



IDENTIFYING PATTERNS AND
TRENDS IN CAMPUS PLACEMENT DATA
USING MACHINE LEARNING
Project Based Experiential Learning Program

INDEX

1. INTRODUCTION

1.1 Overview

1.2 Purpose

2. PROBLEM AND DEFINITION & DESIGN THINKING

2.1 Empathy Map

2.2 Ideation & Brainstorming Map

3. RESULT

4. ADVANTAGE & DISADVANTAGES

5. APPLICATION

6. CONCLUSION

7. FUTURE SCOPE

8. APPENDIX

A PROJECT REPORT

CATEGORY: Machine Learning

PROJECT TITLE: Identifying Patterns and Trends in Campus Placement Data using Machine Learning

NM_ID: 6333E11245B530094D8489CFFAE74877

TEAM LEADER: DEEPAK G M

TEAM MEMBERS: ARTHI R

BHANU USHMA M

BAGAVATRHI DEVI K

Identifying Patterns and Trends in Campus Placement Data using Machine Learning

Introduction

1.1 Overview

Machine learning algorithms can be used to analyze campus placement data and identify patterns and trends. These algorithms can automatically learn from historical data and identify patterns and relationships between different variables, such as student qualifications, employer requirements, and placement outcomes. Machine learning is a method of data analysis that automates analytical model building. These models help you to make a trend analysis of placements data, to predict a placement rate for the students of an upcoming year which will help to analyze the performance during placements. Identifying patterns and trends in campus placement data is an essential task for educational institutions and employers. This can help them understand the job market, identify in-demand skills, and improve their training programs.

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

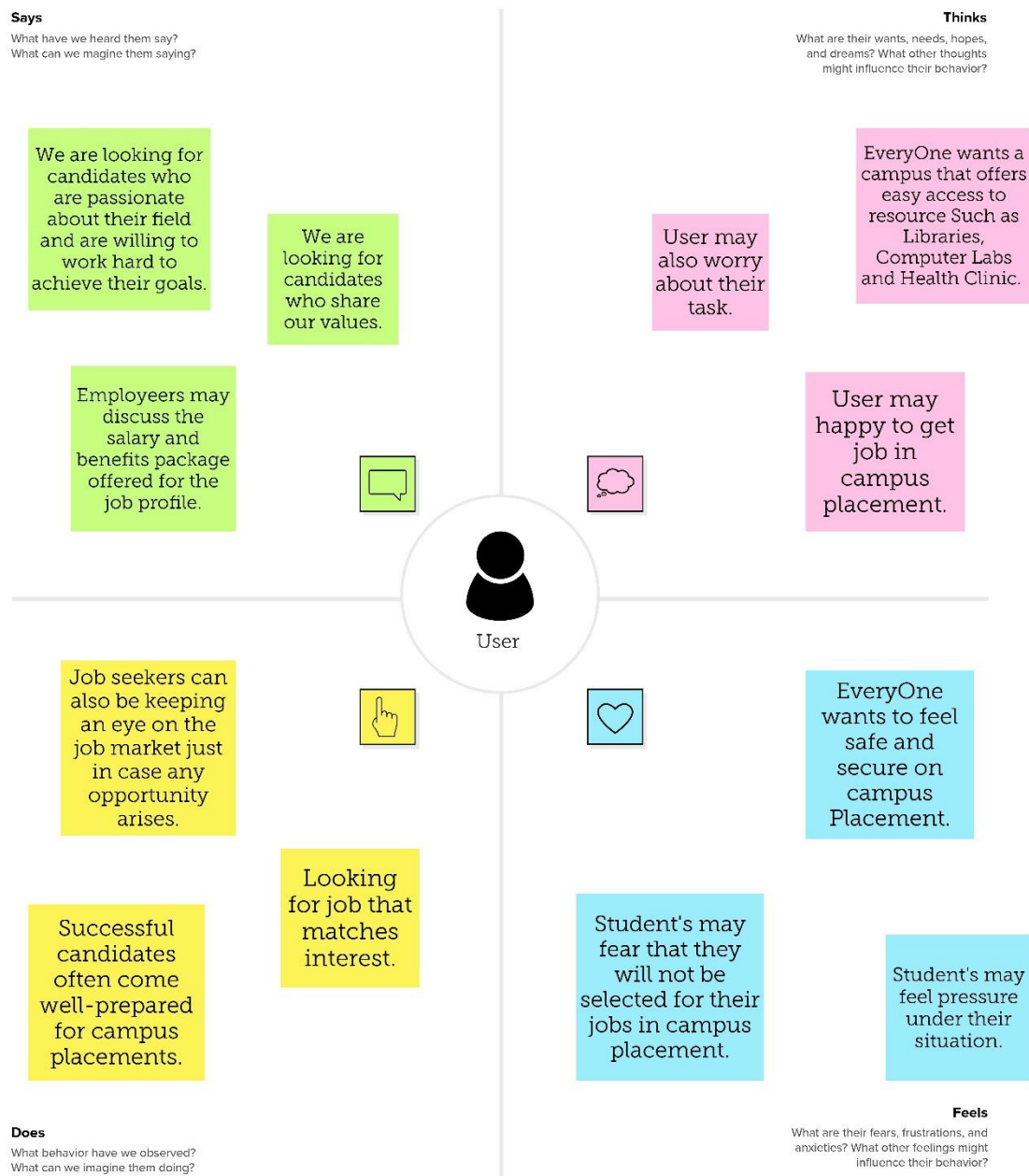
The purpose of identifying patterns and trends in campus placement data using machine learning is to gain insights into the job market and improve the effectiveness of training programs and hiring strategies. By analyzing historical data, machine learning algorithms can identify patterns and relationships between different variables that can influence the placement outcome, such as student qualifications, employer requirements, and placement outcomes.

