Identifying patterns and trends in campus placement data using machine LEARNING

**1.INTRODUCTION**

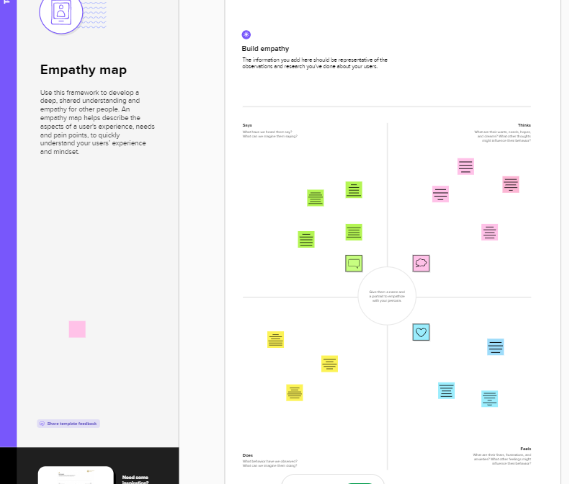
1.1 OVERVIEW

* 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.

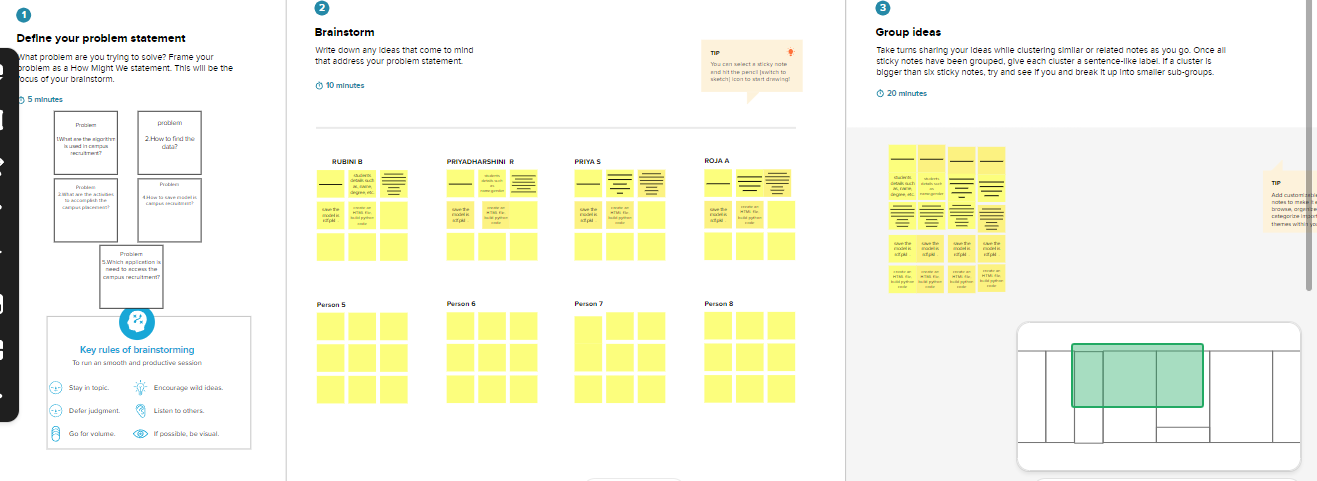
**1.2 PURPOSE**

* 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.

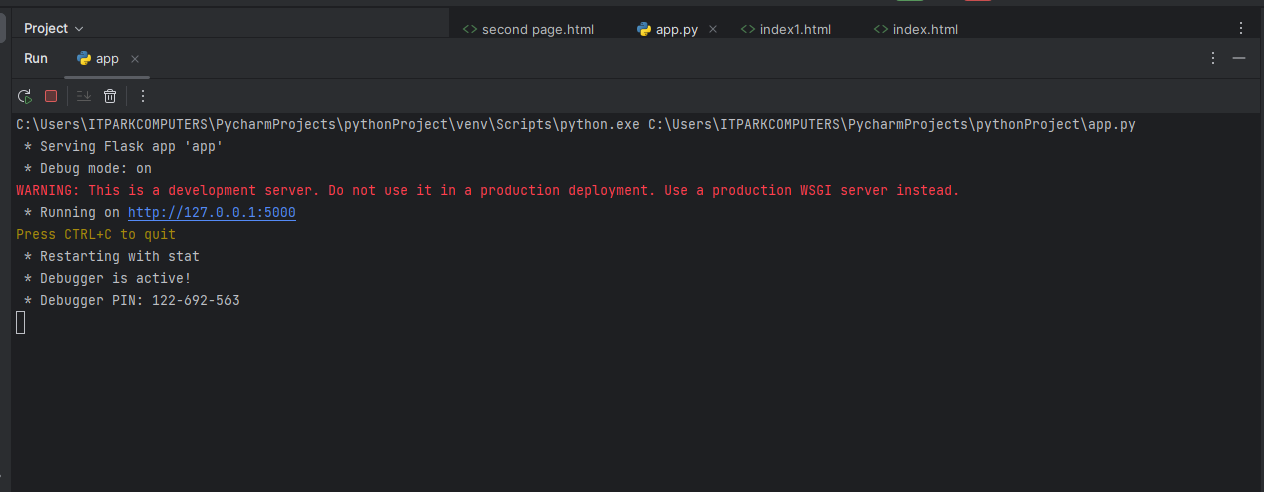
***2. PROBLEM DEFINITION & DESIGN THINKING***

