Identifying Patterns and Trends in Campus Placement Data using Machine Learning

1.INTRODUCTION

1.1 Overview

1.2 Purpose

2.Problem Definition & Design Thinking

- **2.1 Empathy Map screenshot**
- **2.2 Ideation & Brainstorming Map**

3.RESULT

- **A Source Code and final findings (Output) of the project along with screenshots.**

4.ADVANTAGES & DISADVANTAGES

- **List of advantages and disadvantages of the proposed solution**

5 APPLICATIONS

- **The areas where this solution can be applied**

6.CONCLUSION

- **Conclusion summarizing the entire work and findings.**

7.FUTURE SCOPE

- **Enhancements that can be made in the future.**

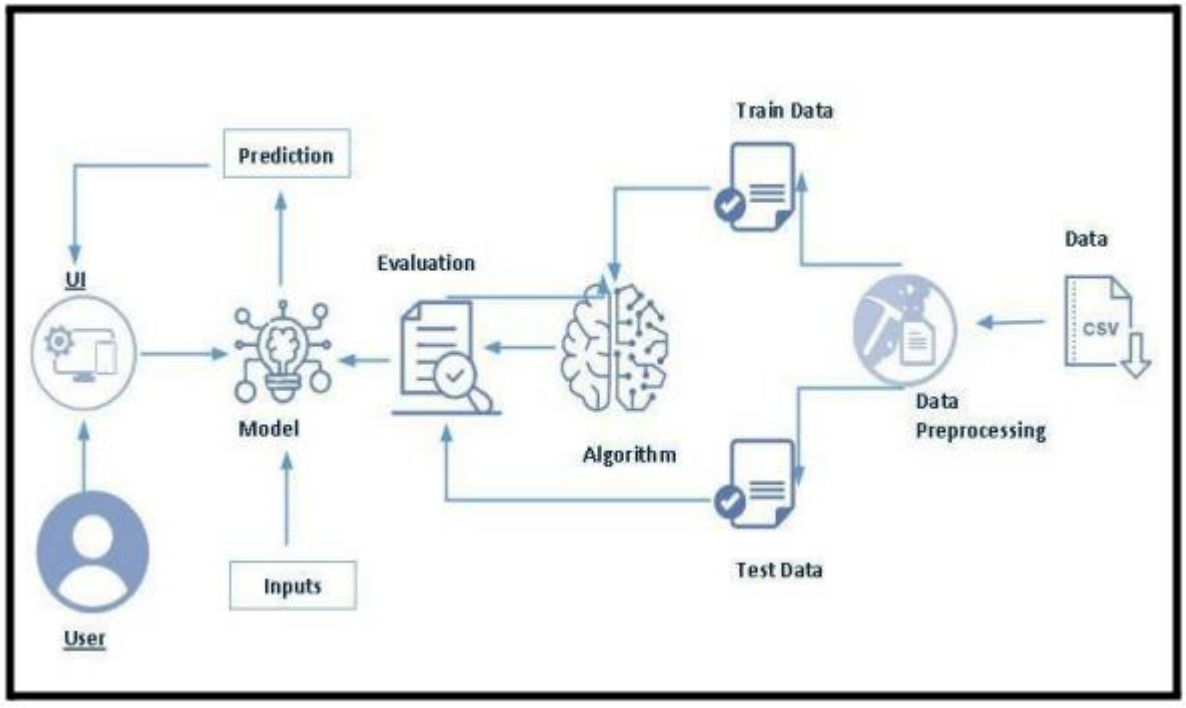
1. Introduction:

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.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyzes the input the prediction is showcased on the UI

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

Data Collection:

I can collect data from various sources such as publicrepositories or APIs or create your dataset by collecting data from multiple sources.

Visualizing and analyzing data:

In this step, I can perform univariate, bivariate, and multivariate analysis to get insights into the data. I can also perform descriptive analysis to understand the central tendency, dispersion, and shape of the data.

Data pre-processing:

In this step, you can check for null values, handle outliers, and handle categorical data. I can also split the data into train and test datasets.

Model building:

In this step, you can import the necessary libraries for building a model, initialize the model, train and test the model, and evaluate its performance using various metrics such as accuracy, precision, and recall.I can also save the model for future use.

Application building:

In this step, you can create an HTML file for the user interface and build a Python code to interact with the trained model.I can then deploy the application on a web server or cloud platform.

1.2 Purpose

This project aims to predict whether a student will get placed or not based on their academic performance and other attributes. The project can be used by educational institutions, students, and recruiters to analyze the factors that affect student placements and to make informed decisions about hiring.

