IDENTIFYING PATTERNS AND TRENDS IN CAMPUS PLACEMENT DATA USING MACHINE LEARNING

1.INTRODUCTION

1.1 OVERVIEW

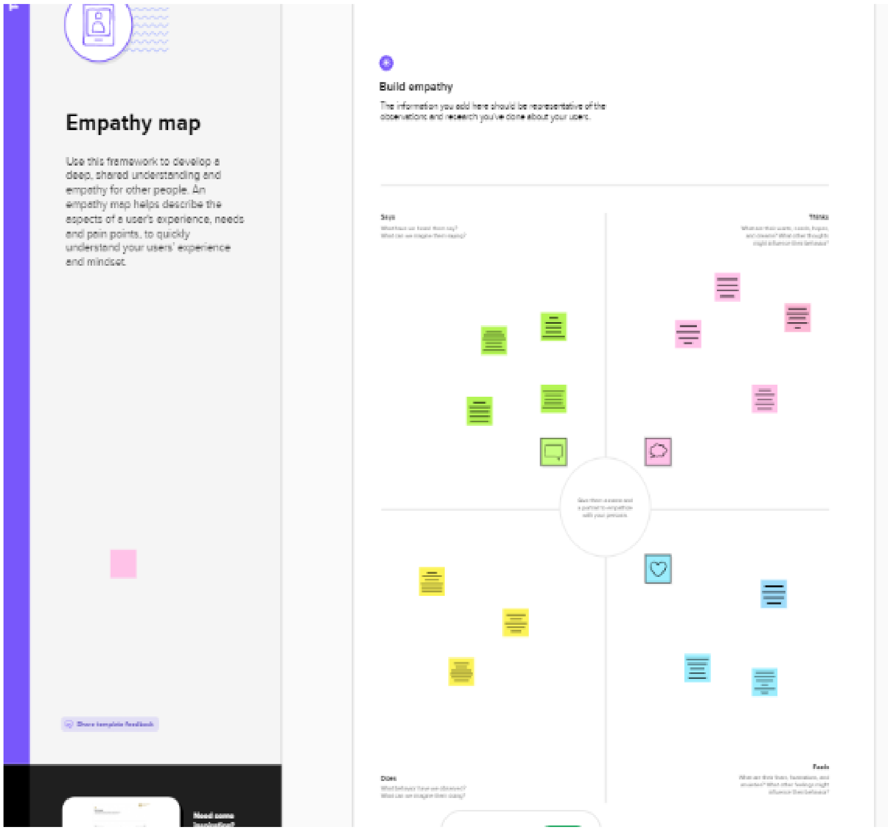
- Campus recruitment is a strategy for sourcing engaging and hiring young talent for internship and entry-level position.
- College recruiting is typically a tactic for medium-to large-sized companies with high-volume recruiting needs, but can range from small efforts to large-scale operations.
- university 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.
- we will be using algorithm such as KNN, SVM and ANN from this the best model is selected and saved in .pkl format.
- we will be doing flask integration and IBM deployment.

1.2 PURPOSE

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

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 EMPATHY MAP



Empathy map

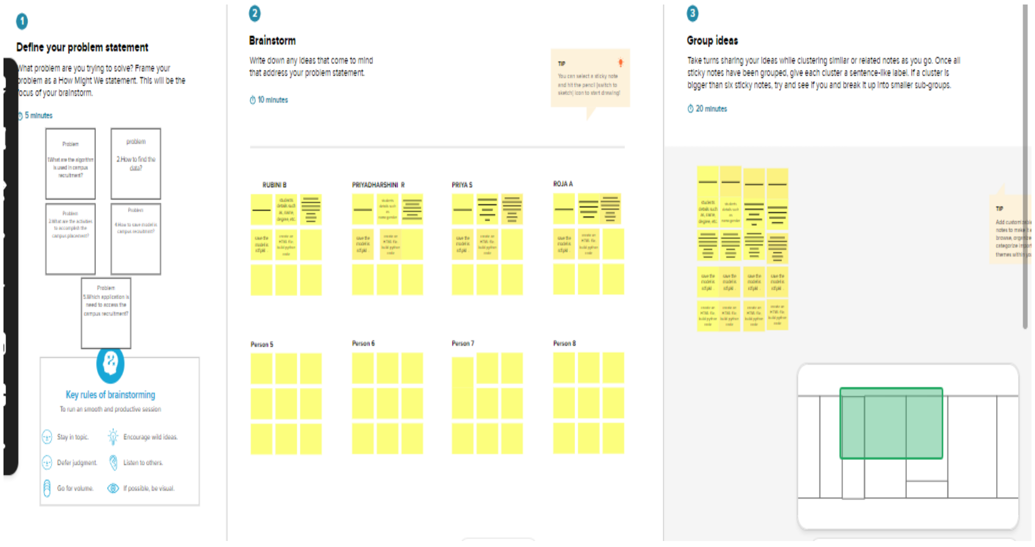
Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

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

Says
What have we heard them say?
What can we imagine them saying?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
Give them a name and a portrait to empathize with your persona.
Does
What behavior have we observed?
What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?
Share template feedback
Need some inspiration?

