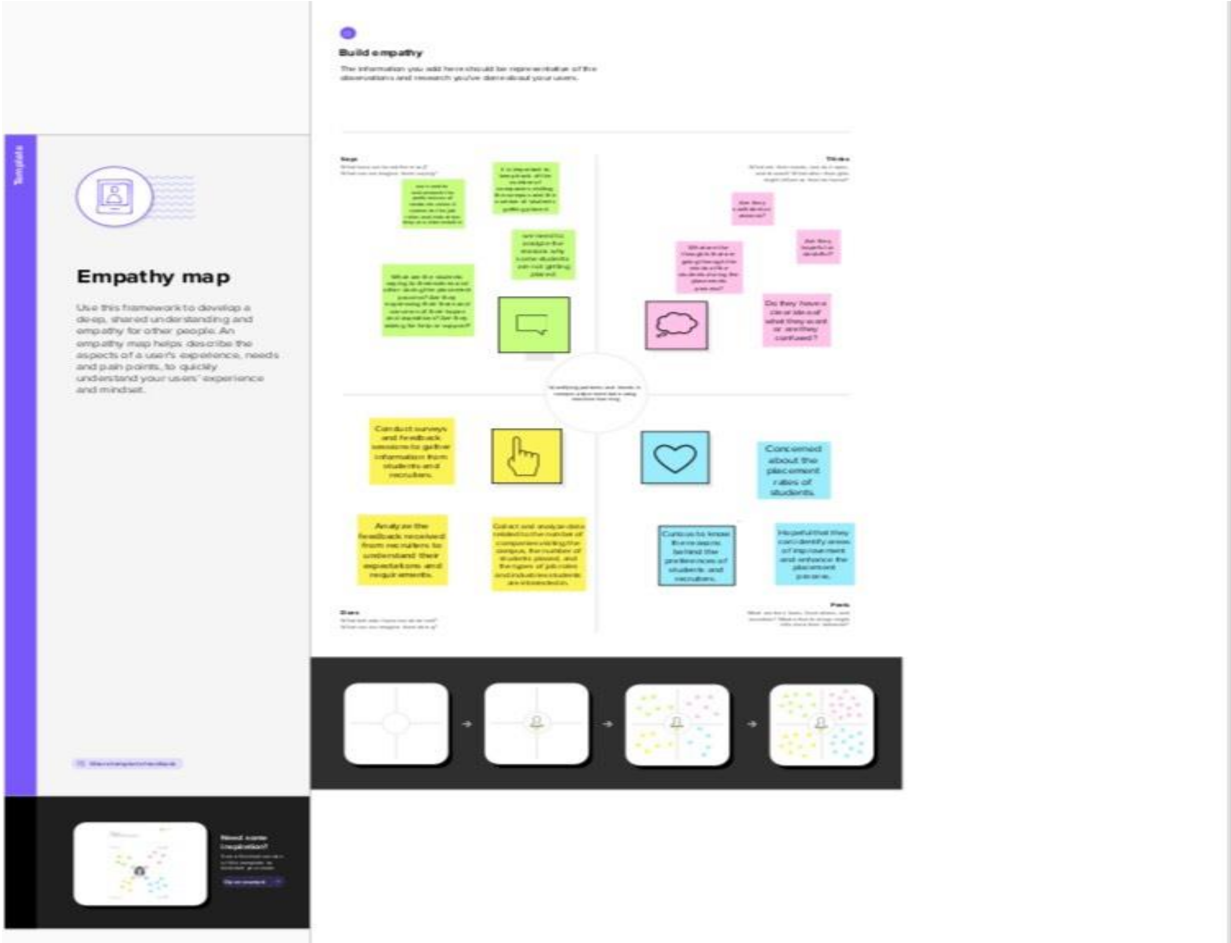
We will be using algorithms such as KNN, SVM and ANN. We will train and test the data with these algorithms. From this the best model is selected and saved in .pkl format. We will be doing flask integration and IBM deployment.

1.2 Purpose:

- Easy to Predict the accurate placement.
- Speed and Efficient Predition.
- More than complexity.

2. Problem Definition & Design Thinking

2.1 Empathy Map



Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

The information you add here should be representative of the observations and research you've done about your users.

we need to analyze the reasons why some students are not getting placed.