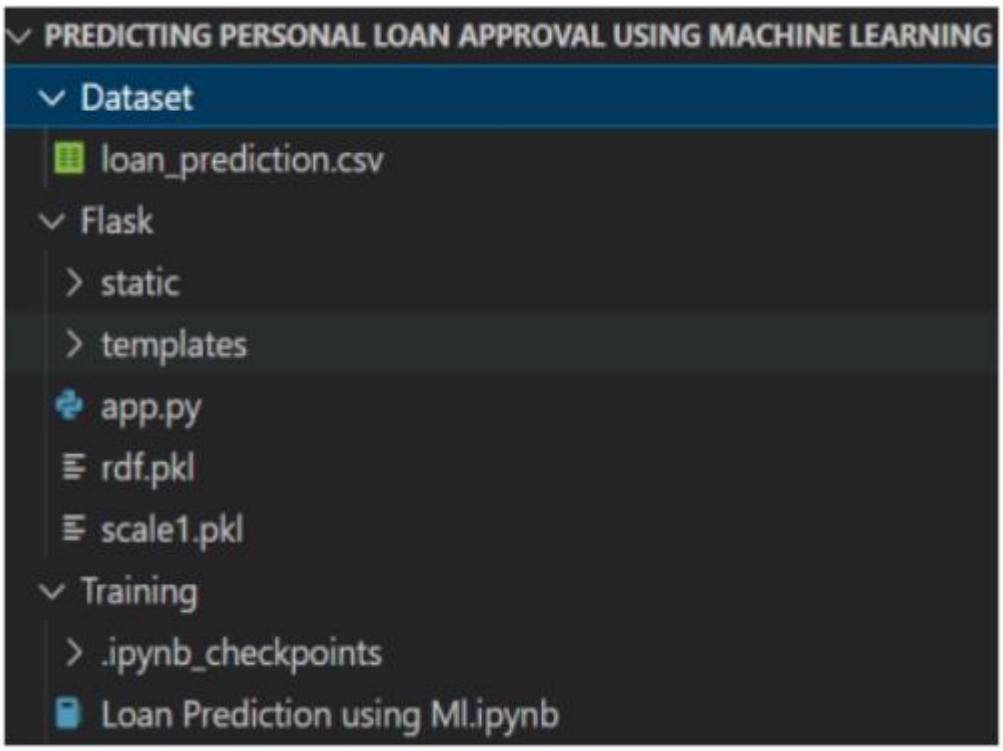
Educational institutions can use this project to analyze the performance of their students and identify areas for improvement. They can also use the insights gained from this project to provide targeted support to students who need it the most.

Students can use this project to understand which factors are most important for getting placed and to identify areas for improvement in their academic performance and skills.

Recruiters can use this project to analyze the factors that affect student placements and to make informed decisions about hiring. They can also use the insights gained from this project to identify the most promising candidates for their organization.

Project Flow:

Create the Project folder which contains files as shown below



● We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting

● rdf.pkl is our saved model. Further we will use this model for flask integration.

● Training folder contains a model training file

Milestone 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
     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 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.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
```

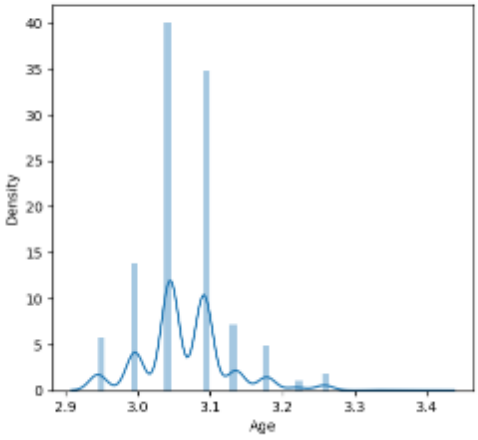
```
[ ] df.isnull().sum()
```

```
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']))
```

```
<ipython-input-5-5a8c293dc427>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(feature)
```



Handling Categorical Values

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

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

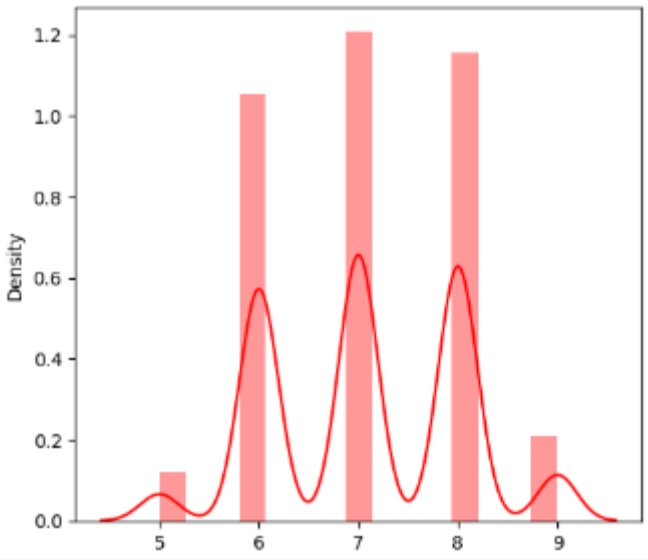
	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	0	2	1	8	1	1
1	21	1	0	0	7	1	1
2	22	1	1	1	6	0	1
3	21	0	1	0	8	1	1
4	22	0	3	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	0	1	0	7	0	0
2962	23	0	3	1	7	0	0
2963	22	0	1	1	7	0	0
2964	22	0	0	1	7	0	0
2965	23	0	5	0	8	0	1

2966 rows × 7 columns

Univariate analysis

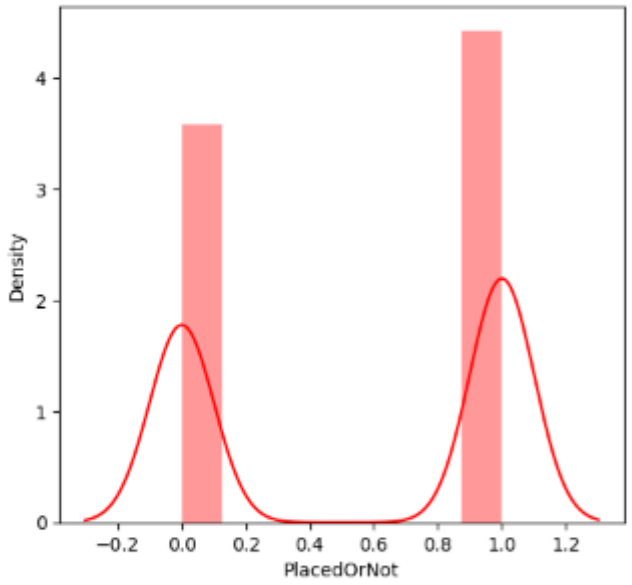
```
[7] plt.figure(figsize=(12,5))
    plt.subplot(121)
    sns.distplot(df['CGPA'],color='r')
```

