RAJESWARI VEDACHALAM GOVERNMENT ARTS COLLEGE CHENGALPATTU

COMPUTER SIENCE DEPARTMENT

Identifying patterns and trends in campus placement data using machine learning

TEAM MEMBERS NAME : T.ABIRAMI

J.SHARMILA

R.MEENA

M.HARIHARAN

CLASS: BSC COMPUTER SCIENCE [lll YEAR]

1.INTRODUCTION

1.1 Overview

Campus recruitment is a strategy for sourcing, engaging and hiring young talent forinternship and entry-level positions. College recruiting is typically a tactic for medium- to large-sized companies with high-volume recruiting needs, but canrange from small efforts (like working with university career centers to source potential candidates) to large-scale operations (like visiting a wide array of college sand 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 testthe data with these algorithms. From this the best model is selected and saved in.pkl format. We will be doing flask integration and IBM deployment.

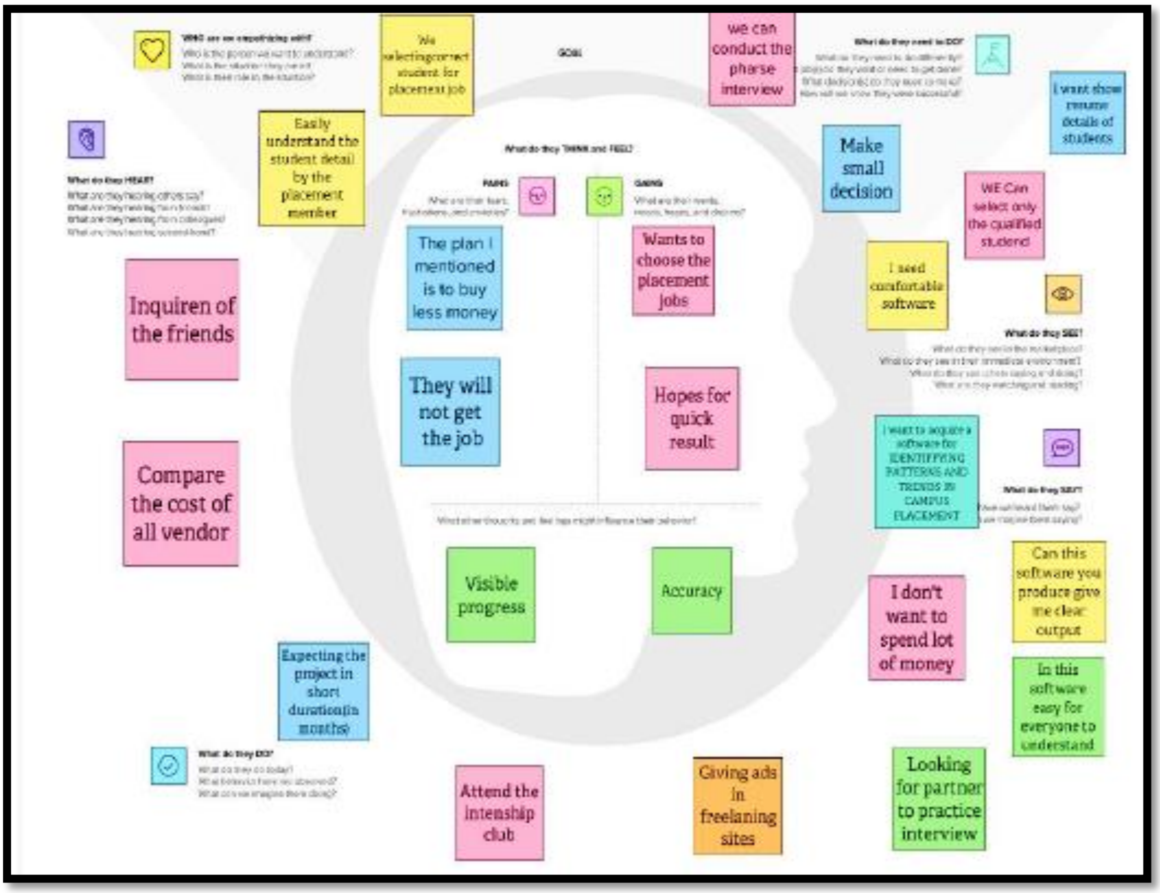
1.2 Purpose

- A selection probability indicator **lets students get an sense of where they're standing and what to do to ensure a decent selection**.
- A placement predictor is a device that can forecast the probability or form of business that a student in the pre-final year has chances of placing.
- Each student dreams of having a work offer in their hands before leaving college. A selection probability indicator lets students get an sense of where they're standing and what to do to ensure a decent selection. A placement predictor is a device that can forecast the probability or form of business that a student in the pre-final year has chances of placing.
- While a forecasting program could help in the academic preparation of an institution for future years. With the emergence of data mining and machine learning, through analyzing the data set of the previous student year, numerous predictive models were applied. To find placement prediction.

2.PROBLEM DEFINITON&DESIGN THINKING

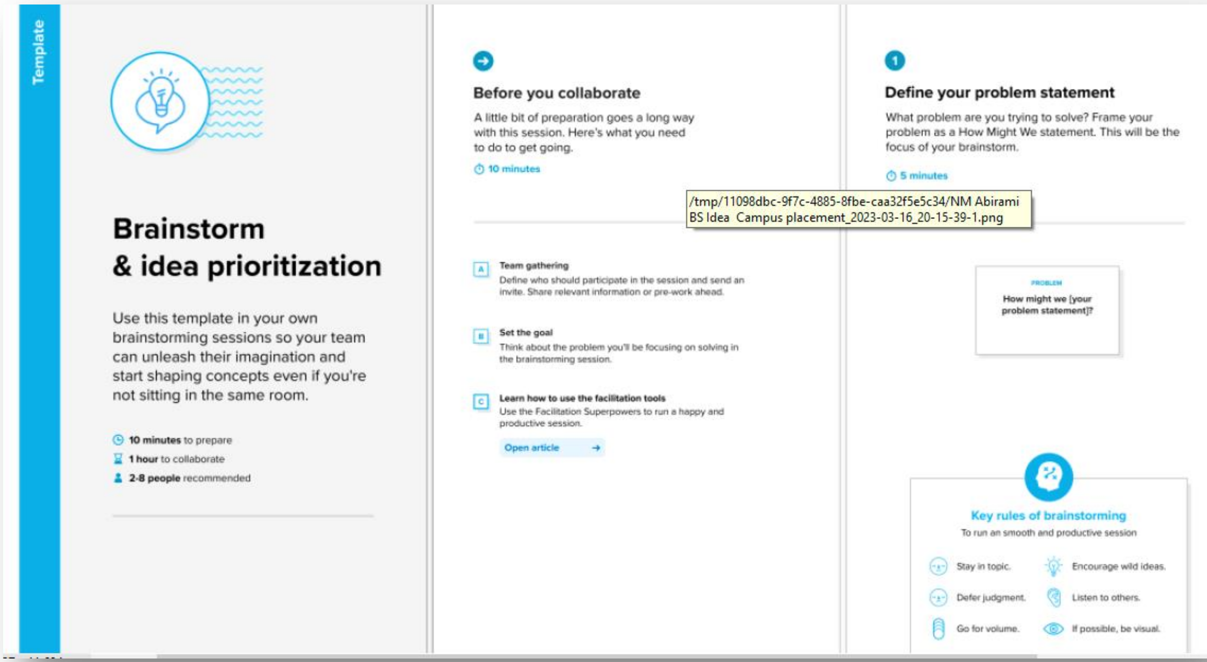
2.1 Empathy Map

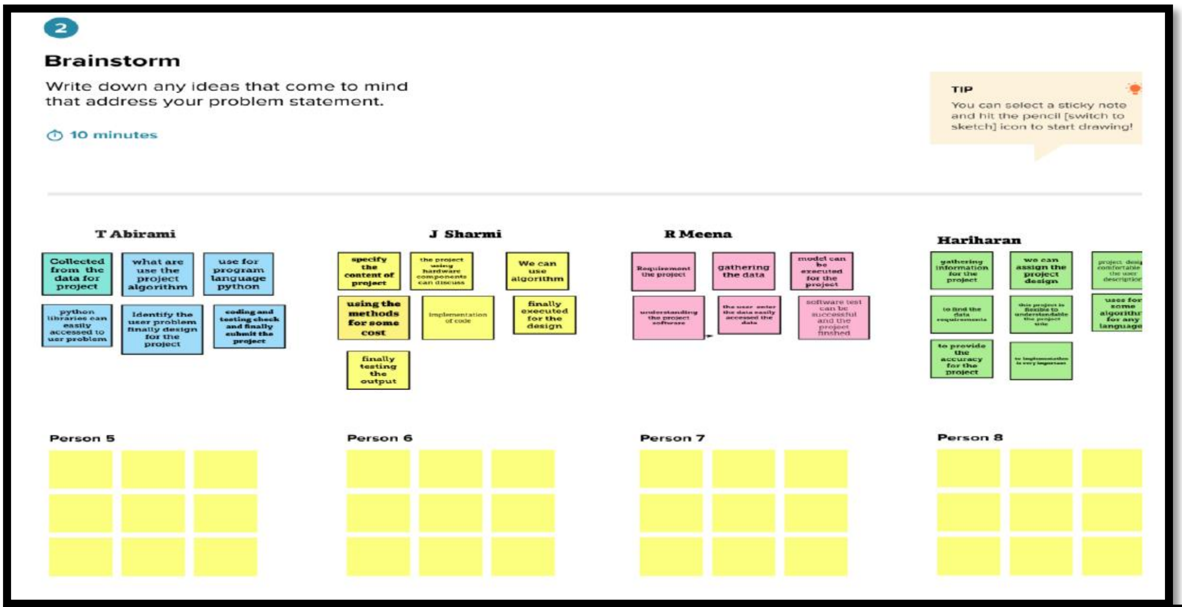
In the ideation phase we have empathized have a client placement trends analysis and we have acquired details. Which are represented in the Empathy Map