Do they have a clear idea of what they want or are they confused?

Conduct surveys and feedback sessions to gather information from students and recruiters.

Analyze the feedback received from recruiters to understand their expectations and requirements.

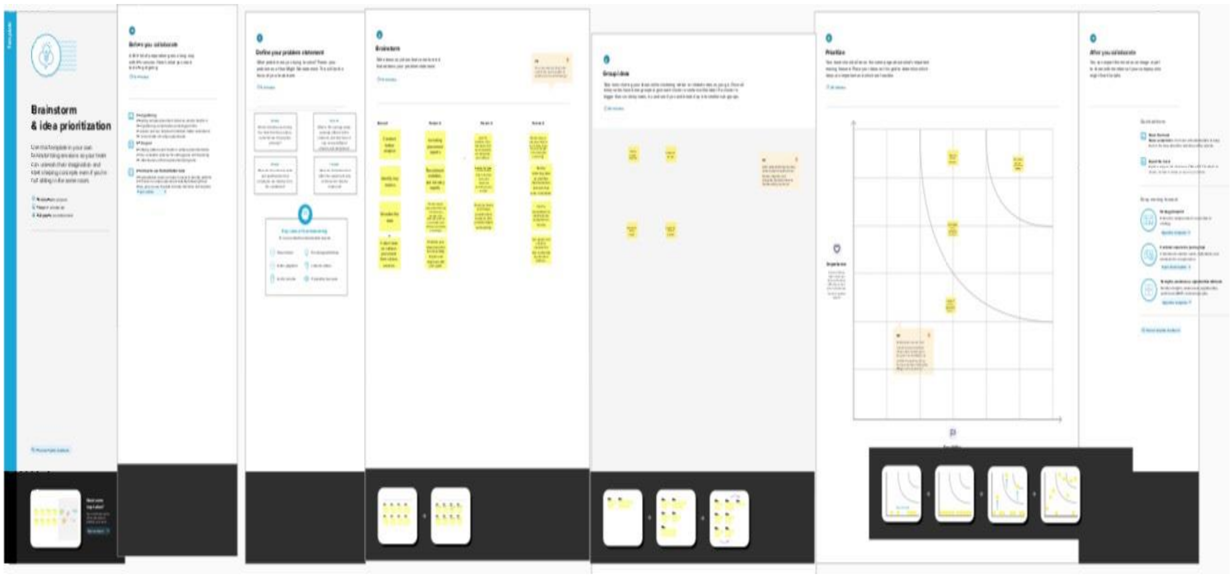
Collect and analyze data related to the number of companies visiting the campus, the number of students placed, and the types of job roles and industries students are interested in.

Concerned about the placement rates of students.