The insights gained from analyzing campus placement data using machine learning can help educational institutions and employers make data-driven decisions to improve their training programs and hiring strategies. For example, educational institutions can use these insights to identify the most in-demand skills and adjust their training programs accordingly. Employers can use these insights to identify the most qualified candidates and improve their hiring strategies.

Overall, the purpose of identifying patterns and trends in campus placement data using machine learning is to improve the alignment between the skills and qualifications of students and the requirements of employers, ultimately leading to better job outcomes for students and a more efficient job market.

Problem Definition & Design Thinking

2.1 Empathy Map




2.1 Problem Definition & Design Thinking



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 **10 minutes** to prepare

 **1 hour** to collaborate

 **2-8 people** recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 **10 minutes**

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#)



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

⌚ 5 minutes

PROBLEM
How might we [your problem statement]?

**Key rules of brainstorming**

To run a smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

⌚ 10 minutes

Person 1 - (Arthi R)

Planning.

Take an Survey.

Create an job Offer.

Post vacancy in social media platform.

Person 2 - (Bagavathi Devi K)

Taking Notes from Articles and Find a solution.

Work Flow Diagram.

practice your skils.

Conduct workshop and seminar.

Person 3 - (Bhanu Ushma M)

Gathering information about an Campus Placement.

Collaborate with companies

Make an proper resume.

Testing the Model.

Person 4 - (Deepak G M)

Make an internship with popular companies.

Identify your strengths and weakness.

Attended many interviews and Train Yourself.

collecting feedback from the unemployed person.

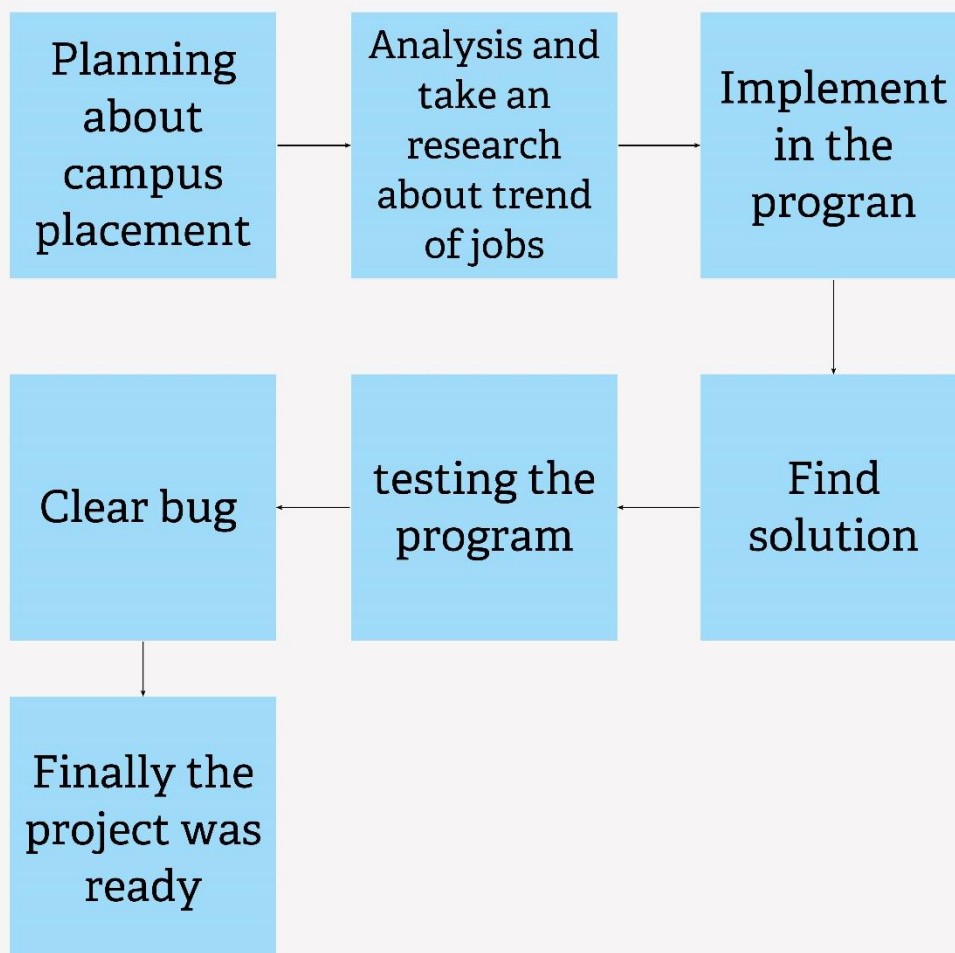
3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

