Identifying Patterns And Trends In Campus Placement Data Using Machine Learning

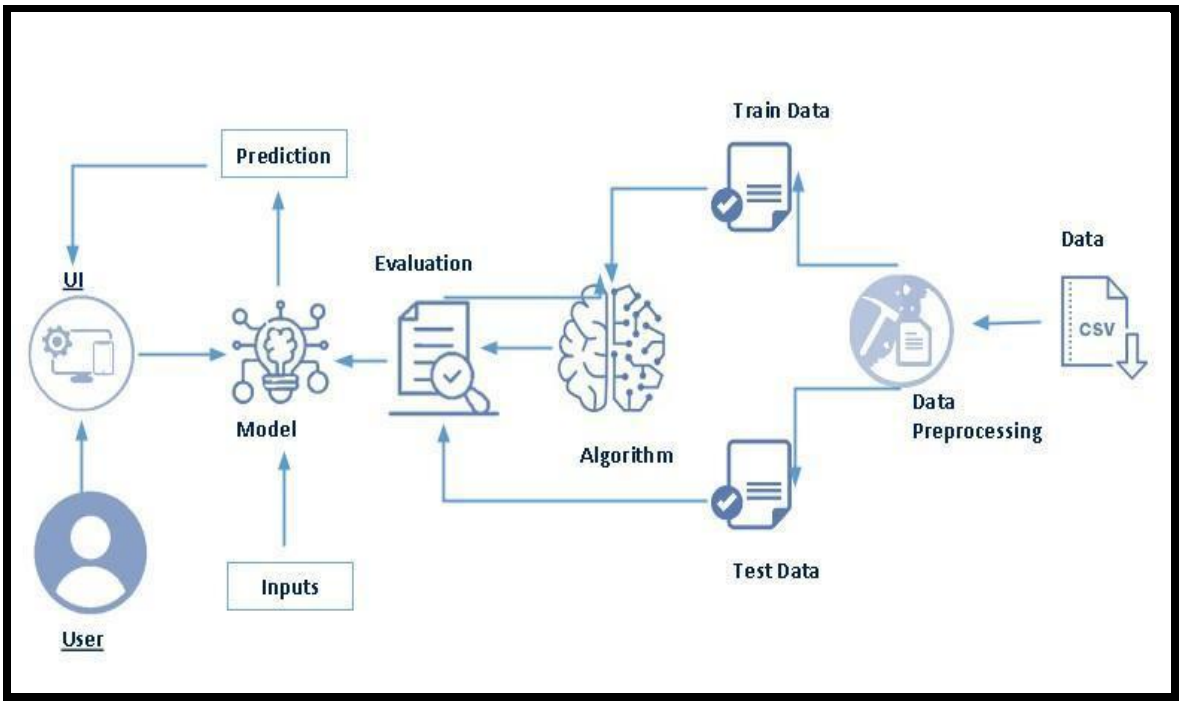
ABSTRACT:

●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:

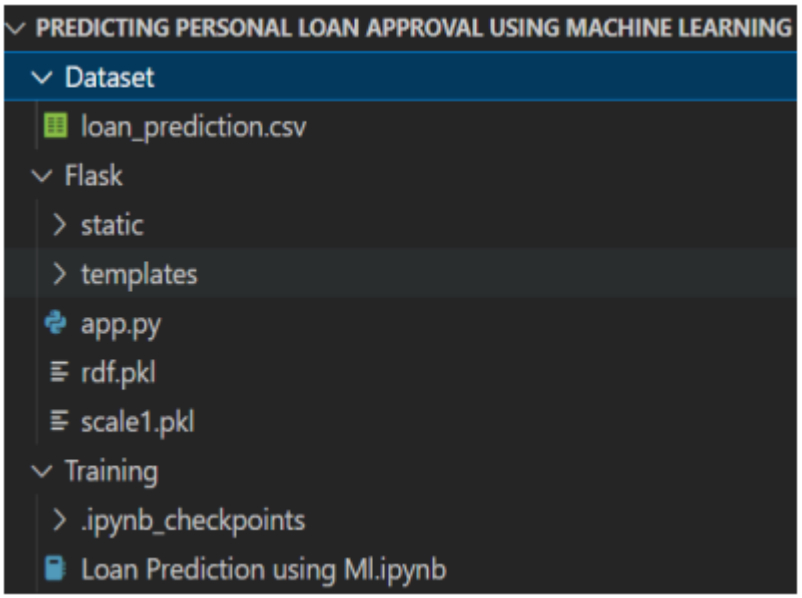
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyzes the input the prediction is showcased on the UI

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

- Data collection:
  - Collect the dataset or create the data
- Visualizing and analyzing data:
  - UniVariate analysis
  - Bivariate analysis
  - Multivariate analysis
  - Descriptive analysis
- Data pre-processing:
  - Checking for null values
  - Handling outlier
  - Handling categorical data
  - Splitting data into train and test
- Model building:
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model
- Application Building:
  - Create an HTML file
  - Build python code

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 1:

Define Problem / Problem Understanding:

In this milestone, we will see the define problem and problem understanding.

Specify the Business Problem:

- 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.

Business Requirements:

The business requirements for a project aimed at "Identifying Patterns and Trends in Campus Placement Data using Machine Learning" would likely include the following:

- Access to campus placement data: The project would require access to data on student performance, qualifications, and job placement outcomes. This data would need to be collected, cleaned, and prepared for analysis.

- Machine learning expertise: The project would require individuals with expertise in machine learning, data science and statistical analysis to develop and implement the algorithms and models needed to analyze the data.

- Data storage and management: The project would require a robust and secure data storage and management system to store and organize the large amounts of data used in the analysis.

- Infrastructure for model deployment: The project would require infrastructure for deploying the models and algorithms developed, including hardware, software, and cloud-based resources.