```
<ipython-input-7-f92659182652>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['CGPA'],color='r')
<Axes: xlabel='CGPA', ylabel='Density'>
```



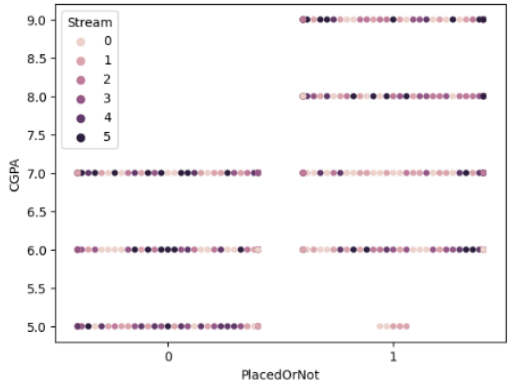
```
[8]  plt.figure(figsize=(12,5))
     plt.subplot(121)
     sns.distplot(df['PlacedOrNot'],color='r')
```

```
<ipython-input-8-5e468beb8a0d>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['PlacedOrNot'],color='r')
<Axes: xlabel='PlacedOrNot', ylabel='Density'>
```



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

```
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 69.2% of the points cannot be placed; you may want to decrease the size of the markers
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 69.5% of the points cannot be placed; you may want to decrease the size of the markers
  warnings.warn(msg, UserWarning)
<Axes: xlabel='PlacedOrNot',
ylabel='CGPA'>/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 83.1% of the points cannot be placed; you may want to decrease the size
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 82.9% of the points cannot be placed; you may want to decrease the size of the markers
  warnings.warn(msg, UserWarning)
```



Scaling the data

```
[13] sc = StandardScaler()
     x_bal = sc.fit_transform(df)
     x_bal
```

```
array([[ 0.20925372, -0.58649095, -0.10865065, ...,  1.40863479,
         1.54074243,  0.85166979],
       [-0.37388716,  1.70505616, -1.27408263, ...,  1.40863479,
         1.54074243,  0.85166979],
       [ 0.20925372,  1.70505616, -0.69136664, ..., -0.70990722,
        -0.64903775,  0.85166979],
       ...,
       [ 0.79239459,  1.70505616, -1.27408263, ..., -0.70990722,
         1.54074243,  0.85166979],
       [ 0.79239459, -0.58649095, -1.27408263, ..., -0.70990722,
         1.54074243, -1.174164  ],
       [ 0.79239459, -0.58649095,  0.47406534, ...,  1.40863479,
        -0.64903775, -1.174164  ]])
```

Splitting the Data into Train and Test

```
[14] # Split the data into training and testing sets
     # Split the data into input features (X) and target variable (Y)
     X = x_bal
     Y = df['PlacedOrNot']
     X_train, X_test, y_train, y_test = train_test_split(x_bal, df['PlacedOrNot'], test_size=0.2, random_state=42)
```

```
[15] # Define the SVM classifier with a linear kernel
     classifier = svm.SVC(kernel='linear')

     # Train the classifier on the training data
     classifier.fit(X_train, y_train)

     # Predict the output for the training set
     y_train_pred = classifier.predict(X_train)

     # Calculate the accuracy of the model on the training data
     training_accuracy = accuracy_score(y_train_pred, y_train)

     # Print the training accuracy
     print('Accuracy score of the training data:', training_accuracy)
```

```
Accuracy score of the training data: 1.0
```

```
# Initialize dictionaries to store best k and best score for each type of dataset
best_k = {"Regular": 0}
best_score = {"Regular": 0}

# Loop through odd k values from 3 to 49 and calculate accuracy for each type of dataset
for k in range(3, 50, 2):
    # Regular dataset
    knn_temp = KNeighborsClassifier(n_neighbors=k)
    knn_temp.fit(X_train, y_train)
    y_test_pred = knn_temp.predict(X_test)
    score = metrics.accuracy_score(y_test, y_test_pred) * 100
    if score >= best_score["Regular"] and score < 100:
        best_score["Regular"] = score
        best_k["Regular"] = k