2.2 IDEATION & BRAINSTORMING MAP



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
1.What are the algorithm is used in campus recruitment?

problem
2.How to find the data?

Problem
3.What are the activities to accomplish the campus placement?

Problem
4.How to save model is campus recruitment?

Problem
5.Which application is need to access the campus recruitment?

Key rules of brainstorming

To run an 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

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

3

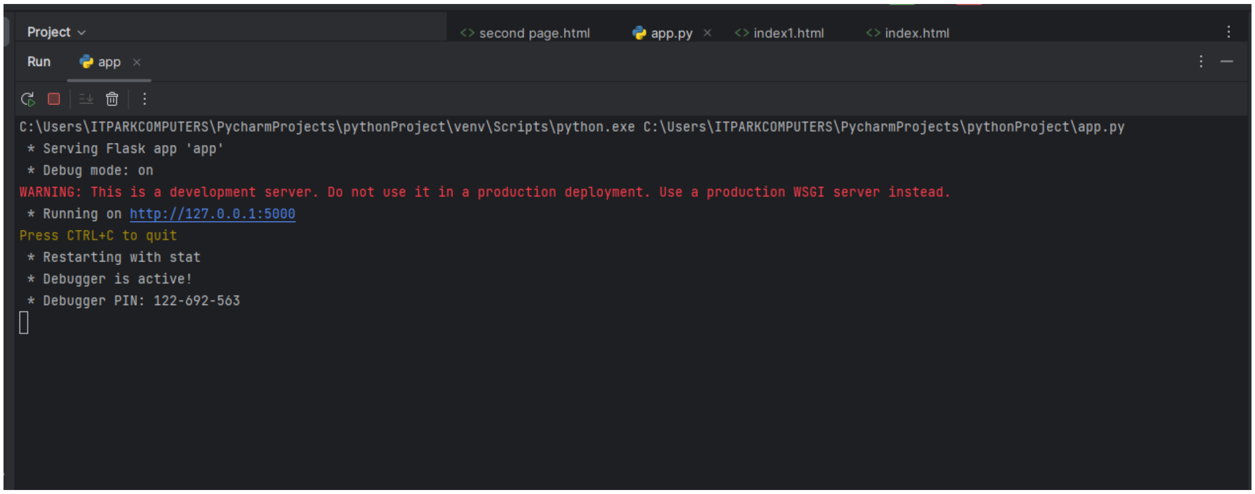
Group ideas

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 and break it up into smaller sub-groups.

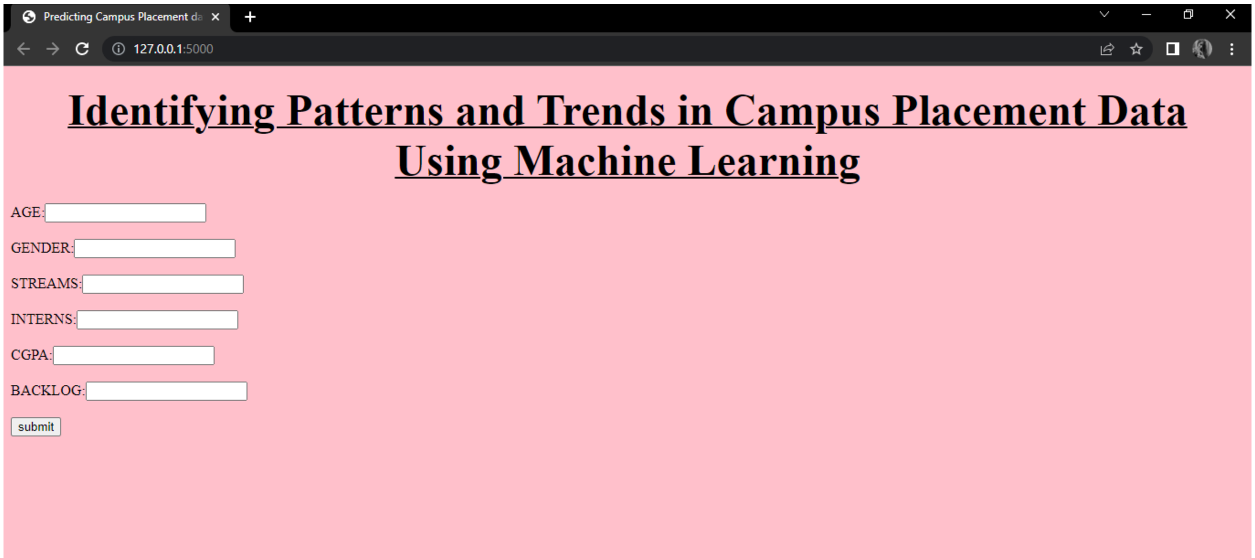
20 minutes

TIP
Add customizable notes to make it e browse, organize categorize import themes within yo