Literature Survey:

- There have been several studies that have used machine learning techniques to identify patterns and trends in campus placement data.
- One study by authors P. K. Rajesh and Dr. G. R. Suresh, published in the

International Journal of Computer Science and Mobile Computing in 2015, used k-means clustering and decision trees to analyze campus placement data and identify patterns that could be used to predict placement outcomes

- Another study by authors V.V. Kulkarni and K.S. Patil, published in the International Journal of Engineering Research and Technology in 2012, used decision tree and neural network algorithms to analyze campus placement data and identify factors that influence student placement.
- A study by authors S.S. Bhosale, S.S. Raut, and D.S. Kulkarni, published in the International Journal of Emerging Research in Management & Technology in 2013, used decision tree and Naive Bayes algorithms to analyze campus placement data and predict student placement outcomes. A study by authors S.S.bhosale, S.S. Raut, and D.S. Kulkarni, published in the International Journal of Emerging Research in Management &

Technology in 2013, used decision tree and Naive Bayes algorithms to analyze campus placement data and predict student placement outcomes.

- In general, these studies found that machine learning techniques were effective at identifying patterns and trends in campus placement data, and could be used to predict student placement outcomes with high accuracy.
- It's important to note that all these studies are quite old now and you might find more recent studies and new techniques which can be useful for your project.

Social or Business Impact:

The business impact of a project that uses machine learning to identify patterns and trends in campus placement data could be significant. By analyzing data on factors such as student performance, qualifications, and job

placement outcomes, the project could help organizations make more informed decisions about recruiting and hiring new graduates.

Empathy map:

Template

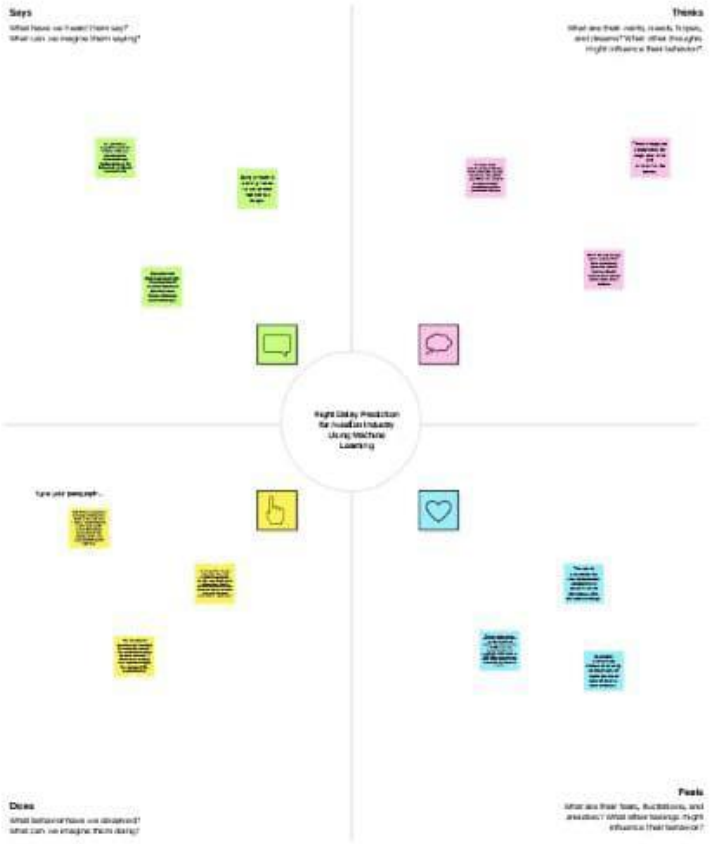
Empathy map

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

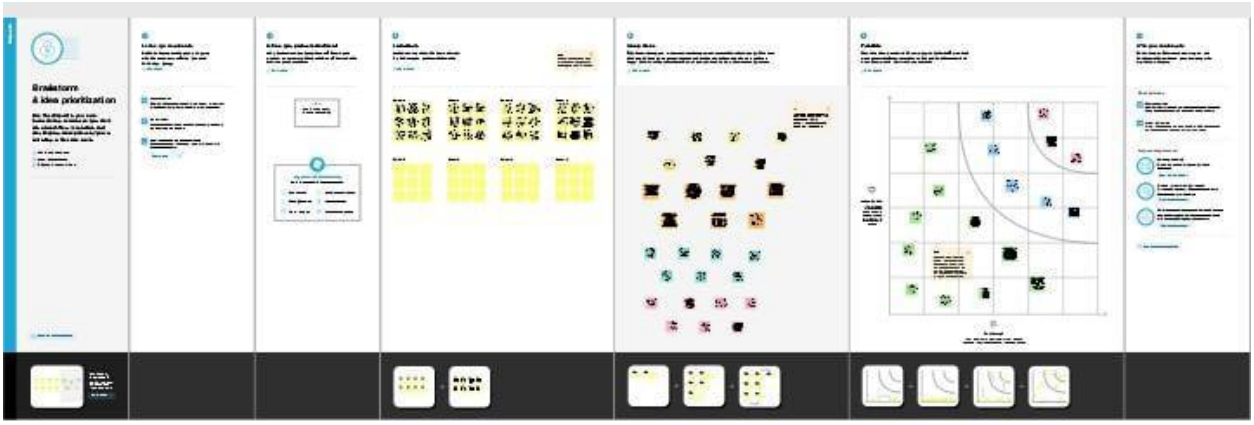
The information you add here should be representative of the observations and research you've done about your users.



Need some inspiration?

Open example

Brainstorm map:



Advantage:

Using pattern recognition techniques provides a large number of benefits to an individual. It not only helps in the analysis of trends but also helps in making predictions.

- It helps in the **identification** of objects at varying distances and angles.
- Easy and highly **automated.**
- It is not rocket science and does not require an out of the box thinking ability.

- Highly useful in the finance industry to make **valuable predictions** regarding sales.
- Efficient solutions to **real-time problems**.
- Useful in the medical fields for **forensic analysis** and DNA (Deoxyribonucleic acid) sequencing.