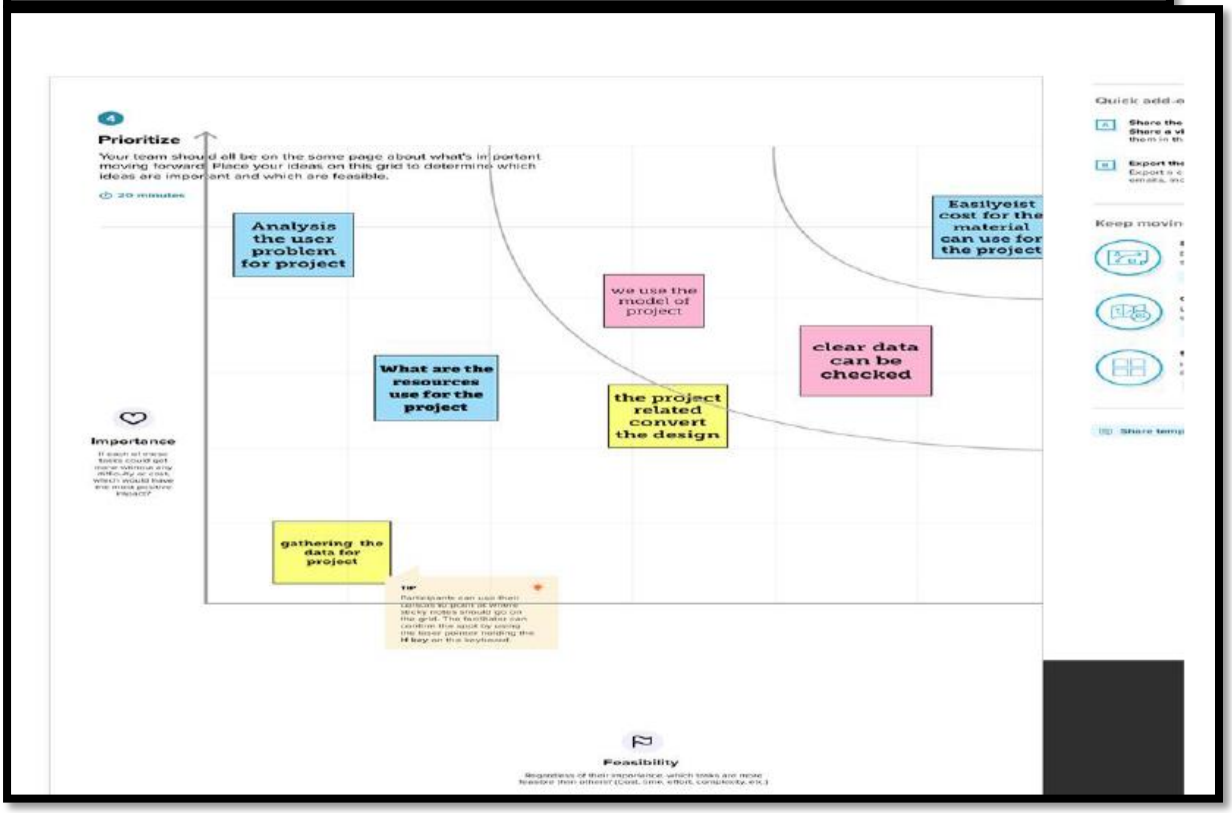


2.2 Ideation & Brainstroming Map

Under this activity our team members have gathered and discussed various ideas to solve our project problem. Each member contributed 6 to 10 ideas. After gathering all ideas we have assessed the impact flexibility of each points finaly we have assigned the priority for each points based on the impact value.




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!
T Abirami
Collected from the data for project
what are use the project algorithm
use for program language python
python libraries can easily accessed to user problem
identify the user problem finally design for the project
coding and testing check and finally submit the project
J Sharmi
specify the content of project
We can use algorithm
using the methods for some cost
finally executed for the design
finally testing the output
R Meena
Requirement the project
gathering the data
model can be executed for the project
understanding the project software
software test can be successful and the project finished
Hariharan
gathering information for the project
we can assign the project design
use for some algorithm for any language
to provide the accuracy for the project
Person 5
Person 6
Person 7
Person 8


4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
Analysis the user problem for project
Easilyeist cost for the material can use for the project
we use the model of project
clear data can be checked
What are the resources use for the project
the project related convert the design
gathering the data for project
Importance
Feasibility

3.RESULT

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. Link:

https://www.kaggle.com/code/neesham/prediction-of-placements/data

Importing the libraries

```
#import libraries


import numpy as np


import pandas as pd
import os
import seaborn as sns

import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')


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

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 wehave to give the directory of the csv file.

```
from google.colab import files
uploads=files.upload()


#Read the file


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

```
df=pd.read_csv('/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()

```
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
```

```
#checking null values
df.isnull().sum()
```

```
#checking null values
df.isnull().sum()
```

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

```
[ ] df['Stream'].unique()
```

```
array(['Electronics And Communication', 'Computer Science',
       'Information Technology', 'Mechanical', 'Electrical', 'Civil'],
      dtype=object)
```

```
[ ] #creating new column
df['CGPA_']=['1-8' if x<=5 else "1-3" if x>5 and x<=6 else '7+' for x in df['CGPA']]
df.head()
```

	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot	CGPA_
0	22	Male	Electronics And Communication	1	8	1	1	1	7+
1	21	Female	Computer Science	0	7	1	1	1	7+
2	22	Female	Information Technology	1	6	0	0	1	1-3
3	21	Male	Information Technology	0	8	0	1	1	7+
4	22	Male	Mechanical	0	8	1	0	1	7+

Removing data

```
#Removing Hostel_column
df=df.drop(['Hostel'],axis=1)
df=df.drop(['CGPA_'],axis=1)
```

```
[ ] df
```

	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	Male	Electronics And Communication	1	8	1	1
1	21	Female	Computer Science	0	7	1	1
2	22	Female	Information Technology	1	6	0	1
3	21	Male	Information Technology	0	8	1	1
4	22	Male	Mechanical	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	Male	Information Technology	0	7	0	0
2962	23	Male	Mechanical	1	7	0	0
2963	22	Male	Information Technology	1	7	0	0

```
#creating dummy dataframe for categorical values
df_cat=df.select_dtypes(include='int')
df_cat.head()
```

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

```
df.describe(include='all')
```

	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
count	2966.000000	2966	2966	2966.000000	2966.000000	2966.000000	2966.000000
unique	NaN	2	6	NaN	NaN	NaN	NaN
top	NaN	Male	Computer Science	NaN	NaN	NaN	NaN
freq	NaN	2475	776	NaN	NaN	NaN	NaN
mean	21.485840	NaN	NaN	0.703641	7.073837	0.192178	0.552596
std	1.324933	NaN	NaN	0.740197	0.967748	0.394079	0.497310
min	19.000000	NaN	NaN	0.000000	5.000000	0.000000	0.000000
25%	21.000000	NaN	NaN	0.000000	6.000000	0.000000	0.000000
50%	21.000000	NaN	NaN	1.000000	7.000000	0.000000	1.000000
75%	22.000000	NaN	NaN	1.000000	8.000000	0.000000	1.000000
max	30.000000	NaN	NaN	3.000000	9.000000	1.000000	1.000000

df.isnull().any()