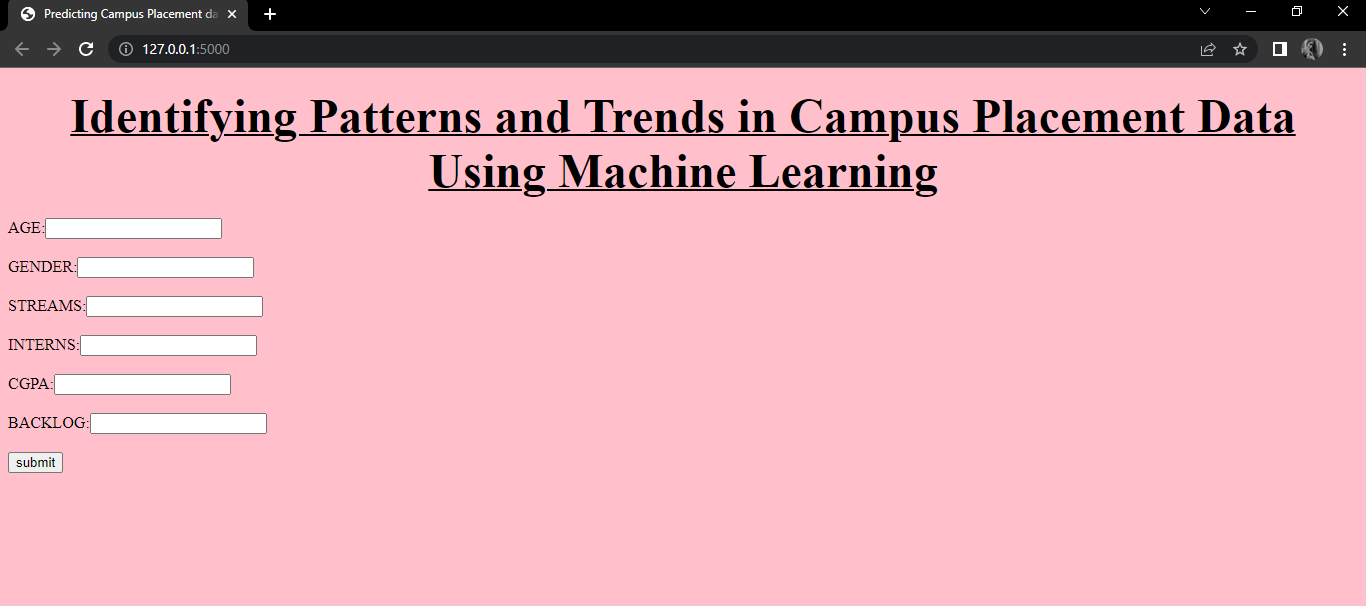
***2.1 EMPATHY MAP***

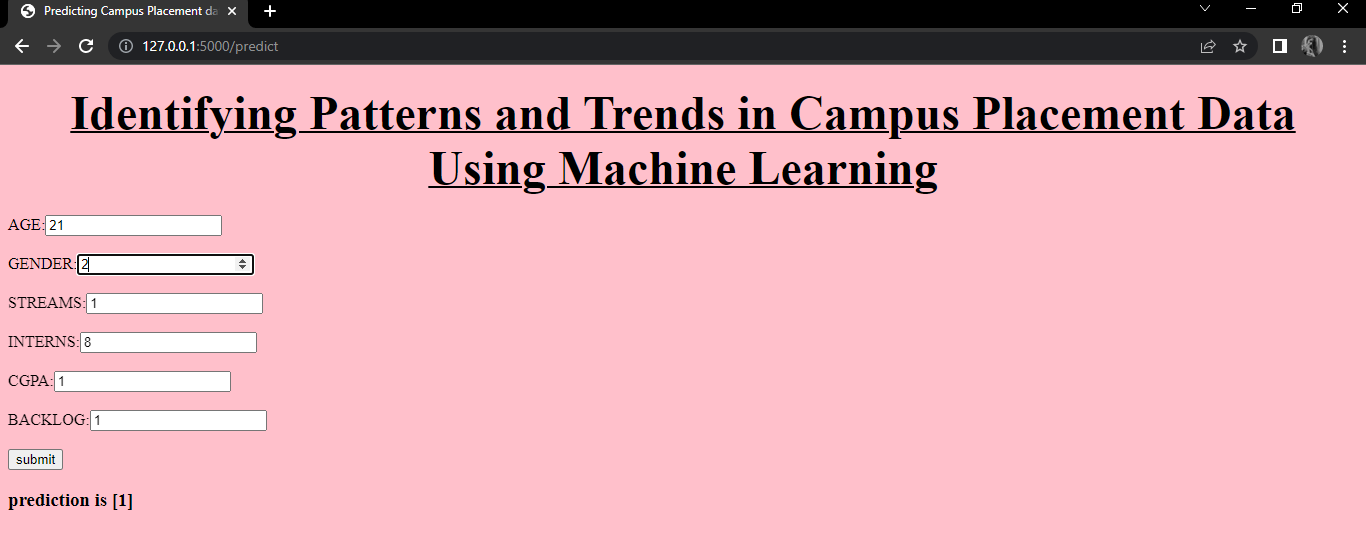
* 1. ***IDEATION & BRAINSTORMING MAP***



***3. RESULT***

******







***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 Learningis 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')