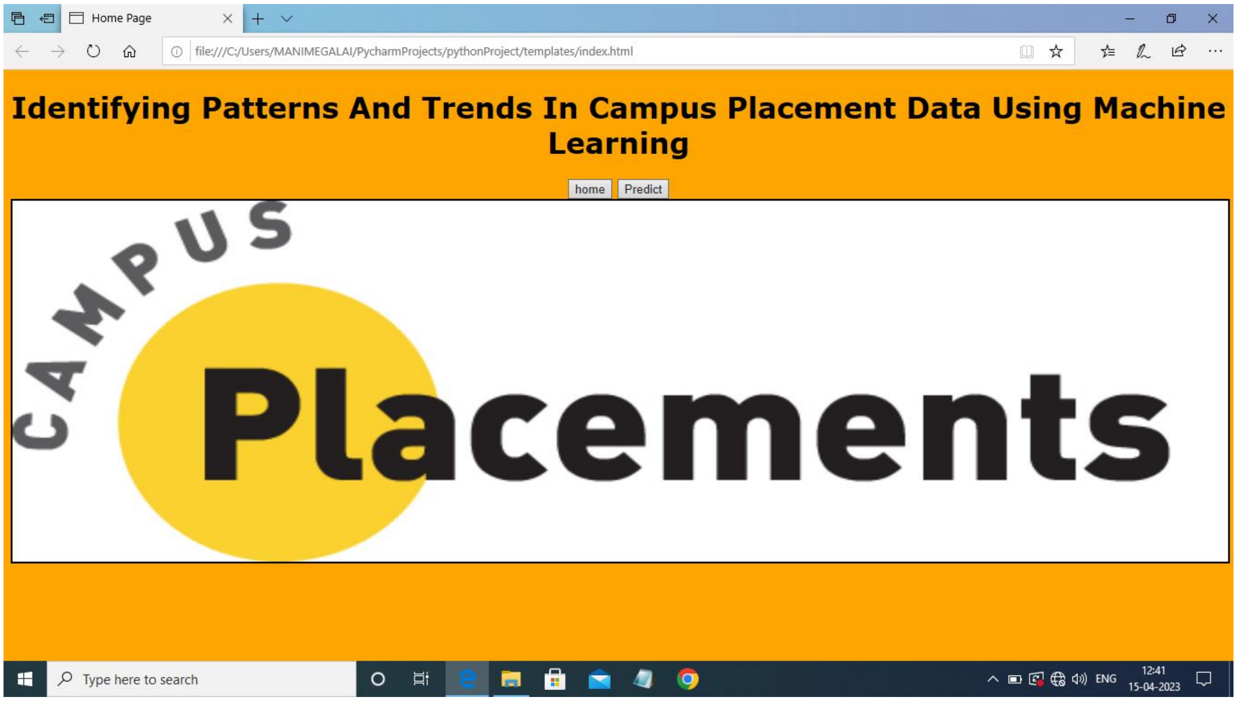
Curious to know the reasons behind the preferences of students and recruiters.

Hopeful that they can identify areas of improvement and enhance the placement process.

2.2 Ideation & Brainstorming Map

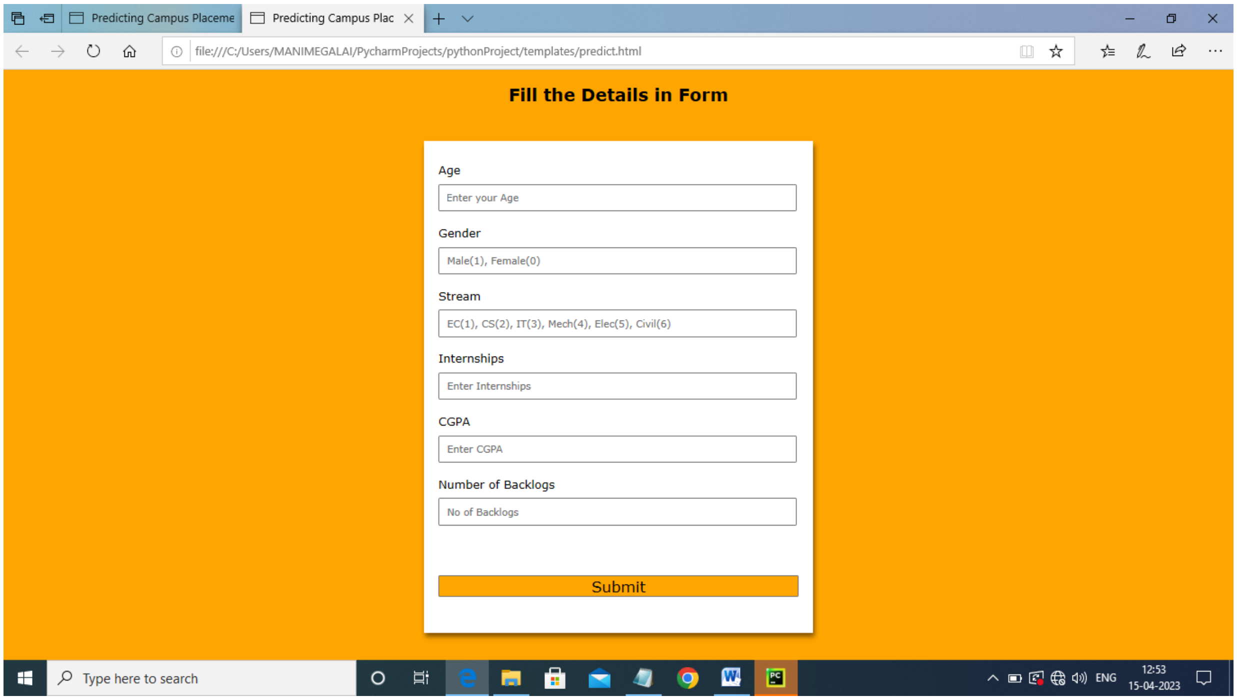


3. RESULT


Home Page
file:///C:/Users/MANIMEGALAI/PycharmProjects/pythonProject/templates/index.html