3. RESULT



```
C:\Users\ITPARKCOMPUTERS\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ITPARKCOMPUTERS\PycharmProjects\pythonProject\app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 122-692-563
```



Identifying Patterns and Trends in Campus Placement Data Using Machine Learning

AGE:

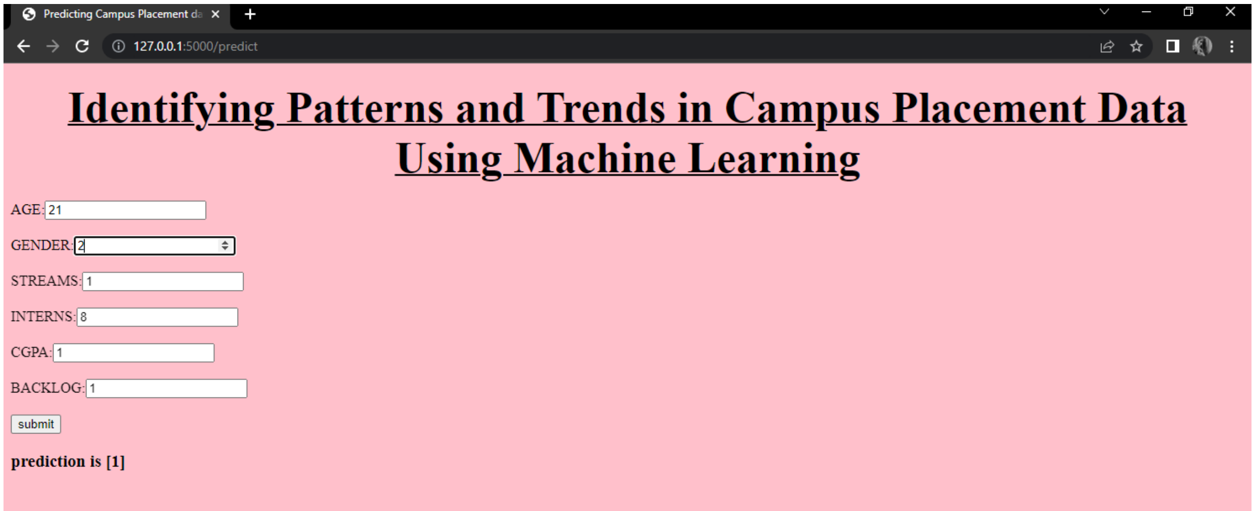
GENDER:

STREAMS:

INTERNS:

CGPA:

BACKLOG:



Identifying Patterns and Trends in Campus Placement Data Using Machine Learning

AGE: 21

GENDER: 2

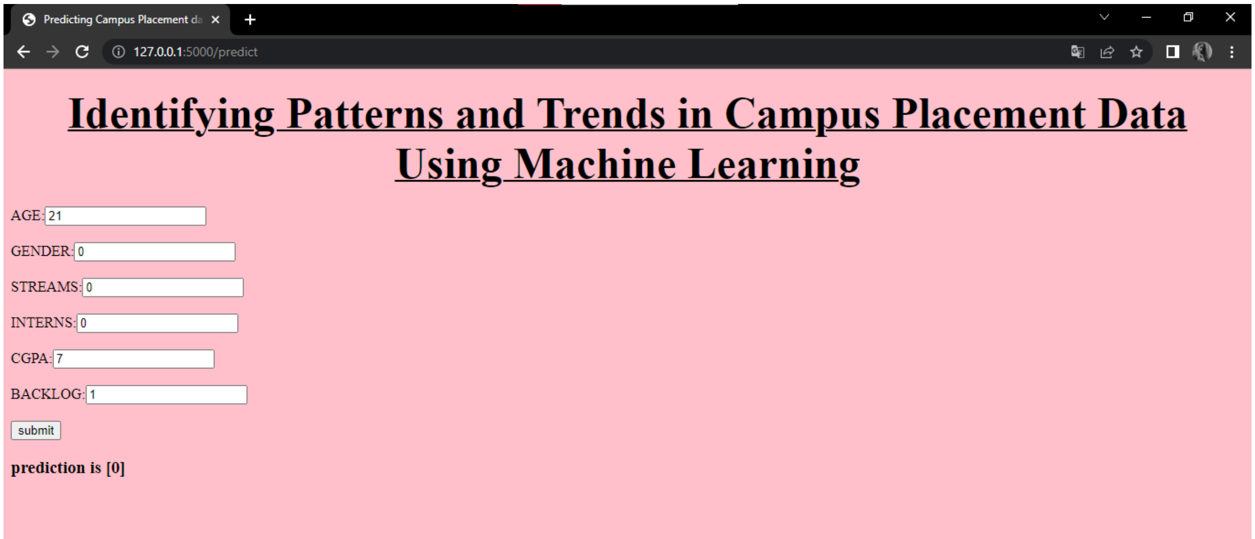
STREAMS: 1

INTERNS: 8

CGPA: 1

BACKLOG: 1

prediction is [1]