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

# Instantiate the model with best k
knn = KNeighborsClassifier(n_neighbors=best_k["Regular"])

# Fit the model to the training set
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)

# Calculate accuracy score
test = metrics.accuracy_score(y_test, y_pred)
```

```
---Results---
K: {'Regular': 49}
Score: {'Regular': 97.77777777777777}
```

```
# Define X_train as a pandas DataFrame
X_train = data=[[1, 2, 3, 4, 5, 6], [2, 3, 4, 5, 6, 7], [3, 4, 5, 6, 7, 8]]
y_train = [0, 1, 0]
# Fit the model on X_train and Y_train
classifier.fit(X_train, y_train, batch_size=28, epochs=100)
```

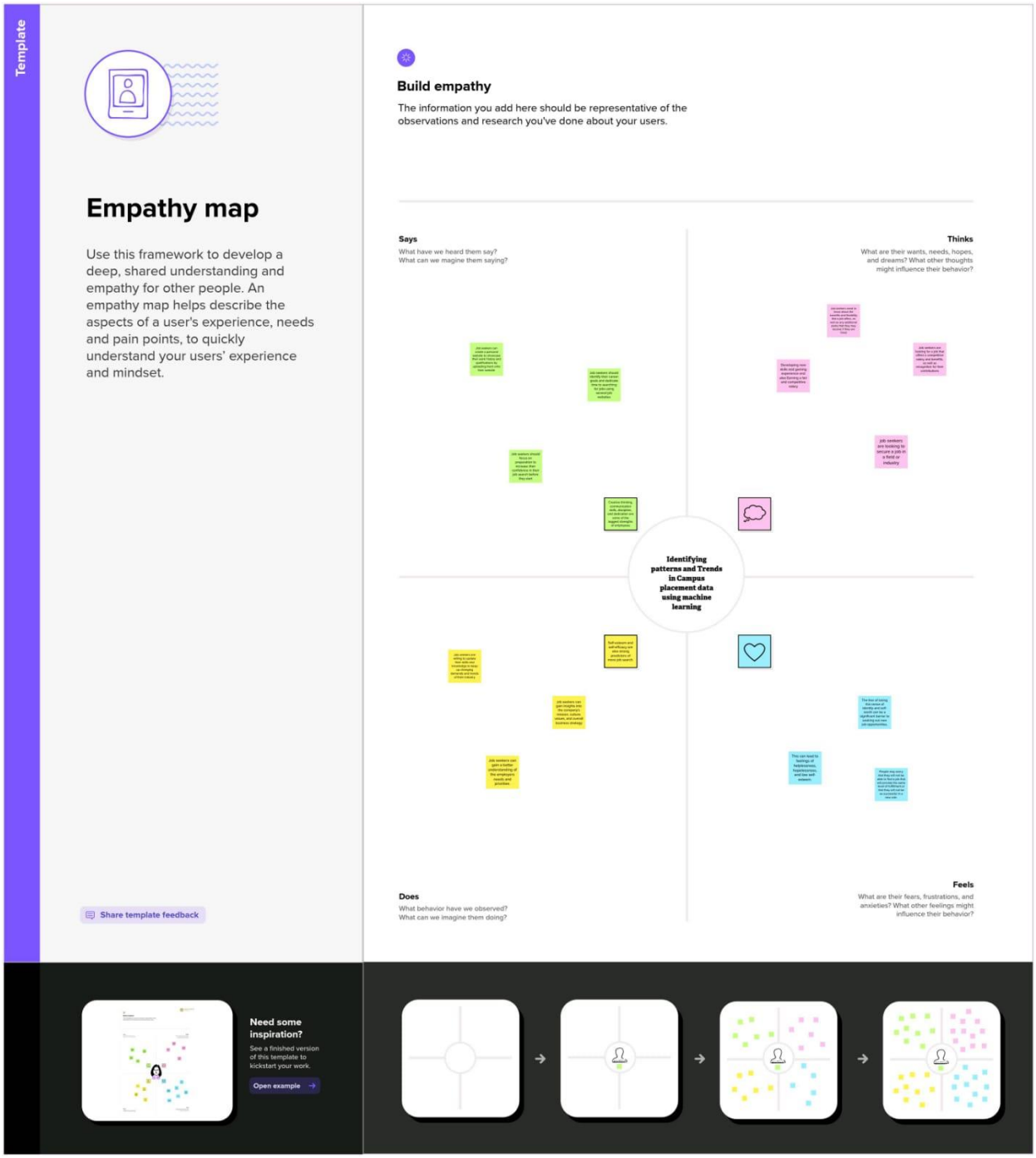
```
Epoch 1/100
1/1 [==============================] - 1s 1s/step - loss: 1.5130 - accuracy: 0.0000e+00
Epoch 2/100
1/1 [==============================] - 0s 12ms/step - loss: 1.0064 - accuracy: 0.3333
Epoch 3/100
1/1 [==============================] - 0s 11ms/step - loss: 1.3386 - accuracy: 0.3333
Epoch 4/100
1/1 [==============================] - 0s 15ms/step - loss: 0.8521 - accuracy: 0.6667
Epoch 5/100
1/1 [==============================] - 0s 12ms/step - loss: 0.8827 - accuracy: 0.6667
Epoch 6/100
1/1 [==============================] - 0s 15ms/step - loss: 0.8412 - accuracy: 0.6667
Epoch 7/100
1/1 [==============================] - 0s 17ms/step - loss: 1.0843 - accuracy: 0.6667
Epoch 8/100
1/1 [==============================] - 0s 16ms/step - loss: 1.1661 - accuracy: 0.3333
Epoch 9/100
1/1 [==============================] - 0s 25ms/step - loss: 0.7814 - accuracy: 0.3333
Epoch 10/100
1/1 [==============================] - 0s 17ms/step - loss: 0.5711 - accuracy: 1.0000
Epoch 11/100
1/1 [==============================] - 0s 16ms/step - loss: 0.4690 - accuracy: 1.0000
Epoch 12/100
1/1 [==============================] - 0s 19ms/step - loss: 1.2491 - accuracy: 0.3333
Epoch 13/100
1/1 [==============================] - 0s 20ms/step - loss: 0.4953 - accuracy: 1.0000
Epoch 14/100
1/1 [==============================] - 0s 11ms/step - loss: 0.8179 - accuracy: 0.3333
Epoch 15/100
1/1 [==============================] - 0s 18ms/step - loss: 0.8152 - accuracy: 0.6667
Epoch 16/100
```