Identifying patterns and trends in campus placement data using machine learning can offer several advantages for educational institutions, students, and recruiters. Some of the key advantages include:

Improved Decision Making:

Machine learning algorithms can analyze large volumes of campus placement data to identify patterns and trends, enabling educational institutions to make informed decisions. For example, institutions can identify the most in-demand skills, industries, or job roles based on historical data, and tailor their

curriculum and placement strategies accordingly. Students can also make more informed decisions about their career choices based on data-driven insights.

Enhanced Placement Success:

Analyzing campus placement data using machine learning can help improve placement success rates. By identifying patterns and trends in successful placements, educational institutions can optimize their placement strategies, such as targeting specific companies or industries, providing additional training in areas with high demand, or offering personalized career counseling. This can result in higher job placement rates for students, improving their career prospects.

Efficient Resource Allocation:

Machine learning can help optimize resource allocation in campus

placements. Institutions can identify which companies or industries have a higher likelihood of hiring their graduates based on historical data, and allocate their resources, such as time and efforts, accordingly. This can help institutions streamline their placement processes and maximize the effectiveness of their placement efforts, saving time and resources.

Personalized Career Guidance:

Machine learning algorithms can provide personalized career guidance to students based on their skills, interests, and historical placement data. This can help students make informed decisions about their career paths, identify skill gaps, and take proactive steps to improve their employability. Personalized career guidance can also help students explore and consider career options they may not have been aware of, based on data-driven insights.

Better Employer-Student Matching:

Machine language analyze campus placement data to match students with employers more effectively. By considering factors such as students' skills, preferences, and historical data, machine learning algorithms can recommend suitable job opportunities to students and help them find the best fit. This can result in higher job satisfaction for students and employers, leading to more successful and long-term job placements.

Disadvantages:

Biased Data:

Machine learning models are only as good as the data they are trained on. If the campus placement data used to train the model is biased or incomplete, it can lead to biased results. For example, if the data used for training is collected only from

a particular group of students or from a specific time period, the model may not accurately reflect the diversity and dynamics of the entire campus population. This can result in biased predictions and recommendations, leading to unfair or discriminatory outcomes.

Overreliance on Historical Data:

Machine learning models rely on historical data to identify patterns and trends. However, campus placement data can change over time due to various factors such as changes in the job market, economic conditions, and student preferences. If the model is solely based on historical data, it may not accurately predict future placement trends. It may fail to capture emerging patterns or shifts in the job market, leading to inaccurate or outdated predictions.

Lack of Contextual Understanding:

Machine learning models may not have the ability to fully understand the context and nuances of campus placement data. For example, they may not consider factors such as student preferences, interests, and career goals that are subjective and difficult to quantify. This can result in recommendations that do not align with the individual needs and aspirations of students, leading to suboptimal outcomes.

Limited Interpretability:

Some machine learning models, such as deep learning models, are often considered "black boxes" as they are complex and difficult to interpret. This lack of interpretability can make it challenging to understand how the model is making predictions and identify any biases or errors in the process. It can also make it difficult to explain the results to stakeholders, such as

students, faculty, and recruiters, which may impact their trust and acceptance of the model's predictions.

Ethical Concerns:

The use of machine learning in campus placement data analysis raises ethical concerns related to privacy, security, and fairness. For example, the use of personal data, such as student demographics, academic performance, and employment history, may raise privacy concerns and require proper consent and data protection measures. Additionally, the use of machine learning to make decisions about students' careers and opportunities may raise questions about fairness, accountability, and transparency, particularly if biases are present in the data or the model.

Human Bias in Model Development:

Machine language has developed by human programmers who make decisions about data selection, feature engineering, and model design, which can introduce human biases into the model. Biases in model development can result in biased predictions or reinforce existing biases in the campus placement process. It is important to be mindful of potential biases during model development and take steps to mitigate them.

Resource Requirements:

Developing and maintaining machine learning models requires significant resources, including computing power, data storage, and technical expertise. Smaller educational institutions or organizations with limited resources may face challenges in implementing and maintaining machine learning models for campus analysis.

Application:

Trend Analysis:

Pattern recognition helps in identifying the trend in the given data on which appropriate analysis can be done. For example, looking at the recent trends in the sales made by a particular company or organization, future sales can be predicted.

Assistance:

Pattern is an integral part of our daily lives. It provides immense help in our day to day activities. A large number of software and applications are there in the market today that use machine learning algorithms to make predictions regarding the presence of obstacles and alerts the user to void miss happenings.

E-Commerce:

Visual search engines recognize the desired item based on its specifications and provide appropriate results. Most of the websites dedicated to online shopping make

use of recommender systems. These systems collect data related to each customer purchase and make suggestions. All these tasks are accomplished by analyzing previous trends to make successful predictions.

Computer vision:

The user interacts with the system by giving an image or video as the input. The machine compares it with thousands or maybe millions of images stored in its database, to find similar patterns. The drawl of the essential features is done by using an algorithm that is mainly directed for grouping similar looking objects and patterns. This is termed as computer vision. Example, cancer detection.

Biometric devices:

These devices secure authentication and security by making using of face recognition and fingerprint detection technologies. On the hidden side, the base that enables the use of

technologies like face and fingerprint recognition is machine learning algorithms.

Predicting job demand:

By analyzing past campus placement data, machine learning algorithms can identify patterns and trends in the job market, such as the industries that are hiring the most, the types of roles that are in demand, and the skillsets that are highly sought after. This information can be used by educational institutions to guide students in choosing their career paths and by recruiters to plan their hiring strategies.

Matching students with job opportunities:

Machine learning algorithms can analyze campus placement data to match students' skills, qualifications, and preferences with job opportunities. By identifying patterns and trends in the data, such as the qualifications and skills of students who were

successfully placed in particular jobs, the algorithms can suggest suitable job opportunities to students and help them make informed career choices.

Improving placement success rate:

Campus placement data can be used to build machine learning models that can predict the likelihood of a student getting placed in a particular job based on their academic performance, skills, and other relevant factors. By identifying patterns and trends in the data, these models can help institutions and students understand the factors that influence placement success and take appropriate actions to improve their chances of getting placed.