Group Flow Ideas

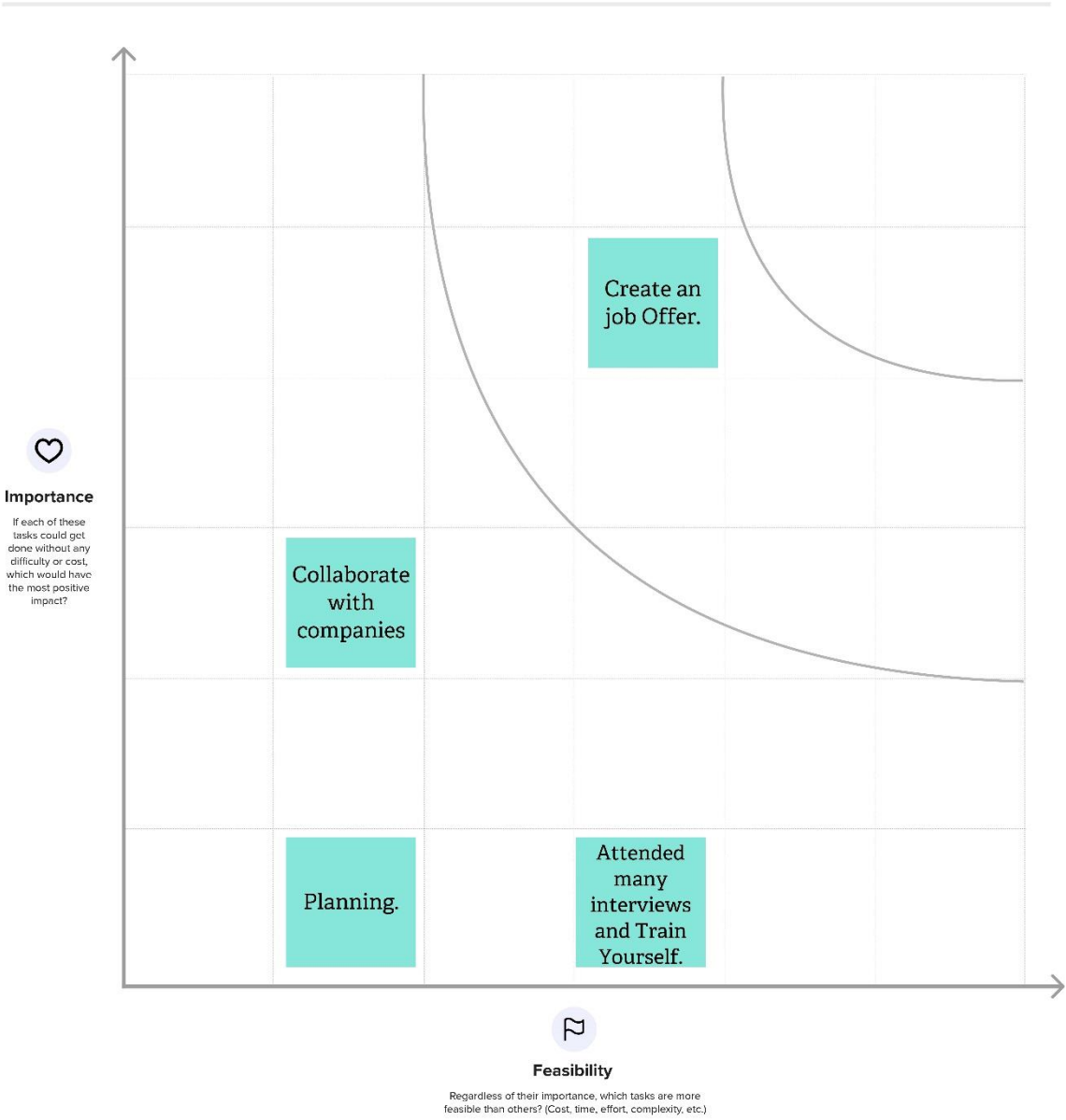


4

Prioritize

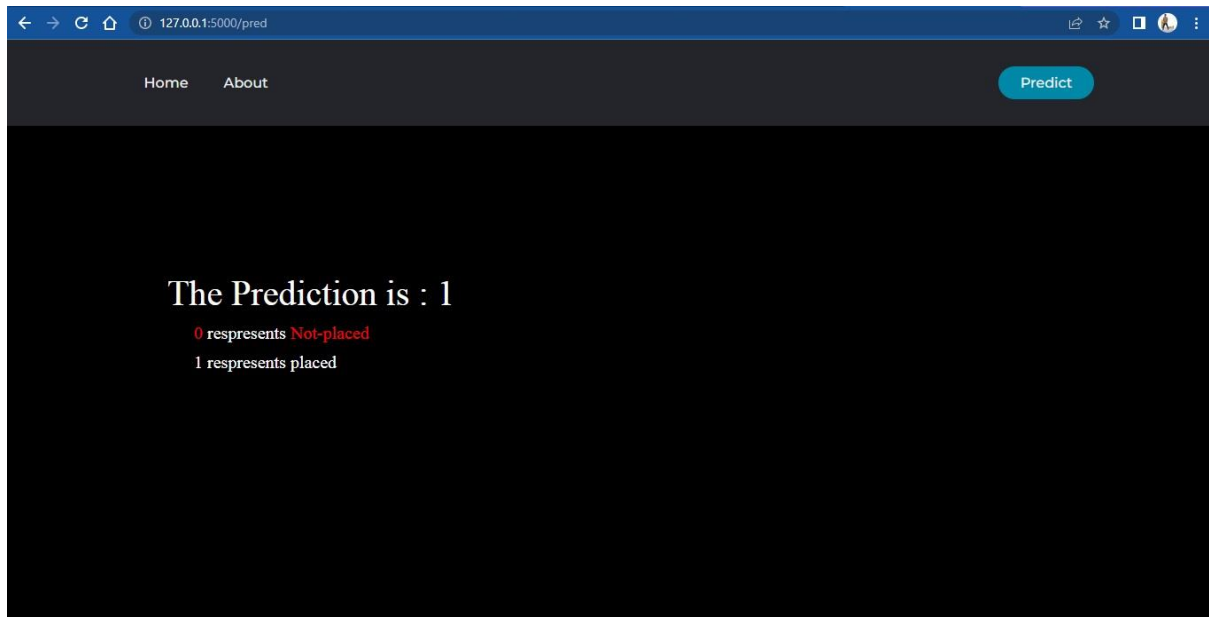
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



Result

Output:



ADVANTAGES & DISADVANTAGES

Advantages

- **Data-Driven Decision Making:** By using machine learning to analyze historical data, educational institutions and employers can make data-driven decisions about their training programs and hiring strategies. This can lead to more effective and efficient programs and strategies, resulting in better outcomes for students and employers.
- **Improved Predictive Ability:** Machine learning algorithms can use historical data to identify patterns and relationships between different variables, allowing them to make more accurate predictions about future outcomes. This can help educational institutions and employers better understand the job market and adjust their programs and strategies accordingly.
- **Scalability:** Machine learning algorithms can be applied to large datasets, making it possible to analyze large amounts of campus placement data quickly and efficiently. This scalability allows for more comprehensive and detailed analyses, leading to more accurate and insightful results.
- **Efficiency:** Machine learning algorithms can automate the analysis of campus placement data, reducing the time and resources required for manual analysis. This allows educational institutions and employers to make data-driven decisions more efficiently and effectively.
- **Flexibility:** Machine learning algorithms can be customized and adapted to suit the specific needs of educational institutions and employers. This flexibility allows for more targeted analyses and insights, leading to more effective programs and strategies.

Disadvantages

- **Data Quality:** The accuracy and completeness of the data used to train the machine learning algorithm can significantly impact the quality of the insights gained. If the data is incomplete or inaccurate, the algorithm may learn incorrect patterns and relationships, leading to incorrect insights.
- **Bias:** Machine learning algorithms can be biased if the data used to train them is biased. This can result in the perpetuation of existing biases and discrimination in the job market.
- **Overfitting:** Machine learning algorithms can be prone to overfitting, which means that they are too closely fitted to the training data and may not generalize well to new data. This can lead to incorrect predictions and insights.
- **Interpretability:** Machine learning algorithms can be difficult to interpret, especially for non-technical stakeholders. This can make it challenging to explain the insights gained from the analysis and to make data-driven decisions based on those insights.
- **Cost:** Implementing machine learning algorithms for analyzing campus placement data can be costly in terms of resources and expertise required. This can make it difficult for smaller educational institutions and employers to utilize these methods effectively.
- **Overall,** the disadvantages of identifying patterns and trends in campus placement data using machine learning include potential issues with data quality, bias, overfitting, interpretability, and cost. These factors should be carefully considered when deciding whether to use machine learning for analyzing campus placement data.