```
#finding null values
df.isnull().any()
```

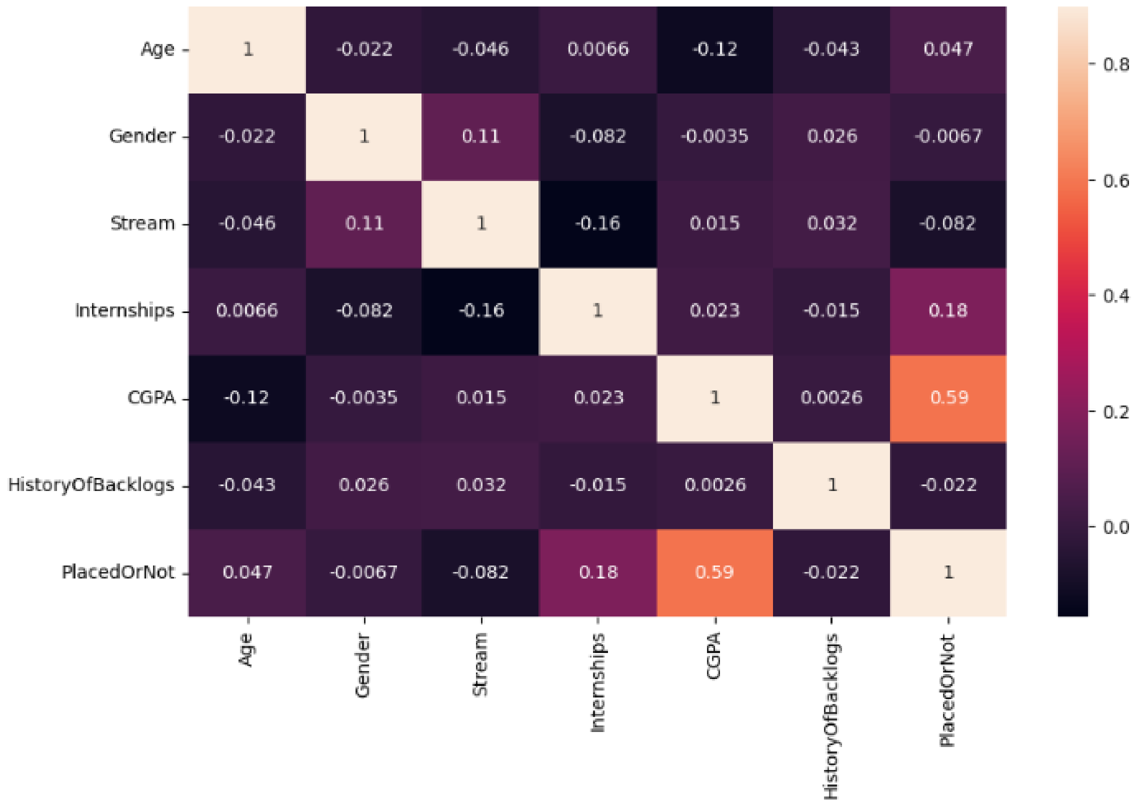
```
Age                  False
Gender               False
Stream               False
Internships          False
CGPA                 False
HistoryOfBacklogs    False
PlacedOrNot          False
dtype: bool
```

```
#data type
df.info()
```

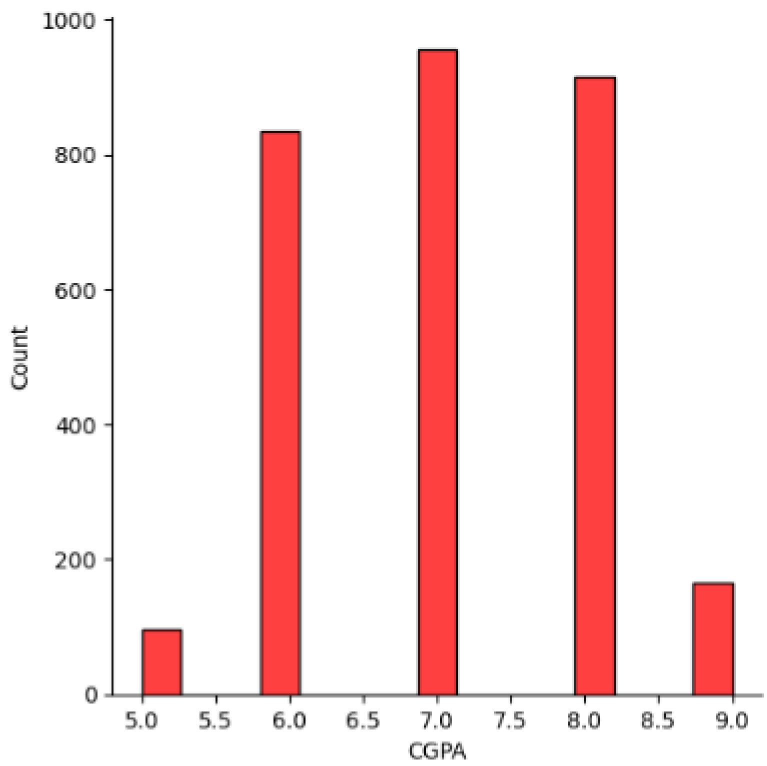
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2966 entries, 0 to 2965
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Age                2966 non-null   int64
 1   Gender             2966 non-null   int64
 2   Stream             2966 non-null   int64
 3   Internships        2966 non-null   int64
 4   CGPA               2966 non-null   int64
 5   HistoryOfBacklogs  2966 non-null   int64
 6   PlacedOrNot        2966 non-null   int64
dtypes: int64(7)
memory usage: 162.3 KB
```

```
plt.figure(figsize=(10,6),dpi=100)
sns.heatmap(df.corr(),vmax=0.9,annot=True)
```

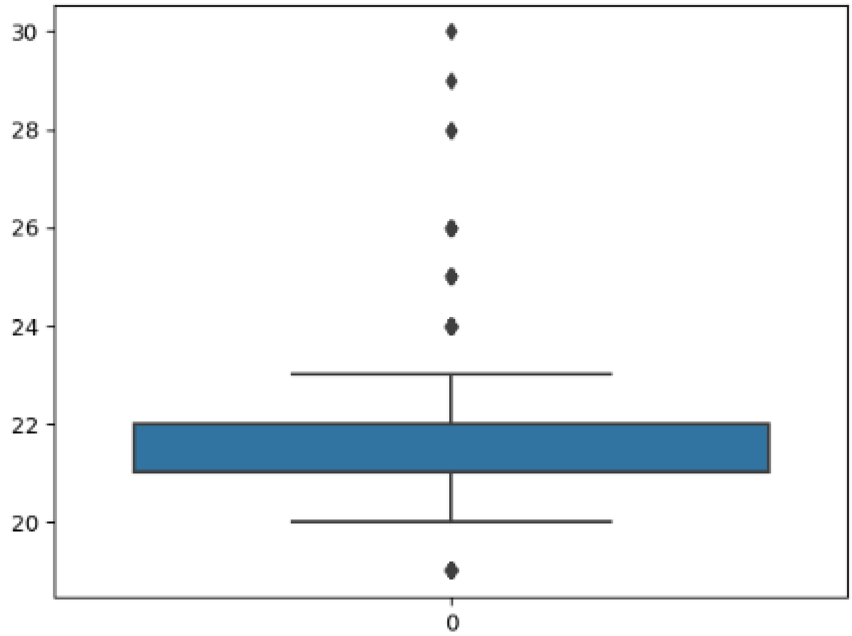
<Axes: >




```
sns.displot(df['CGPA'],color='red')
```




```
sns.boxplot(df['Age'])
```