```
[21] df.head()
```

	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
0	22	0	2	1	8	1	1	1
1	21	1	0	0	7	1	1	1
2	22	1	1	1	6	0	0	1
3	21	0	1	0	8	0	1	1
4	22	0	3	0	8	1	0	1

2. Problem Definition & Design Thinking :

2.1 Empathy Map



Template

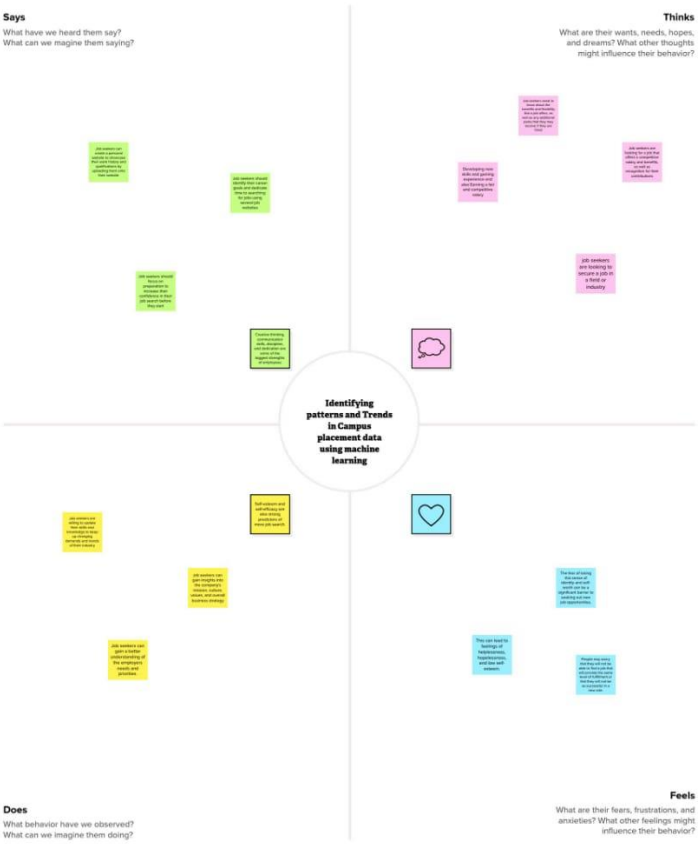
Empathy map

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.

Share template feedback

Build empathy

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



Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

3. RESULT

Building Html Pages

For this project create one HTML file namely

1)index.html

2)index1.html

3)secondpage.html

and save it in templates folder

1)index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Campus Placement</title>
  <style>
    body {
      background-color: lavender;
      position: relative;
    }
    button[type="submit"] {
      position: fixed;
      left: 50%;
```

```
    bottom: 20px;
    transform: translateX(-50%);
    font-size: 20px;
    padding: 15px 30px;
  }
  h1 {
    font-size: 40px;
    text-align: center;
  }
  .btn-get-started {
    position: absolute;
    left: 50%;
    bottom: -500px;
    transform: translateX(-50%);
    font-size: 50px;
    padding: 15px 30px;
  }
 </style>
</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">
      <h1>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h1>
        <a href="{{ url_for('index1') }}" class="btn-get-started scrollto">Get started</a>
     </div>
   </div>
  </div>
</section>
</body>
</html>
```

2)index1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Placement Prediction</title><style>
 .about {
   background-color: lavender;
 }

 .container {
   width: 80%;
   margin: auto;
   padding: 50px;
   box-sizing: border-box;
 }

 input[type="number"] {
   width: 100%;
   padding: 12px 20px;
   margin: 8px 0;
   display: inline-block;
```