Identifying Patterns And Trends In Campus Placement Data Using Machine Learning

home
Predict
CAMPUS
Placements
Type here to search
ENG
12:41
15-04-2023


Predicting Campus Placeme
Predicting Campus Plac
file:///C:/Users/MANIMEGALAI/PycharmProjects/pythonProject/templates/predict.html
Type here to search
ENG
12:53
15-04-2023

Fill the Details in Form

Age

Enter your Age

Gender

Male(1), Female(0)

Stream

EC(1), CS(2), IT(3), Mech(4), Elec(5), Civil(6)

Internships

Enter Internships

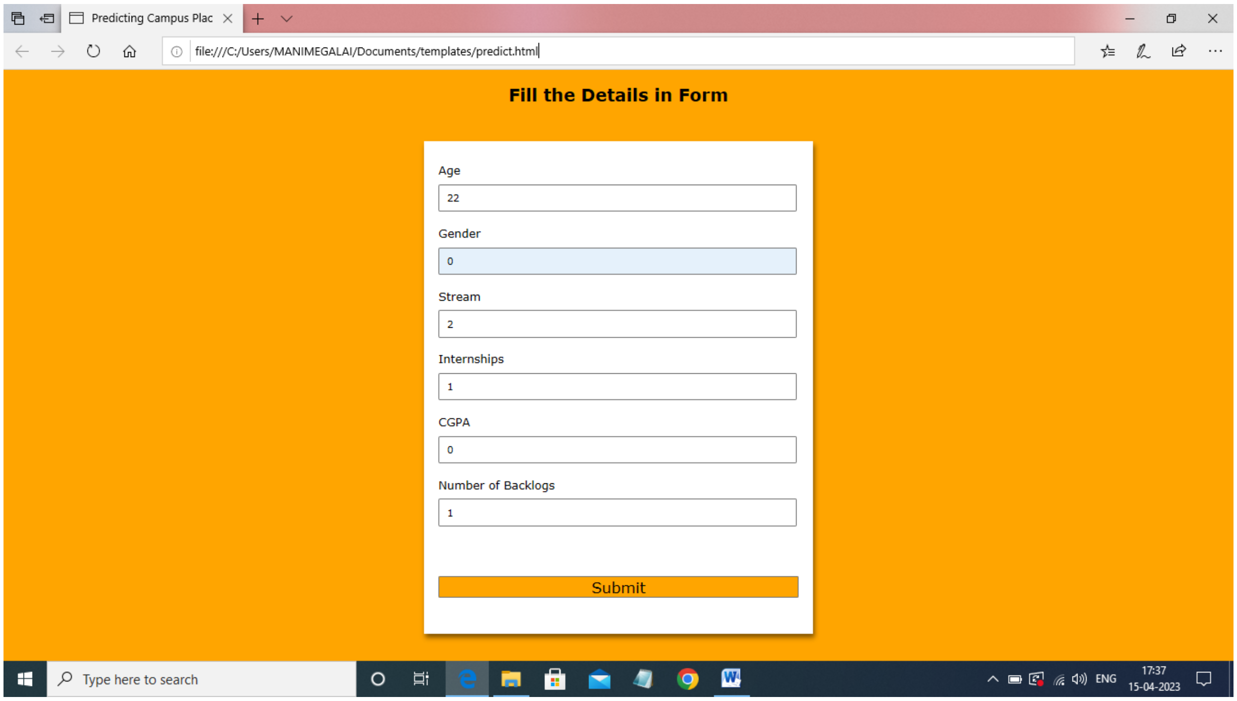
CGPA

Enter CGPA

Number of Backlogs

No of Backlogs

Submit


Predicting Campus Plac
file:///C:/Users/MANIMEGALAI/Documents/templates/predict.html
Type here to search
ENG
17:37
15-04-2023