_ Handling outliers

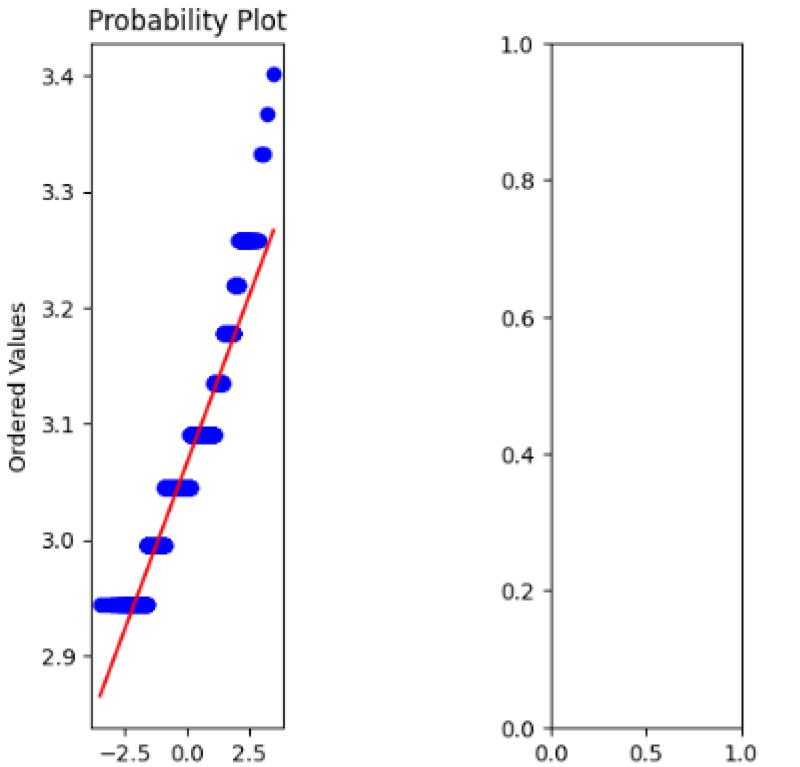
Outliers counting

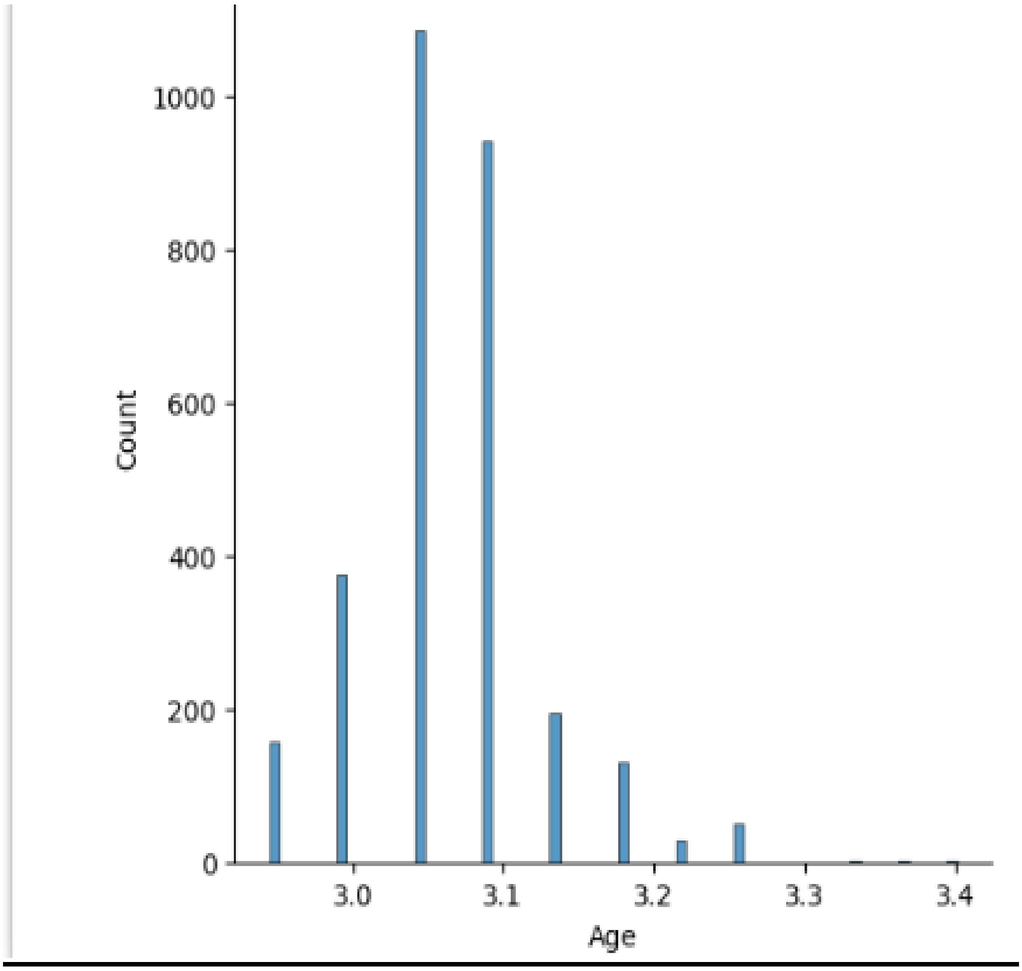
```
#finding the count of outliers
#IQR = q3-q1      upperbound=q3+(1.5*IQR), lower bound=q1-(1.5*IQR)
q1 = np.quantile(df['Age'],0.25)
q3 = np.quantile(df['Age'],0.75)
print('Q1 = {}'.format(q1))
print('Q3 = {}'.format(q3))
IQR=q3-q1
print('IQR value is{}'.format(IQR))
upperBound=q3+(1.5*IQR)
lowerBound=q1-(1.5*IQR)
print('the upper bound value is {} & the lower bound value is {}'.forma
t(upperBound,lowerBound))
```

```
Q1 = 21.0
Q3 = 22.0
IQR value is1.0
the upper bound value is 23.5 & the lower bound value is 19.5
```

```
#handling outline
```

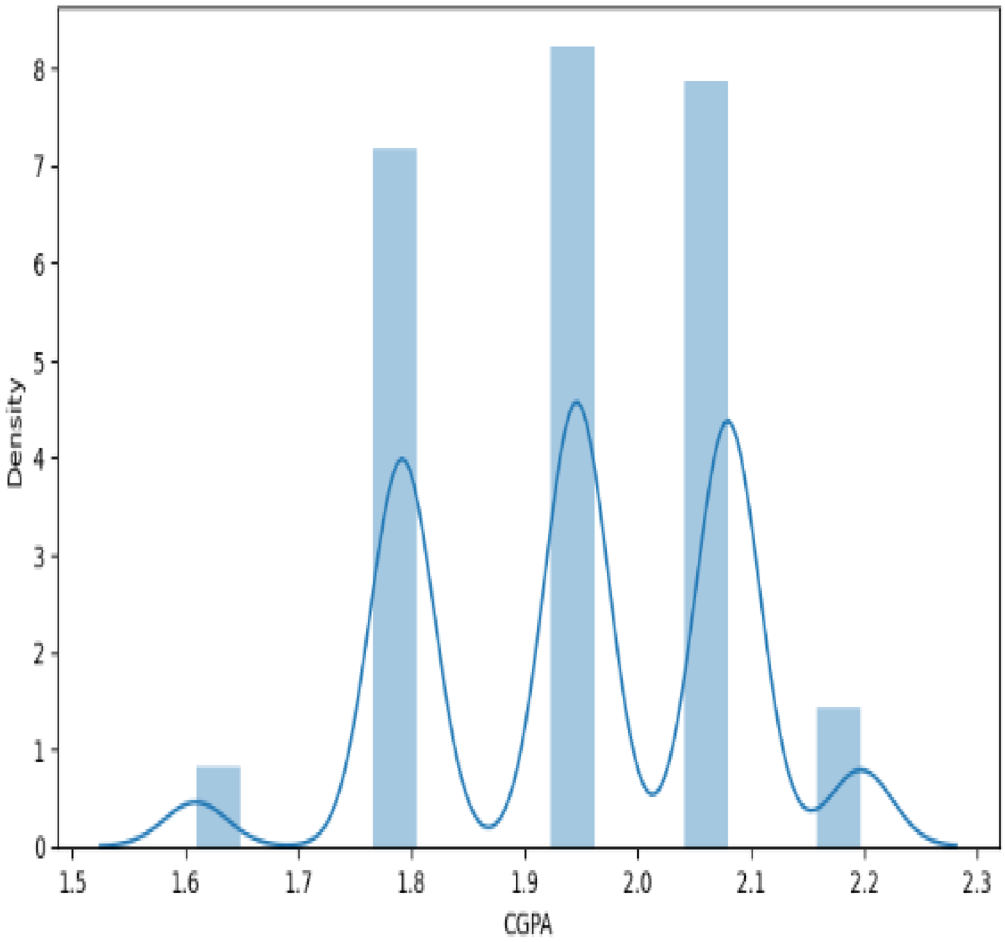
```
from scipy import stats
plt.figure(figsize=(20,4))
plt.subplot(1,3,2)
sns.displot(df['Age'])
plt.subplot(1,3,1)
stats.probplot(np.log(df['Age']),plot=plt)
plt.subplot(1,3,3)
sns.displot(np.log(df['Age']))
```






```
def transformationplot(feature):
    plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    sns.distplot(feature)
transformationplot(np.log(df['CGPA']))
```

```
def transformationplot(feature):
    plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    sns.distplot(feature)
transformationplot(np.log(df['CGPA']))
```



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(['Computer Science','Information Technology','Electronics And Communicatio
n','Mechanical','Electrical','Civil'],
[0,1,2,3,4,5])

df['Gender'] = df['Gender'].replace({'Female':0, 'Male':1})

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

```
df
```

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

2966 rows × 7 columns

Exploratory Data Analysis:

Visual analysis

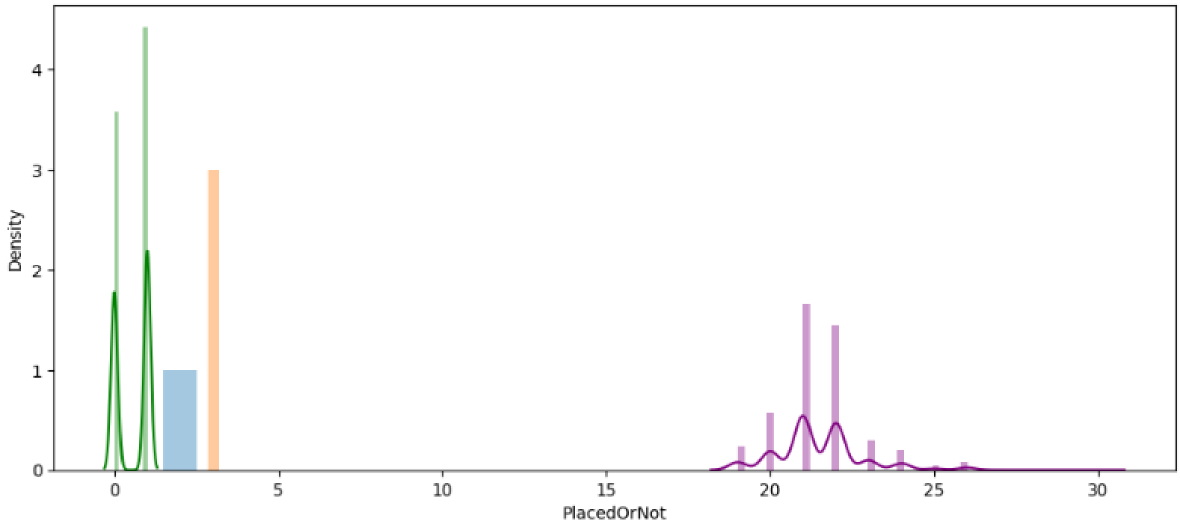
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions

Univariate analysis:

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

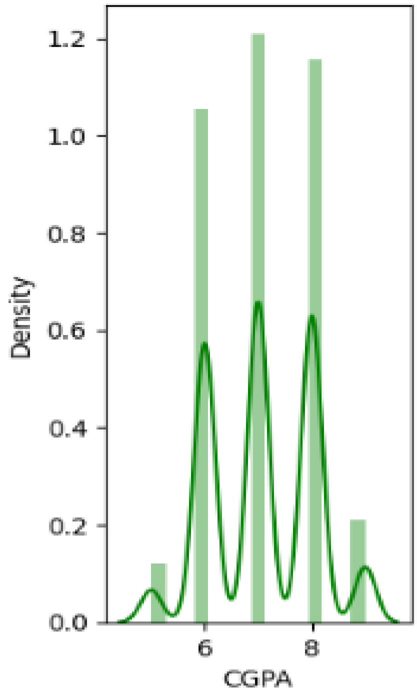
```
#exploratory data Analysis
plt.figure(figsize=(12,5))
sns.distplot(2,1,2)
sns.distplot(df['Age'],color='purple')
sns.distplot(3,3,2)
sns.distplot(df['PlacedOrNot'],color='green')
```

```
<Axes: xlabel='PlacedOrNot', ylabel='Density'>
```




```
#univariate anaiysis
plt.figure(figsize=(23,9))
plt.subplot(1,5,2)
sns.distplot(df['CGPA'],color='Green')
```

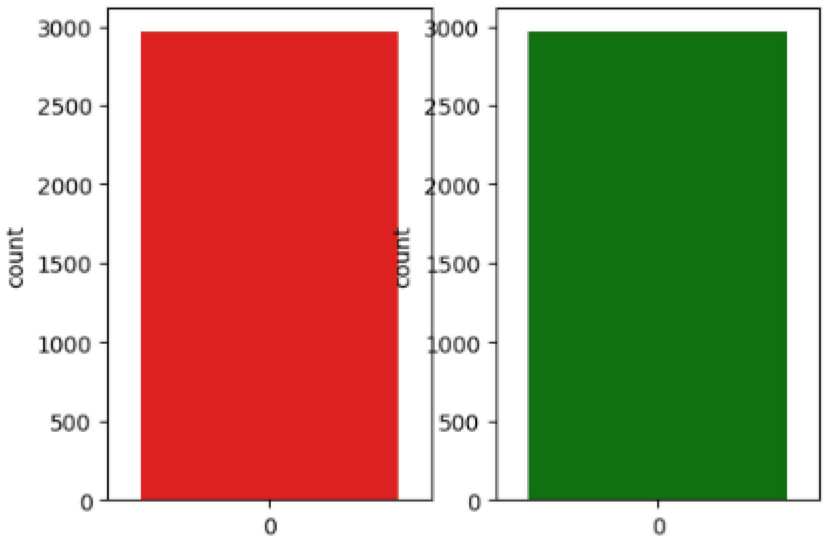
```
<Axes: xlabel='CGPA', ylabel='Density'>
```



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

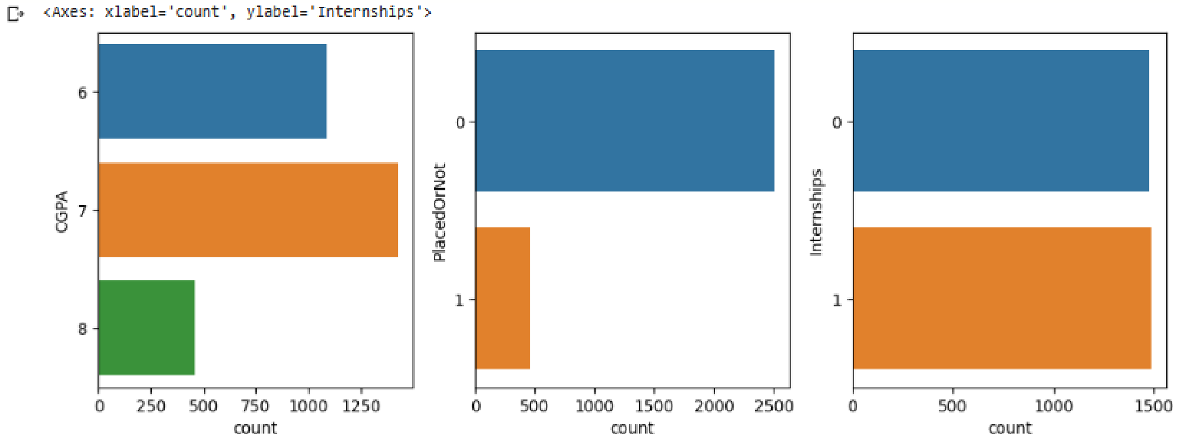
```
plt.figure(figsize=(12,4))
plt.subplot(1,4,1)
sns.countplot(df['Gender'],color='r')
plt.subplot(1,4,2)
sns.countplot(df['Age'],color='g')
plt.show()
```



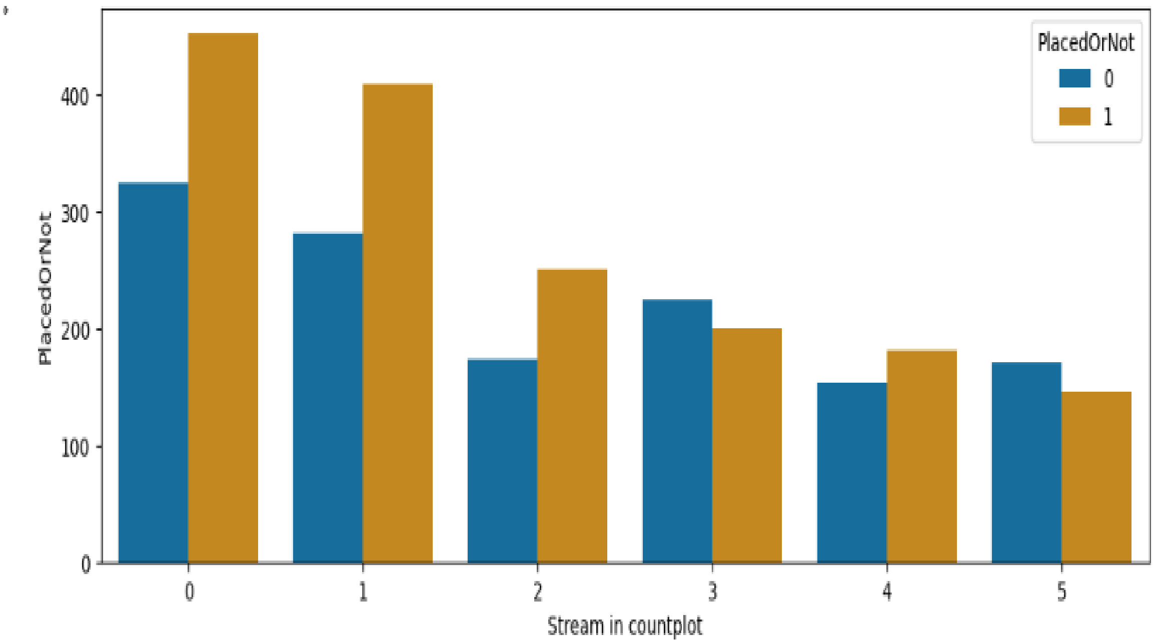
Multivariate analysis

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

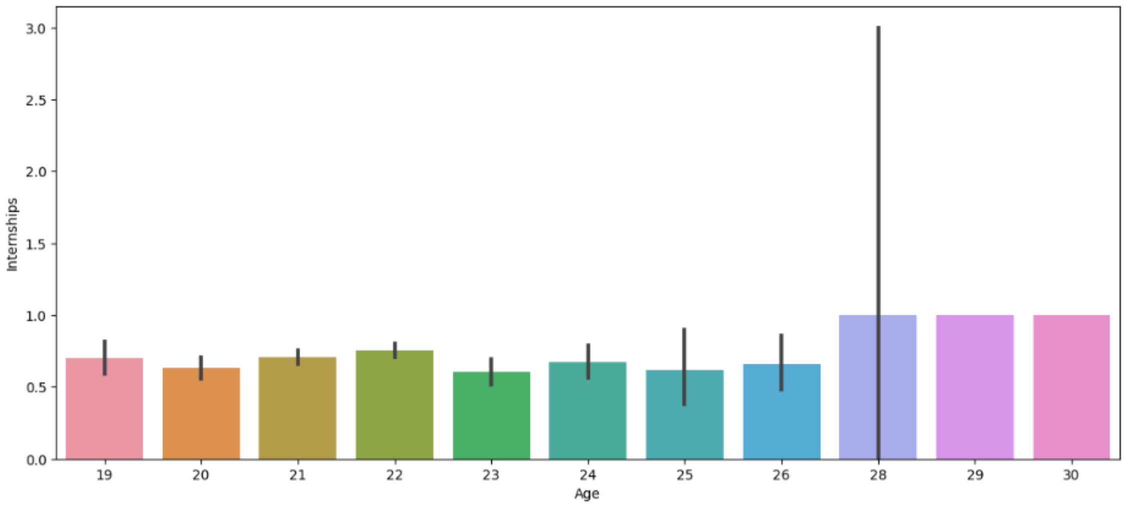
```
plt.figure(figsize=(12,4))
plt.subplot(131)
sns.countplot(df['CGPA'],y=df['Age'])
plt.subplot(132)
sns.countplot(df['PlacedOrNot'],y=df['Age'])
plt.subplot(133)
sns.countplot(df['Internships'],y=df['Age'])
```




```
plt.figure(figsize=(12,4))
sns.countplot(x=df["Stream"],hue=df["PlacedOrNot"], palette="colorblind")
plt.xlabel("Stream in countplot")
plt.ylabel("PlacedOrNot")
plt.show()
```