APPLICATION

Application of Job Prediction

Identifying patterns and trends in campus placement data using machine learning can provide valuable insights into the job market, employer preferences, and student performance. Here are some potential applications of machine learning in campus placement data analysis:

- **Predictive modeling:** Machine learning algorithms can be used to develop predictive models that can forecast future job market trends, identify high-demand industries, and predict which students are likely to receive job offers. These models can help universities and colleges tailor their programs to meet the needs of the job market and help students prepare for the job search.
- **Clustering and segmentation:** Machine learning algorithms can group students based on their academic performance, skills, and job preferences. These groups can be used to tailor career services and job placement assistance to meet the needs of each student segment.
- **Natural language processing:** Natural language processing (NLP) can be used to analyze job descriptions and identify the most in-demand skills and qualifications. This information can be used to develop course offerings and training programs that align with industry needs.
- **Sentiment analysis:** Sentiment analysis can be used to analyze job reviews and identify which companies are preferred by students. This information can be used to target recruiting efforts and develop partnerships with preferred employers.
- **Anomaly detection:** Machine learning algorithms can be used to detect outliers in placement data, such as unusually high or low salaries or job offer rates. These anomalies can be further investigated to identify potential biases or other factors that may be affecting placement outcomes.

CONCLUSION

Conclusion:

In conclusion, machine learning can be a valuable tool for analyzing campus placement data and identifying patterns and trends that can inform decisions related to academic programs, career services, and employer partnerships. As machine learning technology continues to advance, it is likely that its use in campus placement data analysis will become even more sophisticated and effective in providing actionable insights.

FUTURE SCOPE

Future Scope:

The future scope of identifying patterns and trends in campus placement data using machine learning is promising. As machine learning technology continues to advance, it is likely that new techniques and algorithms will emerge that can provide even more accurate and valuable insights into the job market and student performance. Here are some potential future developments in this field:

- **Personalized recommendations:** Machine learning algorithms can be used to provide personalized recommendations to students based on their skills, interests, and job preferences. These recommendations can help students identify career paths that align with their strengths and goals and help them prepare for the job search.
- **Real-time data analysis:** With the growing availability of real-time data, machine learning algorithms can be used to analyze job market trends as they emerge. This can help institutions stay ahead of the curve and adapt their programs and services to meet changing industry needs.
- **Collaboration with employers:** Machine learning can be used to analyze employer preferences and job descriptions to identify which skills and qualifications are most in-demand. This information can be used to develop partnerships with employers and tailor academic programs to meet their needs.
- **Integration with other data sources:** Campus placement data can be integrated with other data sources, such as social media and online job postings, to provide a more comprehensive view of the job market. This can help institutions identify emerging trends and provide students with the most up-to-date information.

Overall, the future scope of identifying patterns and trends in campus placement data using machine learning is promising. As institutions continue to adopt and refine machine learning algorithms, they will be better equipped to prepare students for the job market and provide valuable insights into industry needs.

APPENDIX

Source Code

1. app.py(Source Code)

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

app = Flask(__name__)

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

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

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

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

@app.route('/pred', methods=['post'])
def pred():
    sen1 = request.form['sen1']
    sen2 = request.form['sen2']
    sen3 = request.form['sen3']
    sen4 = request.form['sen4']
    sen5 = request.form['sen5']
    sen6 = request.form['sen6']
    variables = [[int(sen1), int(sen2), int(sen3), int(sen4), int(sen5), int(sen6)]]

    model.predict(variables)
    output = model.predict(variables)
    return render_template('submit.html', Y=output[0])

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

2. home.html

```
<!DOCTYPE html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<title>Navbar</title>
```

```
<meta name="description" content="">
```

```
<meta name="viewport" content="width=device-width, initial-  
scale=1">
```

```
<link rel="stylesheet" href="static/styles.css">
```

```
<link  
href="https://fonts.googleapis.com/css?family=Montserrat:500&displ  
ay=swap" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<header>
```

```
<nav>
```

```
<ul class="nav__links">
```

```
<li><a href="/">Home</a></li>
```

```
<li><a href="/about">About</a></li>
```

```
</ul>
```

```
</nav>
```

```
<a class="cta" href="/predict">Predict</a>
```

```
</header>
```



```

    <div class="idp-text">
        <h1>IDENTIFYING PATTERNS AND TRENDS IN
        CAMPUS PLACEMENT DATA USING MACHINE
        LEARNING</h1>
    </div>
    <div class="idp-b">
        <a class="cta" href="/predict">Predict</a>
    </div>
</body>
</html>

```

3. about.html

```

<!DOCTYPE html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Prediction</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet" href="static/styles.css">
    <link
href="https://fonts.googleapis.com/css?family=Montserrat:500&displ
ay=swap" rel="stylesheet">
</head>
<body>

```

<header>

<nav>

<ul class="nav__links">

Home

About

</nav>

Predict

</header>

<div class="idp-p">

<p>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.

</p>

<p>

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.