Enhancing employer engagement:

Machine learning can be used to analyze campus placement data to identify the preferences and requirements of employers,

such as the types of candidates they tend to hire, the skills they prioritize, and the compensation packages they offer. This information can be used by educational institutions to engage with employers more effectively and tailor their curriculum and training programs to meet industry demands, thus improving the chances of successful placements for their students.

Identifying skill gaps:

Analyzing campus placement data can help identify skill gaps in the job market. By analyzing the skills of students who were not successfully placed, machine learning algorithms can identify the skills that are in demand but lacking in the student population. This information can be used by educational institutions to revise their curriculum and training programs to bridge the skill gaps and prepare students for the job market more effectively.

Monitoring alumni success:

Campus placement data can be used to track the career progression of alumni and identify patterns and trends in their career paths. Machine learning algorithms can analyze data such as job changes, promotions, and salary growth of alumni to identify factors that contribute to their success. This information can be used by educational institutions to improve their curriculum, career counseling services, and alumni engagement programs.

MILESTONE 2:

Data Collection & Preparation:

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.

Task 1:

Problem understanding

1) **Specify business problem**
2) **Business Requirement**
3) **Literature survey**
4) **Social/Business impact**

Task 2:

Data Understanding

1) Data collection
2) Loading Data

Task 3:

EDA

1) Data Cleaning
2) Data Manipulation
3) Visualization

Task 4:

Model building

Task 5:

Testing the model

Task 6:

Deployment

Task 7:

Doc

Importing the Libraries:

```
# Importing required lib

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

```
# Checking for available styles

plt.style.available
```

```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
```

'seaborn-v0_8-muted',
'seaborn-v0_8-notebook',
'seaborn-v0_8-paper',
'seaborn-v0_8-pastel',
'seaborn-v0_8-poster',
'seaborn-v0_8-talk',
'seaborn-v0_8-ticks',
'seaborn-v0_8-white',
'seaborn-v0_8-whitegrid',
'tableau-colorblind10']

Read The Dataset :

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

#Reading csv data

df=pd.read_csv('/content/collegePlace.csv')

df.head()

```
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

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

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().sum()
```

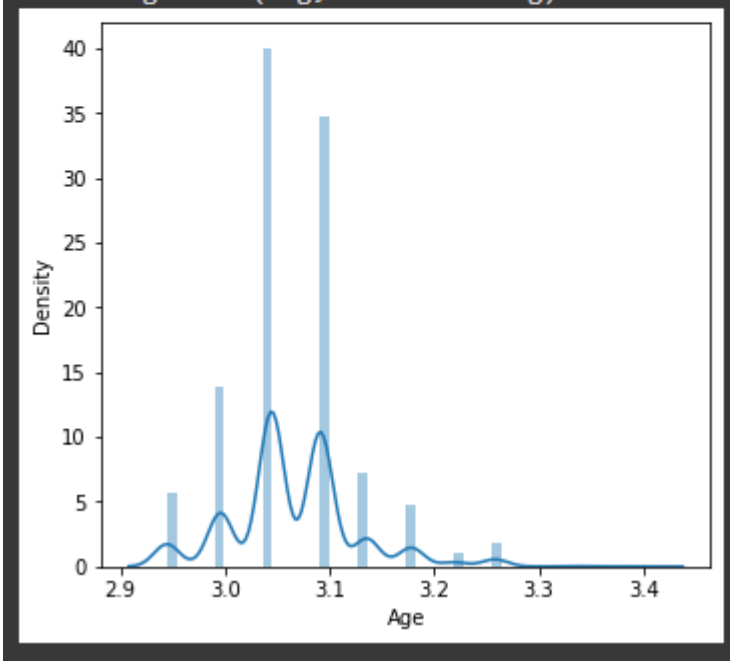
```
Age                 0
Gender              0
Stream              0
Internships         0
CGPA                0
Hostel              0
HistoryOfBacklogs   0
PlacedOrNot         0
dtype: int64
```

Handling Outliers :

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

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

```
/usr/local/lib/python3.8/dist-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)
```



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 = 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 = df.drop(['Hostel'], axis=1)
```

```
df
```

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

2966 rows × 7 columns

MILESTONE 3:

Exploratory Data Analysis:

In this milestone, we will see the exploratory data analysis.

Visual Analysis:

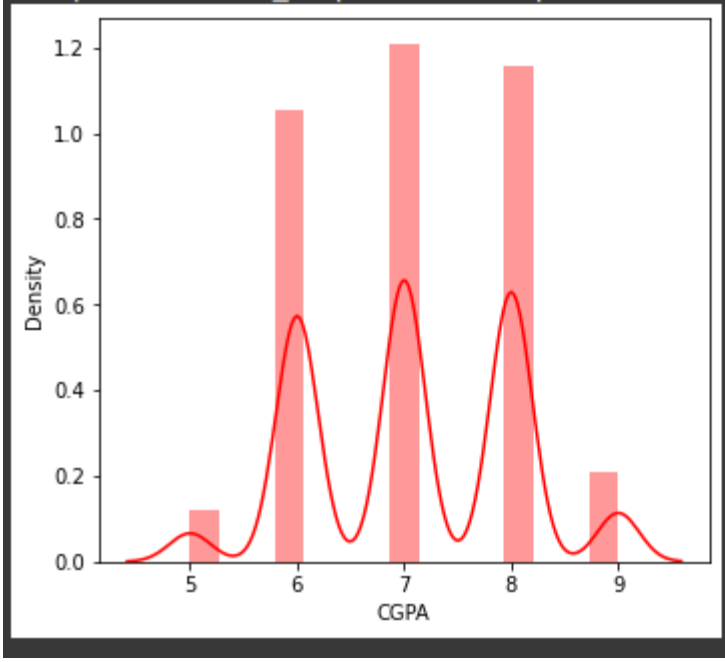
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate Analysis:

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

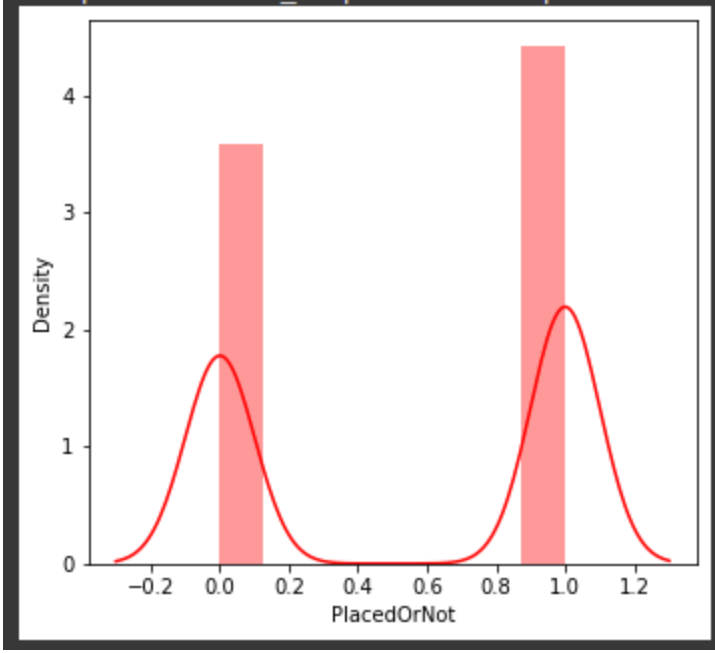
```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['CGPA'],color='r')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:261
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f5463e50d00>
```



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

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: Futu
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f5463d95790>
```

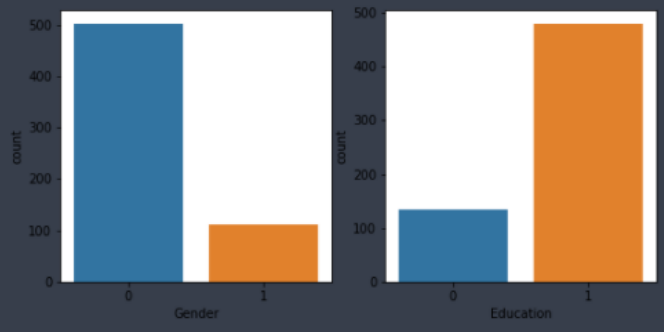


Bivariate Analysis :

Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```
#plotting the count plot
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(data['Gender'])
plt.subplot(1,4,2)
sns.countplot(data['Education'])
plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
```

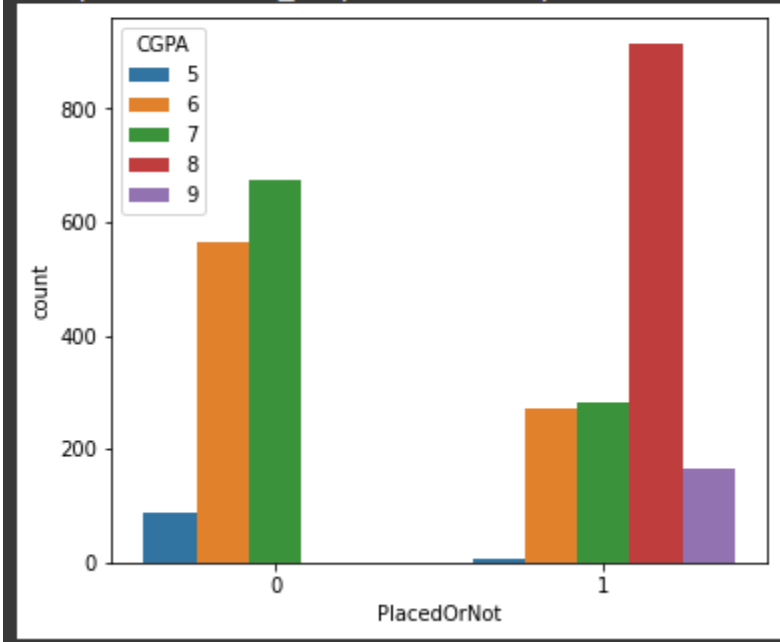


Multivariate Analysis :

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

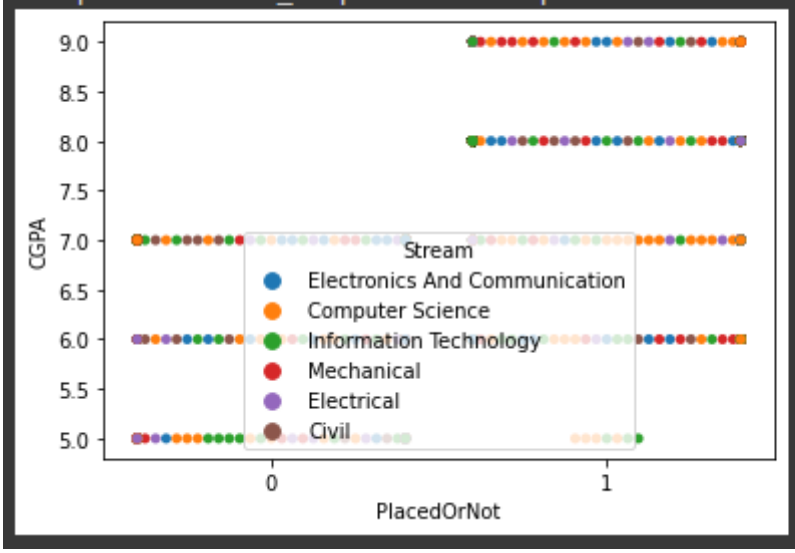
```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(df["PlacedOrNot"],hue=df['CGPA'])
```

```
/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>
```



Scaling The Data :

Scaling is one the important processes we have to perform on the dataset, because data measures in different ranges can lead to mislead in prediction Models such as KNN, Logistic regression needs scaled data, as they

follow distance based method and Gradient Descent concept.

```
# perfroming feature Scaling operation using standard scaller on X part of the dataset becau
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
```

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

We will perform scaling only on the input values.Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

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 = standardized_data
Y = df['PlacedOrNot']

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

MILESTONE 4:

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

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.