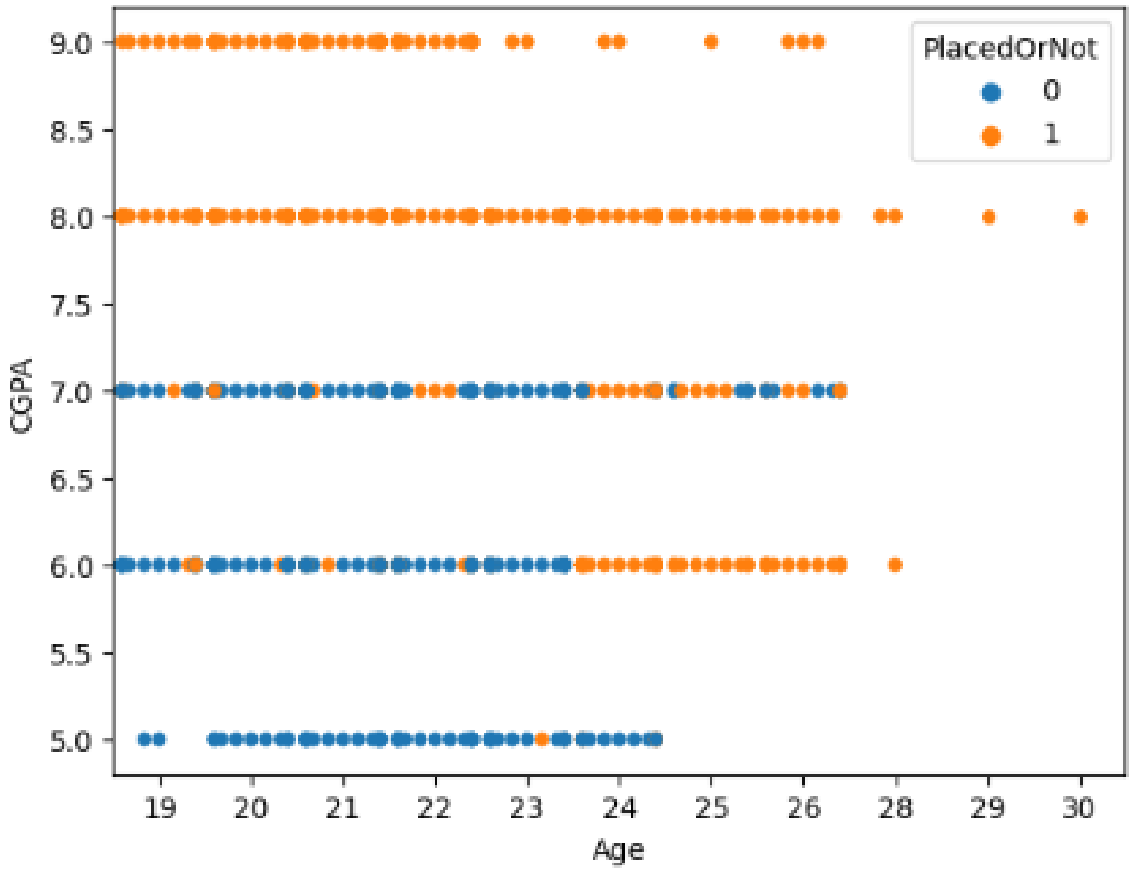

```
plt.figure(figsize=(14,6))
sns.barplot(data=df,x='Age',y='Internships')
plt.show()
```




```
sns.swarmplot(x=df['Age'],y=df['CGPA'],hue=df['PlacedOrNot'])
```

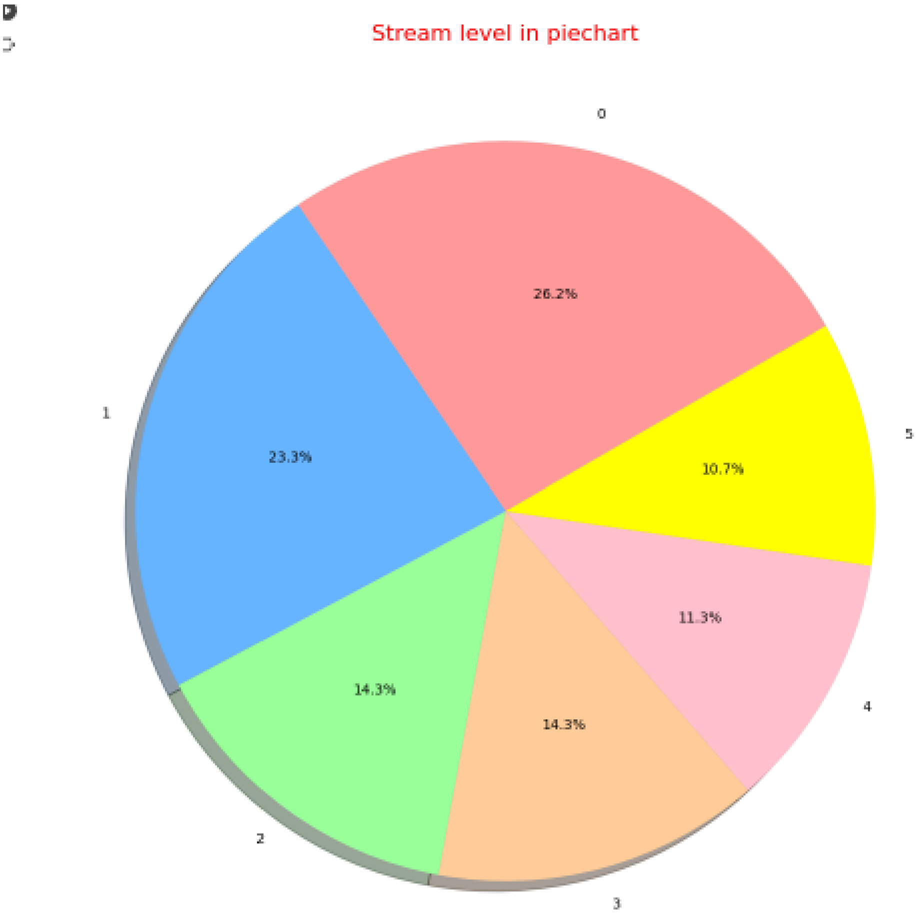
```
sns.swarmplot(x=df['Age'],y=df['CGPA'],hue=df['PlacedOrNot'])
```

```
<Axes: xlabel='Age', ylabel='CGPA'>
```




```
labels=df["Stream"].value_counts().index

sizes=df["Stream"].value_counts()
colors=['#ff9999','#66b3ff','#99ff99','#ffcc99',"pink","yellow"]
plt.figure(figsize=(12,12))
plt.pie(sizes,labels=labels,rotatelabels=False,autopct='%1.1f%%',colors=colors,shadow=True
,startangle=30)
plt.title("Stream level in piechart",color='r',fontsize=16)
plt.show()
```



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.

```
#scaling data
names=data[column]
sc=StandardScaler()
x_bal=sc.fit_transform(x)
x_bal=pd.DataFrame(x_bal,columns=data.columns)
#independent variables
x=df.iloc[:,0:6]
x.head()

#dependent  variables
y=df.iloc[:,6:]
y.head()
```

```
#scaling data
names=data[column]
sc=StandardScaler()
x_bal=sc.fit_transform(x)
x_bal=pd.DataFrame(x_bal,columns=data.columns)
```

```
#independent variables
x=df.iloc[:,0:6]
x.head()
```

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

```
#dependent   variables
y=df.iloc[:,6:]
y.head()
```

	PlacedOrNot
0	1
1	1
2	1
3	1
4	1

_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.

```
#split& train test data
from   sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=45)

print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
```

```
#split& train test data
from   sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=45)
```

```
print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
```

```
(2372, 6) (594, 6)
(2372, 1) (594, 1)
```

Model Building

SVM model

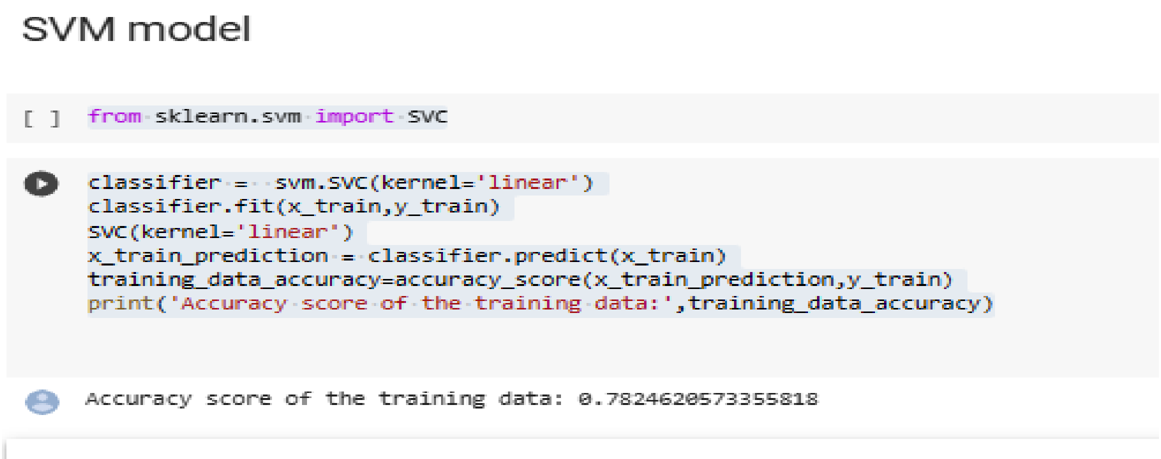
A function named Support vector machine is created and train and test data are passed as the parameters. Inside the function, SVM Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier,RandomForest
Classifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

model_accuracy={}
cv=KFold(n_splits=15,random_state=13,shuffle=True)

from sklearn.svm import SVC
```

```
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)
```



SVM model

```
[ ]  from sklearn.svm import SVC

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: 0.7824620573355818
```

KNN model

A function named KNN is created and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier 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)
  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["Regular"]=k
print("----Results-\nk:  {}nScore:{}".format(best_k,best_score))
knn= KNeighborsClassifier(n_neighbors=best_k["Regular"])
knn.fit(x_train,y_train)
knn_pred=knn.predict(x_test)
testd = accuracy_score(knn_pred,y_
```