</p>

</div>

<div class="idp-c">

Predict

</div>

</body>

</html>

4. predict.html

<!DOCTYPE html>

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>Prediction </title>

<meta name="description" content="">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="static/styles.css">

<link

href="https://fonts.googleapis.com/css?family=Montserrat:500&display=swap" rel="stylesheet">

</head>

<header>

<nav>

<ul class="nav__links">

Home

About

</nav>

Predict

</header>

<div>

<div class="login-box">

<h2>Fill The Details</h2>

<form action="/pred" method= "POST">

<div class="custom">

<label class="label-input">Age</label>

<select class="user-box-op" id="sen1 " name="sen1">

<option value="" selected disabled hidden >Choose your Age</option>

<option value="22">22</option>

<option value="23">23</option>

<option value="24">24</option>

```
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
<option value="32">32</option>
<option value="33">33</option>
<option value="34">34</option>
<option value="35">35</option>
<option value="36">36</option>
<option value="37">37</option>
<option value="38">38</option>
<option value="39">39</option>
<option value="40">40</option>
<option value="41">41</option>
<option value="42">42</option>
<option value="43">43</option>
<option value="44">44</option>
<option value="45">45</option>
</select>

</div>

<div class="custom">
  <label class="label-input">Gender</label>
```

```
<select class="user-box-op" id="sen2"
name="sen2" >
    <option class=wnc value="" selected disabled
hidden >Select Gender</option>
```

```
    <option value="0">Male</option>
    <option value="1">Female</option>
</select>
</div>
```

```
<div class="custom">
    <label class="label-input">Stream</label>
    <select class="user-box-op" id="sen3"
name="sen3">
    <option value="" selected disabled hidden >Select
Education</option>
    <option value="0">Computer
Science</option>
    <option value="1">Information
Technology</option>
    <option value="2">Electronics And
Communication</option>
    <option value="3">Mechanical</option>
    <option value="5">Electrical</option>
    <option value="6">Civil</option>
</select>
</div>
```

```

<div class="custom">
  <label class="label-
input">Internships</label>
  <select class="user-box-op" id="sen4"
name="sen4">
    <option value="" selected disabled hidden
>Select Option</option>
    <option value="1">Yes</option>
    <option value="0">No</option>
  </select>
</div>

```

```

<div class="custom">
  <label class="label-input">CGPA</label>
  <select class="user-box-op" id="sen5"
name="sen5">
    <option value="" selected disabled hidden
>Select CGPA</option>
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
  </select>
</div>

```

```

<div class="custom">

```

```

        <label class="label-input">You have any
arrears in Exams?</label>

        <select class="user-box-op" id="sen6"
name="sen6">

            <option value="" selected disabled hidden
>Select Option</option>

            <option value="1">Yes</option>
            <option value="0">No</option>
        </select>
    </div>

```

```

        <input type = "Submit" value="Predict"
class="cta" >

```

```

    </form>

```

```

</div>

```

```

</div>

```


5. submit.html

```
<!DOCTYPE html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<title>prediction</title>
```

```
<meta name="description" content="">
```

```
<meta name="viewport" content="width=device-width, initial-  
scale=1">
```

```
<link rel="stylesheet" href="static/styles.css">
```

```
<link  
href="https://fonts.googleapis.com/css?family=Montserrat:500&displ  
ay=swap" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<header>
```

```
<nav>
```

```
<ul class="nav__links">
```

```
<li><a href="/">Home</a></li>
```

```
<li><a href="/about">About</a></li>
```

```
</ul>
```

```
</nav>
```

```
<a class="cta" href="/predict">Predict</a>
```

```
</header>
```

```

    <div class="idp-text">
      <h1>The Prediction is : {{ Y }} </h1>
      <h3><span>0</span> represents <span>Not-
placed</span></h3>
      <h3>1 represents placed</h3>

    </div>

  </body>
</html>

```

6. styles.css(Stylesheet)

```

@import
url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,70
0&display=swap');
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

```

```

header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

```

```
padding: 30px 10%;  
background-color: #24252a;  
}
```

```
.nav__links a,  
.cta,  
.overlay__content a {  
  font-family: "Montserrat", sans-serif;  
  font-weight: 500;  
  color: #edf0f1;  
  text-decoration: none;  
}
```

```
.nav__links {  
  list-style: none;  
  display: flex;  
  
}
```

```
.nav__links li {  
  padding: 0px 20px;  
}
```

```
.nav__links li a {  
  transition: color 0.3s ease 0s;  
}
```

```
.nav__links li a:hover {  
    color: #0088a9;  
}
```

```
.cta {  
    padding: 9px 25px;  
    background-color: rgba(0, 136, 169, 1);  
    border: none;  
    border-radius: 50px;  
    cursor: pointer;  
    transition: background-color 0.3s ease 0s;  
}
```

```
.cta:hover {  
    background-color: rgba(0, 136, 169, 0.8);  
}
```

```
body{  
    background-color:black;  
}
```

```
.idp-text{  
    position:absolute;  
    top: 45%;  
    left: 50%;  
    transform:translate(-50% , -30%);
```

```
        user-select:none;
    }
.idp-text h1{
    font-size:40px;
    color:white;
    font-weight: lighter;
    width:1000px;
}
.idp-text h3{
    color: white;
    font-size: 20px;
    font-weight: lighter;
    padding-left: 30px;
    padding-top: 10px;
}
.idp-text h3 span{
    color: red;
}
.idp-b {
    position:absolute;
    top:50%;
    left:15%;
    margin:60px 380px;
}
.idp-p p{
    font-size:30px;
```

```

color:white;
font-weight: lighter;
text-align: justify-all;
padding: 10px 120px;
position: relative;
}
.idp-c {
position:absolute;
top:50%;
left:15%;
margin:275px 380px;
}
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button{
-webkit-appearance: none;
margin: 0;
}
input[type=submit]{
-moz-appearance: textfield;
}

.login-box {
position: absolute;
top: 65%;
left: 50%;
width: 400px;