SVM Model :

A function named Support vector machine is created and train and test data are passed as the parameters. Inside the function, SVM Classifier 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
```

KNN Model :

A function named KNN is created and train and test data are passed as the parameters. Inside the function, K Neighbors Classifier 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}
```

Artificial Neural Network Model :

We will also be using a neural network to train the model.

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

```
[ ] classifier = Sequential()

    #add input layer and first hidden layer
    classifier.add(keras.layers.Dense(6,activation = 'relu', input_dim = 6))
    classifier.add(keras.layers.Dropout(0.50))
    #add 2nd hidden layer
    classifier.add(keras.layers.Dense(6,activation = 'relu'))
    classifier.add(keras.layers.Dropout(0.50))

    #final or output layer
    classifier.add(keras.layers.Dense(1, activation = 'sigmoid'))
```

```
[ ] #Compiling the model

    loss_1 = tf. keras.losses.BinaryCrossentropy()

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

```
[ ] #fitting the model
    classifier.fit(X_train, Y_train, batch_size = 20, epochs = 100)
```

MILESTONE 5:

Model Deployment :

In this milestone, we will see the model deployment.

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'))
```

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

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

```
<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 -->
```

Building Html Pages(Part-2) :

- The below code is written to fetch the details from the user using <form> tag. A submit button is provided at the end which navigates to the prediction page upon clicking.

```
<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
            <input type="number" id="sen4" name="sen4" placeholder="Internships">
            <input type="number" id="sen5" name="sen5" placeholder="CGPA">
            <input type="number" id="sen6" name="sen6" placeholder="Number of backlogs">
            <input type="submit" value="Submit">

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

  </div>
</section><!-- End About Us Section -->
```

- The code for secondpage.html is as shown below. The code includes a section that provides the output on screen.

```
<section id="hero" class="d-flex flex-column justify-content-center">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-xl-8">
        <h1>The Prediction is : {{y}}</h1>
        <h3> 0 represents Not-Placed </h3>
        <h3> 1 represents Placed<h2>


      </div>
    </div>
  </div>
</section><!-- End Hero -->
```

Build Python Code :

In this activity , we will go through the building of the python code.

Import the Libraries :

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

```
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 Page :

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

- Here we will be using a declared constructor to route to the HTML page which we have created earlier.

- In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

- This below code retrieves the value from UI

```
@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)
```

- Here in the above code we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model. predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function :

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

1)Open anaconda prompt from the start menu

2)Navigate to the folder where your python script is.

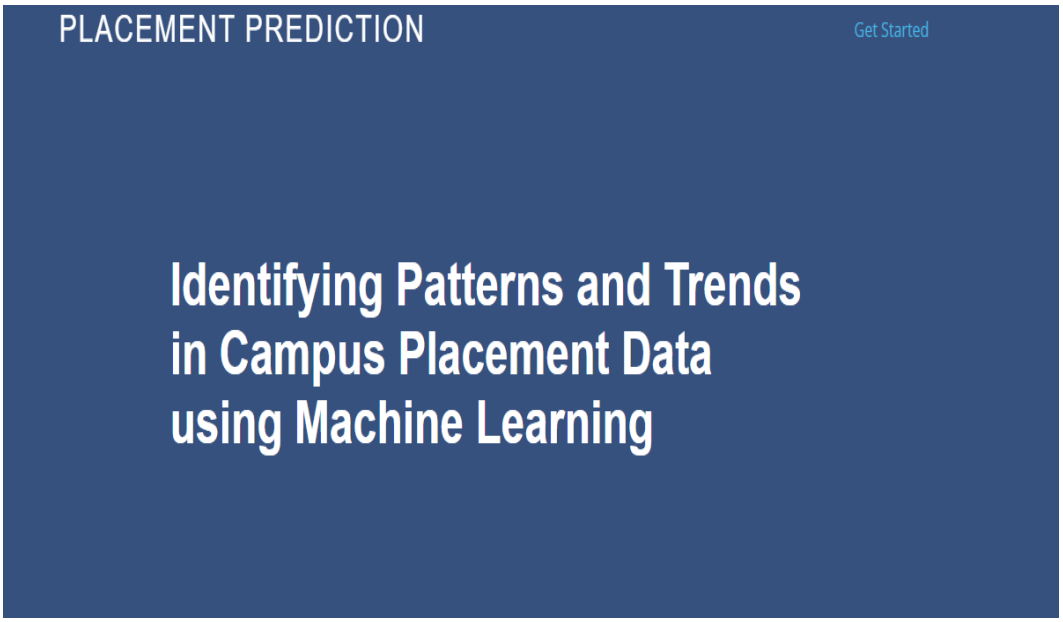
3)Now type “python app.py” command

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

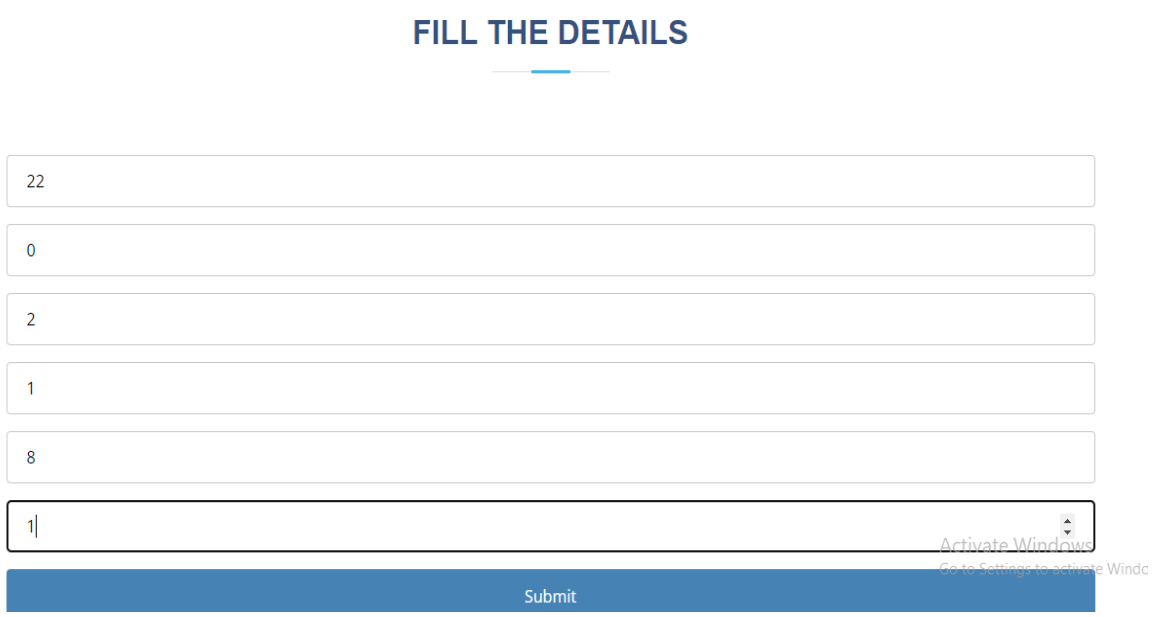
Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
* 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)
```