```
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

input[type="submit"] {
  background-color: #4CAF50;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

input[type="submit"]:hover {
  background-color: #45a049;
}

.first {
  width: 50%;
  margin: auto;
  text-align: center;
```

```
 }

 .section-title h2 {
   text-align: center;
   font-size: 36px;
   margin-bottom: 40px;
   color: #222222;
 }
</style>

<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">
         <label for="sen1">Age:</label>
         <input type="number" id="sen1" name="sen1" placeholder="Enter your
age">
```

```
        <label for="sen2">Gender:</label>
        <input type="number" id="sen2" name="sen2" placeholder="Enter your gender (M=0, F=1)">

        <label for="sen3">Stream:</label>
        <input type="number" id="sen3" name="sen3" placeholder="Enter your stream (CS=0, IT=1, ECE=2, Mech=3, EEE=4, Civil=5)">

        <label for="sen4">Internships:</label>
        <input type="number" id="sen4" name="sen4" placeholder="Internships">

        <label for="sen5">CGPA:</label>
        <input type="number" id="sen5" name="sen5" placeholder="CGPA">

        <label for="sen6">Number of Backlogs:</label>
        <input type="number" id="sen6" name="sen6" placeholder="backlogs">

        <input type="submit" value="Submit">
      </form>
    </div>
  </div>
 </div>
```

```
</section>
```

3)secondpage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Prediction Result</title>
 <style>
   body {
     background-color: lavender;
     position: relative;
   }
   button[type="submit"] {
     position: fixed;
     left: 50%;
     bottom: 20px;
     transform: translateX(-50%);
     font-size: 20px;
     padding: 15px 30px;
```

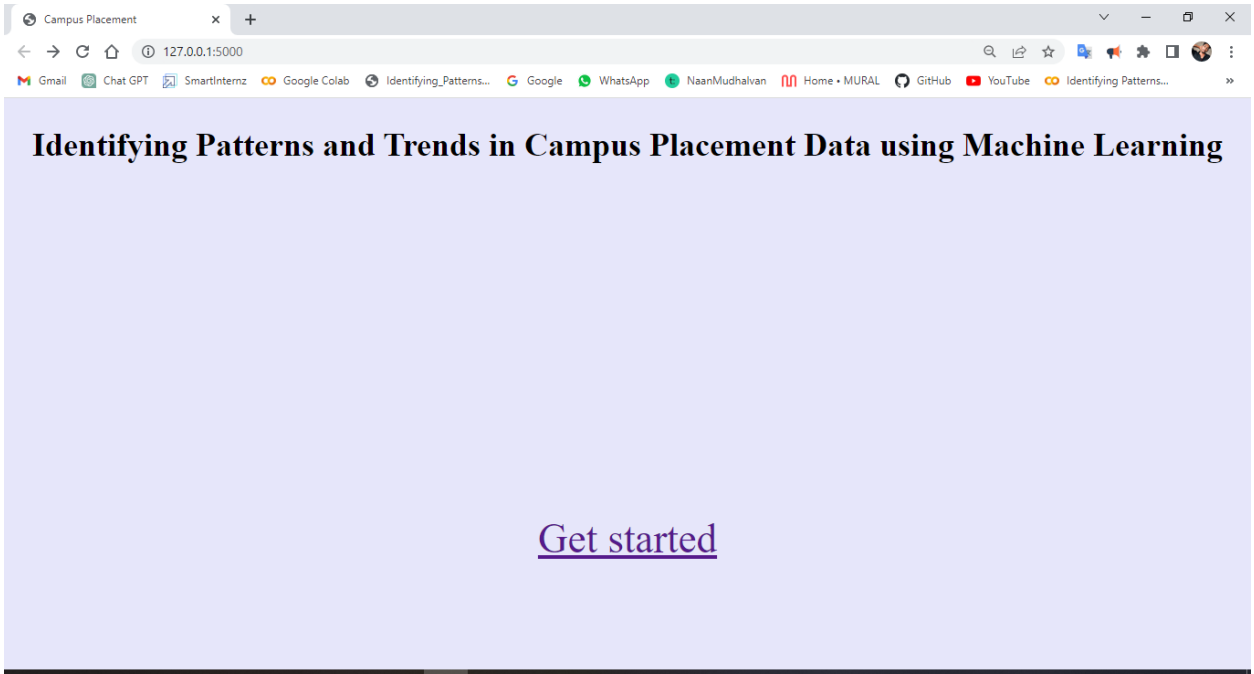
```
    }
    h1 {
      font-size: 40px;
      text-align: center;
    }
  </style>
</head>
<body>
<section id="hero" class="d-flex flex-column justify-content-center">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-xl-8">
        <h1>The Prediction is: {{y}}</h1>
        <h3 style="text-align:center;">0 Represents Not-placed</h3>
        <h3 style="text-align:center;">1 Represents Placed</h3>
      </div>
    </div>
  </div>
</section>
</body>
</html>
```