```

```
padding: 40px;  
transform: translate(-50%, -50%);
```

```
box-sizing: border-box;  
box-shadow: 2px 2px 6px slategray;  
border-radius: 10px;  
}
```

```
.login-box h2 {  
  margin: 0 0 30px;  
  padding: 0;  
  color: #fff;  
  text-align: center;  
}
```

```
.login-box .user-box {  
  position: relative;  
}
```

```
.custom {  
  display: flex;  
  position: relative;  
}
```

```
.custom label {  
  color: #03e9f4;  
  position: relative;
```

```
font-size: 12px;
padding: 10px 0;
position: absolute;
top: -20px;
left: 0;
margin-bottom: 30px;
}
.custom .user-box-op {
position: relative;
top: 10px;
padding: 8px 0;
display: block;
outline: none;
max-width: 400px;
min-width: 80px;
z-index: 10;
margin-bottom: 30px;
border: none;
border-radius: 0;
border-bottom: 1px solid #fff;
background-color: black;
padding: 12px 55px 15px 15px;

}
```



```
.custom select{
  border: none;
  outline: none;
  background: transparent;
  appearance: none;
  border-radius: 0;
  margin: 0;
  display: block;
  width: 100%;
  padding: 15px 55px 15px 15px;
  font-size: 13px;
  color: #fff;
}
&:after{
  position: absolute;
  right: 0;
  top: 0;
  width: 50px;
  height: 100%;
  line-height: 38px;
  content: '\2228';
  text-align: center;
  color: #fff;
  font-size: 24px;
  z-index: -1;
}
```

7. PLACEMENT_PREDICTION.ipynb

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

df = pd.read_csv(r'content\collegeface.csv')
```

➤ READ THE DATASET

```
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

➤ DATA PREPARATION--HANDLING MISSING VALUES

```
df.info()
```

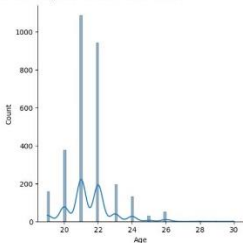
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2866 entries, 0 to 2865
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  --
 0   Age             2866 non-null   int64   
 1   Gender          2866 non-null   object  
 2   Stream          2866 non-null   object  
 3   Internships     2866 non-null   int64   
 4   CGPA            2866 non-null   int64   
 5   Hostel         2866 non-null   int64   
 6   HistoryOfBacklogs 2866 non-null   int64   
 7   PlacedOrNot     2866 non-null   int64   
dtypes: int64(6), object(2)
memory usage: 385.5+ KB
```

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

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

➤ HANDLING OUTLIERS

```
sns.distplot(df['Age'], kde=True,
             bins=binwidth.FacetGrid at 0x7fc7390b5930)
```



➤ HANDLING CATEGORICAL VALUES

```
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

```
df = df.drop(['Hostel'], axis=1)
df = df.replace(['Male'], [0])
df = df.replace(['Female'], [1])
df = df.replace(['Computer Science', 'Information Technology', 'Electronics And Communication', 'Mechanical', 'Electrical', 'Civil'], [6, 7, 8, 9, 5])

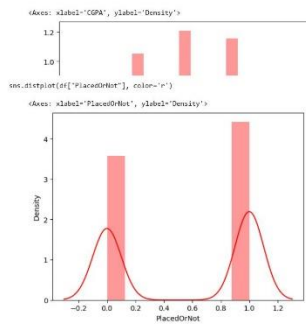
df.head()
```

	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

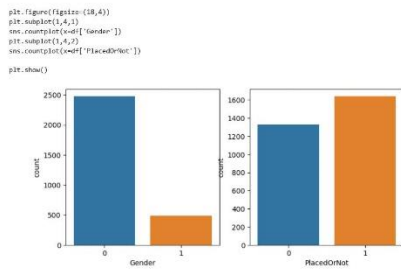
➤ VISUAL ANALYSIS - UNIVARIATE ANALYSIS

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

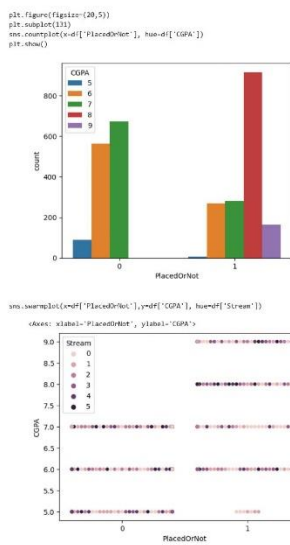
D



BIVARIATE ANALYSIS



Multivariate analysis



SCALING THE DATA

```

y=df["PlacedOrNot"]
X=df.drop(["PlacedOrNot"],axis=1)
X

```

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

2966 rows x 6 columns

SPLITTING THE DATA INTO TRAIN AND TEST

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

• SVM MODEL

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

classifier.fit(X_train, Y_train)