Knn model

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

```
----Results-
k:  {'Regular': 13}nScore:{'Regular': 88.21548821548821}
```

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)
```

```
#fitting the model
classifier.fit (x_train,y_train,batch_size=20,epochs=100)
```

```
119/119 [==============================] - 0s 2ms/step - loss: 0.5584 - accuracy: 0.6703
Epoch 47/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5593 - accuracy: 0.6733
Epoch 48/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5632 - accuracy: 0.6640
Epoch 49/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5677 - accuracy: 0.6577
Epoch 50/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5570 - accuracy: 0.6766
Epoch 51/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5655 - accuracy: 0.6644
Epoch 52/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5551 - accuracy: 0.6716
Epoch 53/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5482 - accuracy: 0.6804
Epoch 54/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5537 - accuracy: 0.6703
Epoch 55/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5500 - accuracy: 0.6754
Epoch 56/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5618 - accuracy: 0.6657
Epoch 57/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5488 - accuracy: 0.6720
Epoch 58/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5620 - accuracy: 0.6678
Epoch 59/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5549 - accuracy: 0.6648
Epoch 60/100
119/119 [==============================] - 0s 3ms/step - loss: 0.5528 - accuracy: 0.6699
Epoch 61/100
119/119 [==============================] - 0s 3ms/step - loss: 0.5497 - accuracy: 0.6745
Epoch 62/100
119/119 [==============================] - 0s 3ms/step - loss: 0.5585 - accuracy: 0.6653
Epoch 63/100
```

LogisticRegression model

```
logr=LogisticRegression(solver ='liblinear')
logr.fit(x_train,y_train)
pred_train=logr.predict(x_train)
pred_train
pred_test=logr.predict(x_test)
pred_test
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,recall_score,f1_score
```

```
print("Train confusion matrix:\n",confusion_matrix(pred_train,y_train))
print("Train confusion matrix:\n",confusion_matrix(pred_test,y_test))
print("test accuracy:",accuracy_score(pred_test,y_test)*100)
```

test accuracy: 71.54882154882155

KNeighborsClassifier model

```
kn_model=KNeighborsClassifier(n_neighbors=3)
kn_model.fit(x_train,y_train)
kn_score=kn_model.score(x_test,y_test)
model_accuracy['Knn']=kn_score*100
kn_score*100
```

accuracy: 82.49158249158249

RandomForestClassifier model

```
ran_model=RandomForestClassifier(n_estimators=5)
ran_model.fit(x_train,y_train)
ran_score=ran_model.score(x_test,y_test)
model_accuracy['RanForest']=ran_score*100
ran_score*100
```

accuracy: *88.04713804713805*

AdaBoostClassifier model

```
ada_model = AdaBoostClassifier()
ada_model.fit(x_train,y_train)
ada_score=ada_model.score(x_test,y_test)
model_accuracy['AdaBoost']=ada_score*100
ada_score*100
```

accuracy: *87.54208754208754*

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

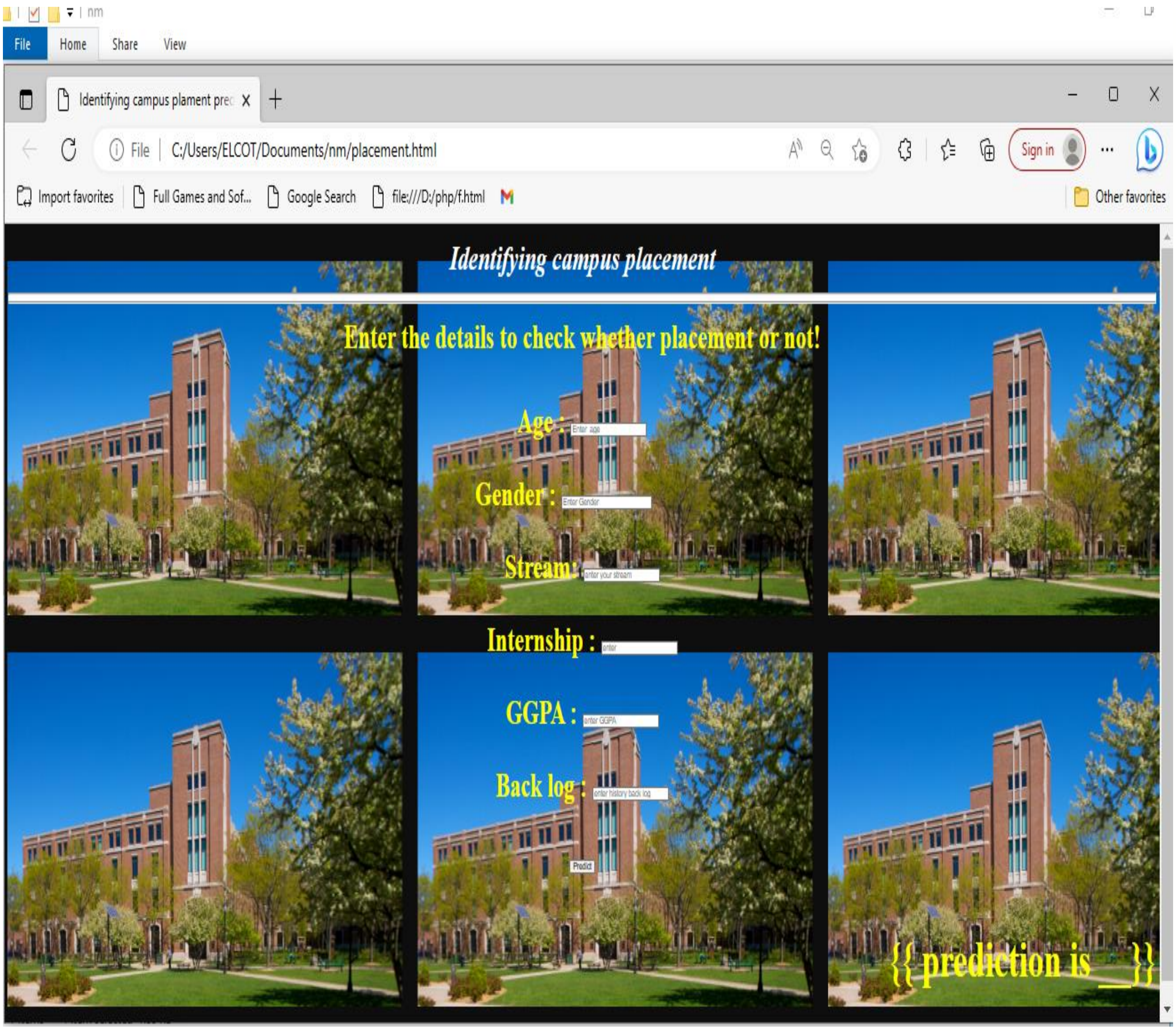
```
<!DOCTYPE html>
<html>
<head>
<title> Identifying campus plament predition</title>
</head>
<body background="sh.png" height="1"width="5" >
<body bgcolor='pink' text='red'>

<h1>
<b>
<i>
<font size="950px" color=" white">
<center> Identifying campus placement</center>
</font></i>
```

```
<hr></div>
<center><h2><font size="150px" color="yellow"> Enter the details to check whether placement or not!</h2></center>
<h4>
<form action="{{url_for('predict')}}" method="post">

<p> Age : <input type='text' name='age' placeholder='Enter  age' Enter Numerical part required='required' />
          <p> Gender : <input type='text' name='Gender' size= "25" placeholder='Enter Gender' Enter 0 for Male 1 for Female required='required' />
         
           <p>   Stream: <input type='text' name='stream' placeholder='enter your stream' required='required' /></p>
             <p>       Internship : <input type='text' name='internship' placeholder='enter' required='required' /></p>
              <p   >GGPA   : <input type='text' name='GGPA ' placeholder='enter GGPA ' required='required' /></p>
              <p> Back log : <input type='text' name='back log' placeholder='enter history back log' required='required' /></p></center>