Identifying Patterns and Trends in Campus Placement Data Using Machine Learning

AGE: 21

GENDER: 0

STREAMS: 0

INTERNS: 0

CGPA: 7

BACKLOG: 1

prediction is [0]

4 ADVANTAGES & DISADVANTAGES

ADVANTAGES

- It is a Automatic.
- It is used in various fields.
- It can handle varieties of data.
- Scope of advancement.
- Can identify trends and pattern.

- Considered best for Education.

DISADVANTAGES

- Chances of error or fault are more
- Data requirement is more
- Time-consuming and more resources required
- Inaccuracy of interpretation of data
- More space required

5 APPLICATION

- 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.

6 CONCLUSION

- The campus placement task is extremely a lot of vital from the organization's point of view as well as the student's point of view.
- In this respect to advance the student's performance, an effort has been studied and predicted using the classification algorithms Decision Tree, Naive Bayes, and the Random forest algorithm to authenticate the methodologies.
- The results recommend that amongst the machine learning algorithm verified, the Random Forest classifier has the potential to significantly progress the conventional classification methods for use in placement.

7 FUTURE SCOPE

- The scope of Machine Learning is not limited to the investment sector.
- Rather, it is expanding across all fields such as banking and finance, technology, media & entertainment, gaming and the automotive industry.
- As the Mach==is very high, there are some areas where

researchers are working toward revolutionizing the world for the future.

8 APPENDIX

Source Code

Import the libraries:

```
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')
```

Read the Dataset:

```
df = pd.read_csv(r"/content/collegePlace.csv")
df.head()

df.shape
```

Data preparation:

Handling missing values

```
df.info()

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

Handling outliers

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

Handling categorical values

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

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

df.info()
```

Univariate values

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

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

Bivariate analysis

```
from matplotlib.offsetbox import martist
plt.figure(figsize=(30,5))
plt.subplot(1,4,1)
sns.countplot(x="PlacedOrNot",data=df, ec='black')
plt.subplot(1,4,2)
sns.countplot(y="Stream",data=df, ec='black')
plt.show()
```

Multivariate analysis

```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(x='PlacedOrNot', data=df, hue='CGPA', ec='black')

sns.swarmplot(x='PlacedOrNot',y='CGPA', hue='Stream', data=df)

df.describe()
```

splitting the data into train and test

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

y

sc = StandardScaler()
x = sc.fit_transform(x)
x = pd.DataFrame(x)

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size= 0.11,
stratify=y, random_state=42)


print(x_train.shape)
print(x_train.shape)
```

Training the model in multiple algorithms

1.SVM model

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(x_train,y_train)
SVC()


from sklearn import svm
classifier = svm.SVC()
x_test = np.array(x_test, dtype = float)
y_test = np.array(y_test, dtype = float)
```

```
classifier.fit(x_train, y_train)
SVC()

x_test_prediction = classifier.predict(x_test)
y_pred= accuracy_score(x_test_prediction,y_test)
y_pred
```

2.KNN Model

```
best_k = {"Regular":0}
best_score = {"Regular":0}
for k in range(3, 50, 2):
  knn_temp = KNeighborsClassifier(n_neighbors=k)
  knn_temp.fit(x_train, y_train)
  knn_temp_pred = knn_temp.predict(x_test)
  score = metrics.accuracy_score(y_test, knn_temp_pred) * 100
  if score >=  best_score["Regular"]and score < 100:
    best_score["Regular"] = score
    best_k["Regualar"] = k



print("---Results---\nk: {}\nScore: {}".format(best_k, best_score))
knn = KNeighborsClassifier(n_neighbors=best_k["Regualar"])
knn.fit(x_train, y_train)
knn_pred = knn.predict(x_test)
testd = accuracy_score(knn_pred, y_test)
```

ANN

```
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 second 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 th model
classifier.fit(x_train, y_train, batch_size = 20, epochs = 100)
```

MODEL DEPLOYMENT

Save the best model

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

input_data = [[22,0,2,1,8,1]]

prediction = knn.predict(input_data)
print(prediction)
if (prediction[0]==0):
  print('not placed')
else:
      print('placed')
```