# SVM
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: 0.76849140488089
```

• KNN MODEL

```
best_k = "Regular"
best_score = "Regular"
for k in range(1, 20, 2):
    knn_clf = KNeighborsClassifier(n_neighbors=k)
    knn_clf.fit(X_train, Y_train)
    knn_pred = knn_clf.predict(X_test)
    score = metrics.accuracy_score(Y_test, knn_pred)
    if score > best_score["Regular"] and score > 100:
        best_score["Regular"] = score
        best_k["Regular"] = k

print("...Result...")
print("Best_k: {}".format(best_k))
knn_clf = KNeighborsClassifier(n_neighbors=best_k["Regular"])
knn_clf.fit(X_train, Y_train)
knn_pred = knn_clf.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, knn_pred)

print("...Result...")
print("Best_k: {}".format(best_k))
print("Score: {}".format(test_data_accuracy))
```

• ARTIFICIAL NEURAL NETWORK MODEL

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

classifier = Sequential()

classifier.add(layers.Dense(units=64, activation='relu', input_dim=4))
classifier.add(layers.Dense(units=64))

classifier.add(layers.Dense(units=64, activation='relu'))
classifier.add(layers.Dense(units=64))

classifier.add(layers.Dense(units=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=32, epochs=100)

Epoch 1/100 [.....] - 2s 1ms/step - loss: 3.1854 - accuracy: 0.5543
Epoch 2/100 [.....] - 0s 2ms/step - loss: 1.5348 - accuracy: 0.5897
Epoch 3/100 [.....] - 0s 1ms/step - loss: 0.9182 - accuracy: 0.4996
Epoch 4/100 [.....] - 0s 1ms/step - loss: 0.7561 - accuracy: 0.4933
Epoch 5/100 [.....] - 0s 2ms/step - loss: 0.7876 - accuracy: 0.4928
Epoch 6/100 [.....] - 0s 2ms/step - loss: 0.6868 - accuracy: 0.5195
Epoch 7/100 [.....] - 0s 2ms/step - loss: 0.6739 - accuracy: 0.5493
Epoch 8/100 [.....] - 0s 2ms/step - loss: 0.6624 - accuracy: 0.5594
Epoch 9/100 [.....] - 0s 2ms/step - loss: 0.6574 - accuracy: 0.5721
Epoch 10/100 [.....] - 0s 2ms/step - loss: 0.6482 - accuracy: 0.5784
Epoch 11/100 [.....] - 0s 2ms/step - loss: 0.6438 - accuracy: 0.5766
Epoch 12/100 [.....] - 0s 2ms/step - loss: 0.6509 - accuracy: 0.5548
Epoch 13/100 [.....] - 0s 2ms/step - loss: 0.6469 - accuracy: 0.5548
Epoch 14/100 [.....] - 0s 2ms/step - loss: 0.6378 - accuracy: 0.5766
Epoch 15/100 [.....] - 0s 2ms/step - loss: 0.6377 - accuracy: 0.5725
Epoch 16/100 [.....] - 0s 2ms/step - loss: 0.6330 - accuracy: 0.5822
Epoch 17/100 [.....] - 0s 2ms/step - loss: 0.6326 - accuracy: 0.5800
Epoch 18/100 [.....] - 0s 1ms/step - loss: 0.6318 - accuracy: 0.5856
Epoch 19/100 [.....] - 0s 2ms/step - loss: 0.6269 - accuracy: 0.6050
Epoch 20/100 [.....] - 0s 2ms/step - loss: 0.6156 - accuracy: 0.6168
Epoch 21/100 [.....] - 0s 2ms/step - loss: 0.6225 - accuracy: 0.5919
Epoch 22/100 [.....] - 0s 2ms/step - loss: 0.6227 - accuracy: 0.5995
Epoch 23/100 [.....] - 0s 2ms/step - loss: 0.6116 - accuracy: 0.5888
Epoch 24/100 [.....] - 0s 2ms/step - loss: 0.6352 - accuracy: 0.6118
Epoch 25/100 [.....] - 0s 1ms/step - loss: 0.6112 - accuracy: 0.6142
Epoch 26/100 [.....] - 0s 2ms/step - loss: 0.6117 - accuracy: 0.6092
Epoch 27/100 [.....] - 0s 2ms/step - loss: 0.6079 - accuracy: 0.6164
Epoch 28/100 [.....] - 0s 2ms/step - loss: 0.6072 - accuracy: 0.6275
Epoch 29/100 [.....] - 0s 2ms/step - loss: 0.5916 - accuracy: 0.6328

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