Fill the Details in Form

Age

22

Gender

0

Stream

2

Internships

1

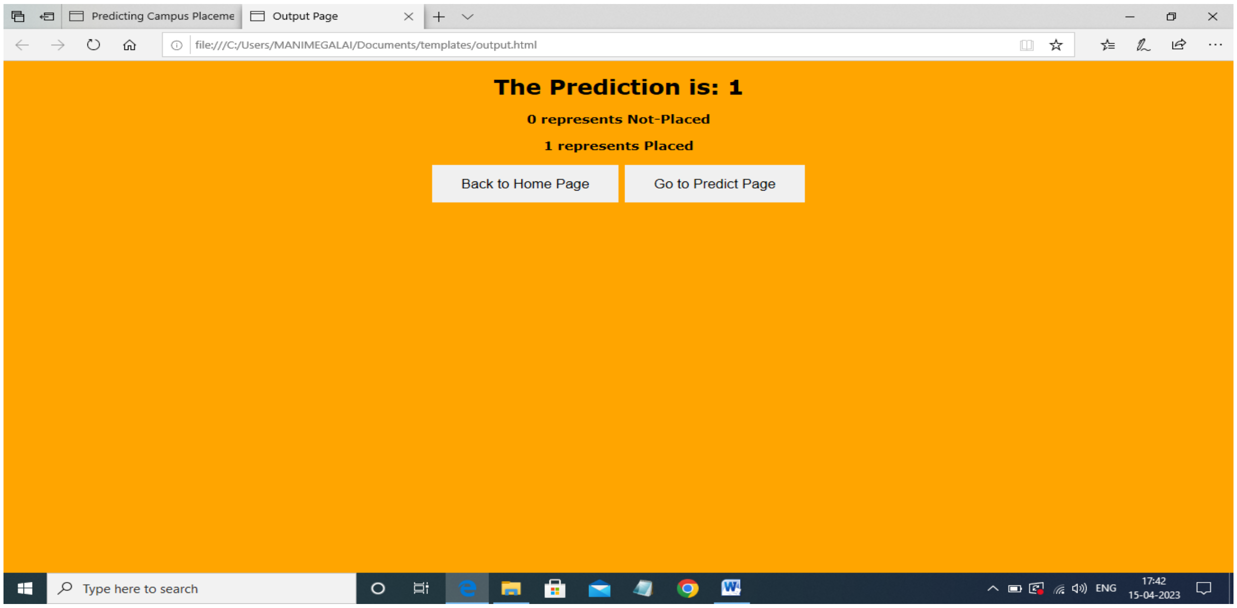
CGPA

0

Number of Backlogs

1

Submit


Predicting Campus Placeme
Output Page
file:///C:/Users/MANIMEGALAI/Documents/templates/output.html
Type here to search
ENG
17:42
15-04-2023

The Prediction is: 1

0 represents Not-Placed

1 represents Placed

Back to Home Page
Go to Predict Page

4. 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.

5. APPLICATIONS:

- Predicting future placement trends: Machine learning algorithms can analyze past data and identify patterns and trends that can help predict future placement trends, such as which companies are likely to participate in campus placement, which job roles are in demand, and what salaries can be expected.
- Personalized recommendations for students: Machine learning algorithms can provide personalized recommendations to students based on their unique characteristics and placement history, such as which companies and job roles they are most likely to be successful in and what skills they need to improve.
- Identifying factors affecting placement: Machine learning algorithms can identify the factors that affect placement, such as academic performance, extracurricular activities, and work experience, and help students optimize these factors to improve their chances of placement.
- Monitoring placement progress: Machine learning algorithms can monitor placement progress and provide alerts when a student is falling behind or needs additional support to improve their chances of placement.
- Colleges, Univercities,etc.,

6. CONCLUSION:

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.

7. FUTURE SCOPE:

- Personalized recommendations for students: Machine learning algorithms can provide personalized recommendations to students based on their unique characteristics and placement history, such as which companies and job roles they are most likely to be successful in and what skills they need to improve.

- Identifying factors affecting placement: Machine learning algorithms can identify the factors that affect placement, such as academic performance, extracurricular activities, and work experience, and help students optimize these factors to improve their chances of placement.