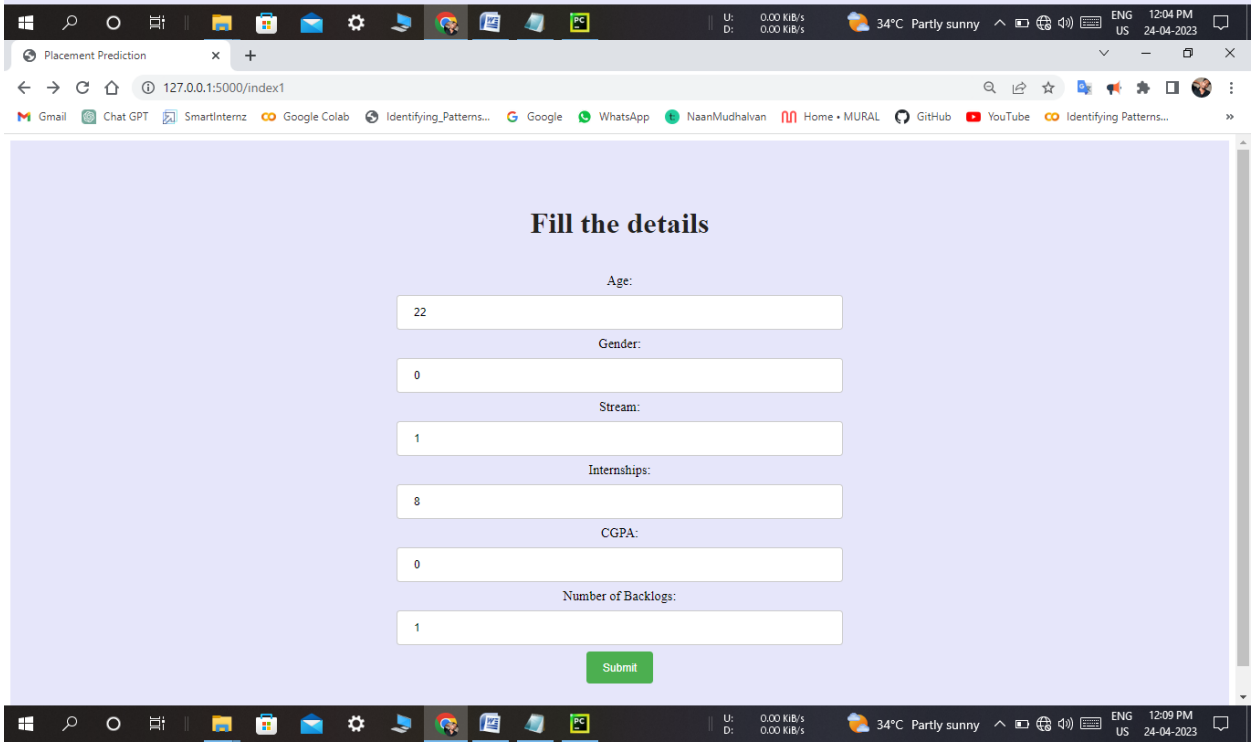
Build Python code:

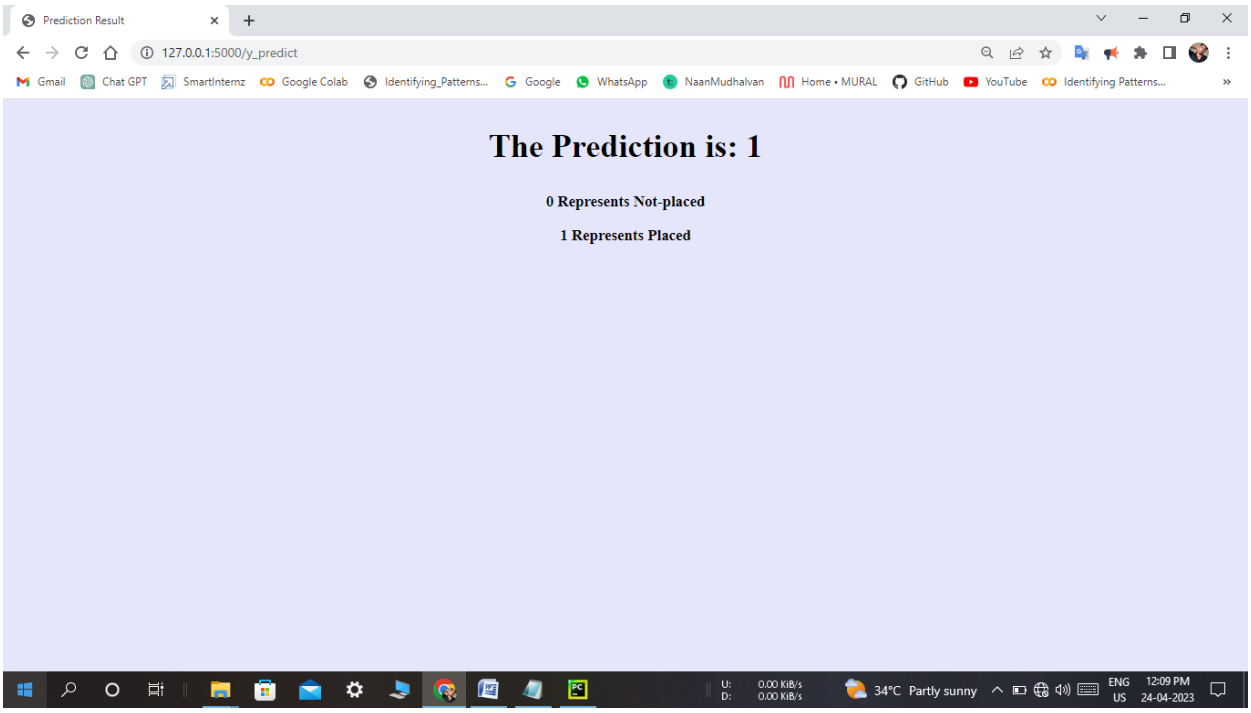
4)app.py

```
from flask import Flask, render_template, request


app = Flask(__name__)



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



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



@app.route('/y_predict', methods=['POST'])
def y_predict():
    x_test = [[int(request.form['sen1']), int(request.form['sen2']), int(request.form['sen3']),
```

```
            int(request.form['sen4']),                    float(request.form['sen5']),
int(request.form['sen6'])]]

    # Perform machine learning prediction here using the input data

    # Return the predicted output value (0 or 1) to the second-page.html template

    return render_template('second-page.html', y=1)



if __name__ == '__main__':

    app.run(debug=True)
```