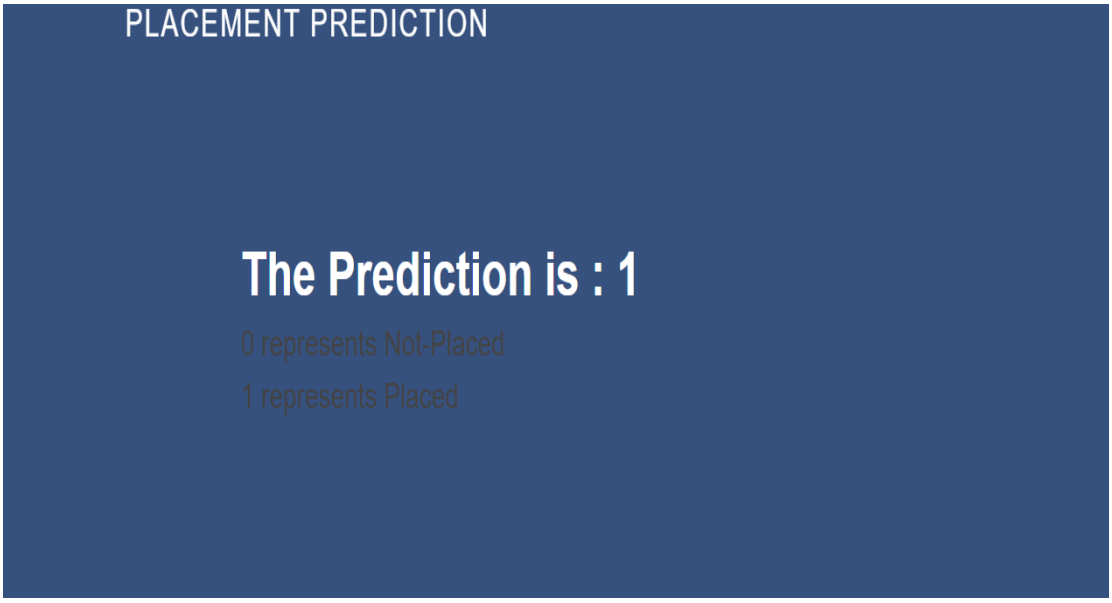
- Copy the link from the prompt and paste it in chrome. A web page opens up as described below
- Let’s see how our page looks like :



- 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:



MILESTONE 6:

Project Demonstration & Documentation :

Project Deliverables to be submitted along with other deliverables.

Record Explanation Video For Project End To End Solution :

Record explanation Video for project end to end solution.

Project Documentation-Step By Step Project Development Procedure :

Project Documentation-Step by step project development procedure.

RESULTS:

The result of identifying patterns and trends in campus placement data using machine learning can be the generation of insights and predictions that can inform decision-making related to campus placements. For example, a machine learning model trained on campus placement data can identify which majors, skills, or backgrounds are most in demand by employers, which companies are most likely to hire from a particular program, which geographic locations offer the most opportunities, and what salary ranges are typical for different roles.

These insights can help students make more informed decisions about their career paths, guide institutions in curriculum development and program design, and assist employers in their recruitment strategies. Machine learning models can also provide predictive analytics that forecast future trends and opportunities, enabling institutions and students to be proactive in their decision-making.

By leveraging machine learning to analyze campus placement data, institutions can gain a competitive advantage in preparing students for successful careers, creating more efficient and effective recruitment processes, and better understanding the needs and expectations of the job market. Ultimately, the result of identifying patterns and trends in campus placement data using machine learning can lead to more informed and strategic

decisions that benefit all stakeholders involved in the campus placement process.

Machine learning can be used to identify patterns and trends in campus placement data, which can help organizations make data-driven decisions about their hiring processes. Some of the common machine learning techniques used in this context include:

Regression analysis:

This can be used to identify relationships between various factors and the placement outcomes. For example, you could use regression analysis to identify which factors (such as academic performance, internships, or extracurricular activities) have the most impact on the salary or job offers received by students.

Classification algorithms:

These can be used to classify students into different categories based on their placement outcomes. For example, you could use a decision tree algorithm to identify which factors (such as degree type, GPA, or major) are most important in determining whether a student will receive a job offer or not.

Clustering analysis:

This can be used to group students based on their placement outcomes and other factors. For example, you could use clustering to identify groups of students who have similar placement outcomes and then identify the factors that differentiate these groups from each other.

Using machine learning to analyze campus placement data can yield several

insights and trends that can help universities, employers, and students make better decisions. Some of the potential findings may include:

Identifying factors that affect placement rates:

Machine learning algorithms can help identify the factors that have the most significant impact on placement rates, such as academic performance, extracurricular activities, internships, etc.