- Monitoring placement progress: Machine learning algorithms can monitor placement progress and provide alerts when a student is falling behind or needs additional support to improve their chances of placement.

8. APPENDIX

A. Source Code

Importing the Libraries

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

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	1	6	0	0	1

			Technology					
3	21	Male	Information Technology	0	8	0	1	1
4	22	Male	Mechanical	0	8	1	0	1

Handling Missing Values

```
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
```

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

Handling Outliers

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

Handling Catogarical Values

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

```
df()
```

Univariate Analysis

CGPA

```
plt.figure(figsize=(12,5))

plt.subplot(121)

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

PlacedorNot

```
plt.figure(figsize=(12,5))

plt.subplot(121)

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

Bivariate Analysis

```
plt.figure(figsize=(18,4))
plt.subplot(141)
sns.countplot(x=df['Gender'])
plt.subplot(142)
sns.countplot(x=df['Stream'])
plt.show()
```

Multivariate Analysis

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

```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.swarmplot(x=df['PlacedOrNot'], y=df['CGPA'], hue=df['Stream'])
```

Scaling the Data

```
# separate features and target variable
x = df.drop(['HistoryOfBacklogs'], axis=1)
y = df['Internships']

# create a StandardScaler object
sc = StandardScaler()

# standardize the values of the features in x
x_bal = sc.fit_transform(x)

# print the standardized dataset
print(x_bal)


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

Splitting the Data into Train and Test

```
# check the dataframe columns
print(df.columns)

# convert categorical variables to numerical using one-hot encoding
if 'Gender' in df.columns and 'Stream' in df.columns:
    df = pd.get_dummies(df, columns=['Gender', 'Stream'], drop_first=True)

# separate features and target variable
X = df.drop(['PlacedOrNot'], axis=1)

# create a StandardScaler object
scaler = StandardScaler()

# standardize the values of the features in X
standardized_data = scaler.fit_transform(X)

# assign the standardized features to X
X = standardized_data

# assign the "PlacedOrNot" target variable to Y
Y = df['PlacedOrNot']

# split the dataset into training and testing sets
```

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

# print the shape of the training and testing sets
print("X_train shape:", X_train.shape)
print("Y_train shape:", Y_train.shape)
print("X_test shape:", X_test.shape)
print("Y_test shape:", Y_test.shape)
```

Model Building

SVM Model

```
# create an SVM classifier with a linear kernel
classifier = svm.SVC(kernel='linear')
# train the classifier on the training data
classifier.fit(X_train, Y_train)
# print the accuracy of the classifier on the training and testing data
print("Training accuracy:", classifier.score(X_train, Y_train))
print("Testing accuracy:", classifier.score(X_test, Y_test))


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)
```

KNN Model

```
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 mode
                                                        l to the training set
knn_temp_pred = knn_temp.predict(X_test)                        # Predict on t

                                                                   he test set
```

```
score = metrics.accuracy_score(Y_test, knn_temp_pred)*100   # Get accuracy
if score >= best_score["Regular"] and score < 100:          # Store best p
                                                                     arams
        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
```

Artificial Neural Network Model

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

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
'])
```

Model Deployment

```
import pickle

pickle.dump(knn, open("placement.pk1", 'wb'))
model=pickle.load(open('placement.pkl', 'rb'))
```

Building Html pages

```
<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-->
<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(5)">
<input type="number" id="sen4" name="sen4" placeholder="Internship">
<input type="number" id="sen5" name="sen5" placeholder="CGPA">
<input type="number" id="sen6" name="sen6" placeholder="Number of backlog"
>
<input type="submit"value="submit">

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

</div>
</section><--End about us section-->




<section id="hero"class="d-flex flex-column justify-conter">
  <div class="container">
  <div class="row justify-content-center">
    <div class="col-xl-8">
    <h1>The prediction is :{{y}}</h1>
    <h3> 0 represents not-laced</h3>
    <h3> 1 represents placed<h2>

    </div>
    </div>
    </div>
    </section>
```

Build python code

```
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')
```

Render HTML Pages

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

Mainfunction

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

Video Demonstration Link:

https://drive.google.com/file/d/1dR_uiYHxXiTRre-7f2GELTA9z-hXLfv4/view?usp=drivesdk