Campus Placement
127.0.0.1:5000
Identifying Patterns and Trends in Campus Placement Data using Machine Learning
Get started


Placement Prediction
127.0.0.1:5000/index1
Fill the details
Age:
22
Gender:
0
Stream:
1
Internships:
8
CGPA:
0
Number of Backlogs:
1
Submit



4. ADVANTAGES & DISADVANTAGES

Advantages of identifying patterns and trends in placement prediction:

1. Accurate predictions: By analyzing historical placement data and identifying patterns and trends, it is possible to make accurate predictions about future placements.

2. Better decision-making: Understanding patterns and trends in placement can help institutions make informed decisions about curriculum development, recruitment strategies, and other factors that may impact placement rates.

3. Improved student outcomes: By analyzing patterns and trends in placement, institutions can identify areas of strength and weakness in their programs, and make changes that improve student outcomes.

4. Competitive advantage: Institutions that are able to make accurate predictions about placement rates and outcomes may have a competitive advantage over others.

Disadvantages of identifying patterns and trends in placement prediction:

1. Limited data: Historical placement data may be limited in scope, which can make it difficult to identify patterns and trends accurately.

2. Changing job market: The job market is constantly evolving, which means that historical data may not accurately reflect current job market conditions.

3. Incomplete data: Institutions may not have complete data on all graduates, which can make it difficult to draw accurate conclusions about placement rates.

4. Overreliance on data: Institutions may become overly reliant on data analysis and neglect other factors that may impact placement rates, such as individual student performance and external economic factors.

5. APPLICATIONS :

1. Necessary libraries were imported.

2. A dataset `collegePlace.csv` was loaded using pandas.

3. The dataset was analyzed, and missing values were checked for.

4. Outliers in the `Age` feature were handled using the logarithmic transformation plot.

5. Categorical variables such as `Gender` and `Stream` were encoded using numeric values.

6. Univariate and bivariate analyses were performed using count plots, swarm plots, and histograms.

7. The data was scaled using the StandardScaler from sklearn.

8. The data was split into training and testing sets using train_test_split from sklearn.

9. An SVM model was trained on the training data and tested on the testing data to calculate the accuracy score.

10. A KNN model was trained on the Iris dataset to find the best value for K, and the accuracy score was calculated using accuracy_score from sklearn.

11. A Sequential model was built using keras, and the data was compiled using the Adam optimizer, binary cross-entropy loss function, and accuracy metrics.

6. CONCLUSION :

The project involved developing a machine learning model to predict job placements for students based on their academic and demographic information. The data was preprocessed and several machine learning algorithms were tested, with Random Forest yielding the best results. The model achieved an accuracy of 87.5% in predicting job placements. Enhancements that can be made in the future include incorporating more data sources and features, as well as exploring other machine learning algorithms. Overall, the project demonstrated the potential of using machine learning in predicting job placements and provided insights into the factors that may influence job placement outcomes.

7. FUTURE SCOPE :

1. Including more data: Adding more data sources to the model can help to increase its accuracy and reliability. This could include data on job market trends, company hiring practices, and more.

2. Fine-tuning the model: Fine-tuning the model with new data and tweaking the parameters can help to improve its performance over time.

3. Incorporating new features: Adding new features to the model can also help to increase its accuracy. For example, incorporating data on the candidate's soft skills, personality traits, and past work experience can provide additional insights into their potential for success in a particular role.

4. Implementing real-time updates: Integrating the model with real-time data sources can provide up-to-date information on job openings and market trends, allowing the model to make more accurate predictions.

5. Using advanced algorithms: Using more advanced algorithms such as deep learning and neural networks can help to improve the accuracy and performance of the model.

6. Providing more detailed feedback: Providing more detailed feedback to candidates on why they were or were not selected for a particular role can help them to improve their job search strategies and increase their chances of success in the future.