Predicting future placement rates:

By analyzing historical placement data, machine learning models can predict future placement rates with a high degree of accuracy, helping universities and students plan accordingly.

Identifying patterns in salary offers:

Machine learning can help identify patterns in salary offers, such as which industries or companies offer the highest

salaries, which degrees or majors are most likely to lead to higher salaries, etc.

Identifying skills in demand:

By analyzing job postings and placement data, machine learning can identify the skills that are most in demand among employers, allowing universities to tailor their curriculum to better prepare students for the job market.

Identifying trends in industry demand:

Machine learning can help identify emerging industries and trends in the job market, allowing universities and students to stay ahead of the curve.

Overall, by using machine learning to analyze campus placement data, organizations can gain valuable insights into the factors that drive successful placements and use this information to improve their hiring processes.

Overall, the insights gained from machine learning analysis of campus placement data can help universities and students make more informed decisions about academic programs, job searches, and career paths.

CONCLUSION:

In conclusion, while machine learning can provide valuable insights from campus placement data, there are potential disadvantages that need to be considered, including biased data, overreliance on historical data, lack of contextual understanding, limited interpretability, ethical concerns, human bias in model development, and resource requirements. It is important to carefully evaluate and mitigate these disadvantages when using machine learning for campus placement data analysis to ensure fair,

accurate, and responsible use of the technology.

In conclusion, identifying patterns and trends in campus placement data using machine learning can offer valuable insights and help inform decision-making. However, it is essential to be aware of the potential disadvantages and limitations of using machine learning in this context, such as biased data, overreliance on historical data, lack of contextual understanding, limited interpretability, ethical concerns, human bias in model development, and resource requirements.

To mitigate these disadvantages, it is important to carefully curate diverse and representative data, validate the model's accuracy and fairness, consider the context and subjective factors involved in campus placements, ensure transparency and explain ability of the model, adhere to ethical

guidelines and data protection regulations, be mindful of human biases during model development, and allocate appropriate resources for model implementation and maintenance.

Overall, machine learning can be a valuable tool for identifying patterns and trends in campus placement data, but it should be used with caution, critically evaluated, and complemented with human judgment and expertise to ensure responsible and fair use in the context of campus placements.

The algorithm of machine learning we have discussed are can used to find the trend of placement, which will be helpful for university to get more admission in future. We compared the algorithm and find out the accuracy by considering some of attributes off students. Here we used deep neural network classifier with the 1000, 2000, 5000 iteration with 71%, 77%, and 91% of accuracy.

Machine learning is one of the buzz words in the 21st century. It is highly in demand due to popular machine learning applications and advantages. It has revolutionized all the industries with its amazing capabilities. Machine learning has different fields and scopes some of which include pattern recognition, data mining, analysis, etc.

Pattern recognition in machine learning is widely used in almost every industry today be it technical or non-technical. It has helped in the analysis and visualization of various trends. It has not only increased the efficiency and ease of analysis and prediction making but has also increased the job opportunities in the field. Top-notch companies such as Microsoft, Google, Amazon are looking for individuals skilled in the art of pattern recognition and

data analysis for making useful predictions. Thus, we can conclude by saying that pattern recognition is one of the most advancing fields in machine learning.

OVERVIEW OF THE PROGRAM:

The program of identifying patterns and trends in campus placement data using machine learning involves using algorithms to analyze historical campus placement data to identify patterns and trends that can be used to make predictions about future placement outcomes.

The program typically involves the following steps:

Data collection:

Collecting campus placement data from previous years, including information on the companies that participated in the placement process, the job roles offered, the salaries offered, and the number of students who were placed.

Data preprocessing:

Cleaning and transforming the data to make it suitable for analysis. This may involve removing duplicates, filling missing values, and encoding categorical variables.

Data exploration and visualization:

Exploring the data using descriptive statistics and visualization techniques to identify patterns and trends in the data.

Feature selection:

Selecting relevant features that are likely to influence the placement outcomes, such as academic performance, work experience, and skills.

Model selection:

Choosing an appropriate machine learning algorithm, such as logistic regression, decision trees, or neural networks, based on the nature of the data and the problem being solved.

Model training:

Training the machine learning model using the selected algorithm and the preprocessed data.

Model evaluation:

Evaluating the performance of the trained model using appropriate metrics such as accuracy, precision, recall, and F1-score.

Prediction:

Using the trained model to make predictions about future placement outcomes based on new data.

Visualization and interpretation:

Visualizing the results and interpreting the findings to gain insights into the factors that influence campus placements and to identify opportunities for improvement.

Overall, the program of identifying patterns and trends in campus placement data using machine learning is a powerful tool for universities and colleges to improve their

placement processes and help students achieve successful career outcomes.

FEATURES SCOPE:

Identifying patterns and trends in campus placement data using machine learning can involve a range of features depending on the specific goals and context of the analysis. Some possible features that could be used to identify patterns and trends in campus placement data using machine learning include:

Job-specific data:

This includes data related to the specific jobs that were offered, such as job title, job description, salary, and benefits.

Candidate-specific data:

This includes data related to the candidates who were offered jobs, such as their educational qualifications, skill set, experience, and performance in interviews.

Company-specific data:

This includes data related to the companies that participated in campus placements, such as their size, industry, location, and reputation.

Time-based data:

This includes data related to the timeline of campus placements, such as the duration of the placement season, the number of companies that participated in each year, and the number of candidates who were placed each year.

Demographic data:

This includes data related to the demographic characteristics of the candidates and companies, such as age, gender, race, and ethnicity.

Social media data:

This includes data related to social media activity, such as job postings, company news, and candidate profiles.

Online assessments:

This includes data related to online assessments, such as aptitude tests, coding challenges, and personality tests.

Alumni data:

This includes data related to the post-placement performance of candidates, such as

their job satisfaction, career growth, and salary progression.

Using machine learning algorithms, these features can be analyzed to identify patterns and trends in campus placement data, such as which jobs are in high demand, which companies are most popular among candidates, which demographic groups are more successful in getting placements, and which skills are most valued by recruiters.