      <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>
</h4>
<h2>
<b>
<p align="right">{{ prediction is __}}</p>
</b>
</h2>
</body>
</html>
```


Identifying campus plament pred
C:/Users/ELCOT/Documents/nm/placement.html
Identifying campus placement
Enter the details to check whether placement or not!
Age :
Gender :
Stream:
Internship :
GGPA :
Back log :
Predict
{{ prediction is __}}

4.ADVANTAGES & DISADVANTAGES

Advantages of Campus Recruitment

- The companies will be benefited from getting wide choice of candidates to select for different job posts. Companies can select the right and talented candidate from a vast pool of young applicants within a limited time. On the other hand, students have the advantage of getting a good job according to their qualification level even before the completion of their academic course in college.
- Campus recruitment helps in saving time and efforts of the companies. The entire campus recruitment process from a college is not a tedious toil. It prevents the occurrence of unusual expenditures related to recruitment process such as advertisement, initial screening, and final selection procedures etc. This in turn turns to be useful in reduced manpower effort and time as well.
- An organization through effective campus recruitment finds an opportunity to establish a link with the next batch of students. This in turn paves way to serve the future and long term recruitment needs of the company. Students participating in internships and summer training programs may have direct recruitment to different job positions offered by the company.
- Campus recruitment helps in increased selection ratio. More number of quality candidates can be selected through this recruitment process.
- The organizations can built up more company loyalty through campus selection process. Fresh and talented graduates will work more closely with their first company. Hence, this in a way will increase the brand loyalty among different applicants.

Disadvantages of Campus Recruitment

Campus recruitment is an expensive affair for majority of the companies as it adds up costs to the bottom line. Companies incur different expenses related to travel, boarding, training etc while conducting campus selection process. The experienced and skilled candidates having practical job exposures cannot be recruited through campus placements. Fresh candidates selected through campus placements require adequate training for work. This is an additional expense for the company. Also, students can’t work with their dream company and will have to remain satisfied with the company that recruits them during campus selection.

5.APPLICATION

- This system will reduce the chaos caused at the end ofthe final year. Students will start improving themselves from second year itself about their career awareness learning new skills throughout their graduation course
- . This system will help them to achieve their dream company as well as they will learn how to overcome their weaknesses. Students will be clear about their career growth and what various options are available in the market and how far they can improve themselves .Also with the result generated from the proposed system college placement cell will be well aware of what new skills can introduced by upbringing new training sessions in the college so that maximum student can get benefited out of the training .

- Also suggestions of new recruitments and the company criteria and requirement of skills to be known will be messaged to the students at very early stage .
- This system covers all the aspects for increasing the placement in undergraduate students.This system will also helpful for development of college as new project skills will be created by student and percentage of placement will be increased overall.
- Predicting the placement of a student gives an idea to the Placement Office as well as the student on where they stand. Not all companies look for similar talents. If the strengths and weaknesses of the students are identified it would benefit the student in getting placed.
- The placement Office can work on identifying the weaknesses of the students and take measures of improvement so that the students can overcome the weakness and perform to the best of their abilities.
- Thus the key lies in assessing the capabilities of the student in the right areas and subjecting them to the right training.

6.CONCLUSION

- The campus placement activity is incredibly a lot of vital as institution point of view as well as student point of view. In this regard to improve the student's performance, a work has been analyzed and predicted
- All said and done when approchrd correctly campus recuirment may be an unrivalled source of talent. By connecting with applicants online developing an attractive brand and hiring a diverse team campus recruitment can significantly help companies and and application.
- Using the least digital practices and technologies will futher enhance the process and ensure that it is a mutually beneficial relationship
- We have develop the mechine learning model using python programing language and the report are shown above.
- This system is helpful for institutions to predict student's campus placement. This system would help reduce tedious job of manual placement system. The placement officer can work on identifying the weaknesses of each students and can suggest improvements so that the students can overcome the weakness and perform to the be of their abilities.

7.FUTURE SCOPE

- It would of great help if we revise and update our curriculum and other extra activities for each semester in accordance with public, private and government sector requirement. We can also predict which company picks which hcategory of students. Make a list of skill a particular company looking for, then on the basis of that we can train our student. These traits will make prediction process more accurate.
- We can use more optimized algorithms for better predictions. We can also integrate online courses and services or students to improve their skills and knowledge. The system can also be used to predict the suitable courses for higher studies
- Students will be alerted through sms on cell and through mail about their progress and how they can achieve it during course of time. Also students who carried a backlog in their result will be given sms alerts about differentoptions regarding different companies and what skills to be achieved to fulfill the company criteria . This process will be carried out throughout their engineering course and different suggestions and options will be suggested after every result so that students remain focused about getting placed in campus selection.
- Also colleges can opt for different technical and language skill development at a very early stage seeing the scenario of the companies demand where students can learn not only academic concepts but also communication skill and behavioural skills to enhance their performance during the interview period. Finally students feedback will be taken for creating the new academic google form and how these proposed system helped them to get placed in companies
- Predicting the placement of a student gives an idea to the Placement Office as well as the student on where they stand. Not all companies look for similar talents. If strengths and weaknesses of the students are identified it would benefit the student in getting placed.

8.APPENDIX

A. source code

```
import numpy as np
import pandas as pd
import os
import seaborn as sns

import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')


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
from google.colab import files
uploads=files.upload()
df=pd.read_csv('/content/campus placement.csv')
df.head()
#checking data type
df.info()
#checking null values
df.isnull().sum()
df['Stream'].unique()
#creating new column
df['CGPA_']=['1-8' if x<=5 else "1-3" if x>5 and x<=6 else '7+' for x in df['CGPA']]
df.head()
#Removing Hostel_column
df=df.drop(['Hostel'],axis=1
df=df.drop(['CGPA_'],axis=1)
df
#finding the shape of data
df.shape
#finding null values
df.isnull().any()
#creating dummy dataframe for categorical values
df_cat=df.select_dtypes(include='int')
df_cat.head()
```

```
#descriptive analysis
df.describe(include='all')
#handling categorical  values
df=df.replace(['Computer Science','Information Technology','Electronics And Communicatio
n','Mechanical','Electrical','Civil'],
[0,1,2,3,4,5])
df['Gender'] = df['Gender'].replace({'Female':0, 'Male':1})
df.head()
#descriptive analysis
df.describe(include='all')
#data type
df.info()
df.isnull().sum()
df.isnull().any()
plt.figure(figsize=(10,6),dpi=100)
sns.heatmap(df.corr(),vmax=0.9,annot=True)
#check data distribution

sns.displot(df['CGPA'],color='red')
#handling outliers
sns.boxplot(df['Age'])
#finding the count of outliers
#IQR = q3-q1      upperbound=q3+(1.5*IQR), lower bound=q1-(1.5*IQR)
q1 = np.quantile(df['Age'],0.25)
q3 = np.quantile(df['Age'],0.75)
print('Q1 = {}'.format(q1))
print('Q3 = {}'.format(q3))
IQR=q3-q1
print('IQR value is{}'.format(IQR))
upperBound=q3+(1.5*IQR)
lowerBound=q1-(1.5*IQR)
print('the upper bound value is {} & the lower bound value is {}'.format(upperBound,lowerB
ound))
#skwed data
print ('skwed data:',len(df[df['Age']>upperBound]))
#handling outline

from scipy import stats
plt.figure(figsize=(12,4))
plt.subplot(1,3,2)
sns.displot(df['Age'])
plt.subplot(1,3,1)
stats.probplot(np.log(df['Age']),plot=plt)
plt.subplot(1,3,3)
sns.displot(np.log(df['Age']))
def transformationplot(feature):
   plt.figure(figsize=(12,5))
   plt.subplot(1,2,1)
```

```
    sns.distplot(feature)
transformationplot(np.log(df['CGPA']))
#exploratory data Analysis
plt.figure(figsize=(12,5))
sns.distplot(2,2,2)
sns.distplot(df['Age'],color='blue')
sns.distplot(3,3,3)
sns.distplot(df['PlacedOrNot'],color='red')
plt.figure(figsize=(12,4))
for i,j in enumerate(df_cat):
  plt.subplot(1,7,i+1)
  sns.countplot(df[j])
plt.figure(figsize=(12,6))
sns.histplot(data=df,x='Age',color='purple',kde=True,bins=12,legend=True)
#univariate anaiysis
plt.figure(figsize=(12,5))
plt.subplot(1,5,2)
sns.distplot(df['CGPA'],color='Green')
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.distplot(df['PlacedOrNot'],color='r')



plt.figure(figsize=(12,4))
plt.subplot(1,4,1)

sns.countplot(df['Gender'],color='r')
plt.subplot(1,4,2)

sns.countplot(df['Age'],color='g')
plt.show()
labels=df["Stream"].value_counts().index
sizes=df["Stream"].value_counts()
colors=['red','blue','green','violet',"grey","yellow"]
plt.figure(figsize=(12,12))
plt.pie(sizes,labels=labels,rotatelabels=False,autopct='%1.1f%%',colors=colors,shadow=True
,startangle=30)
plt.title("Stream level in piechart",color='r',fontsize=16)
plt.show()
plt.figure(figsize=(12,4))
plt.subplot(131)
sns.countplot(df['CGPA'],y=df['Age'],color='red')
plt.subplot(132)
sns.countplot(df['PlacedOrNot'],y=df['Age'],color='green')
plt.subplot(133)
sns.countplot(df['Internships'],y=df['Age'],color='blue')
plt.figure(figsize=(12,4))
sns.countplot(x=df["Stream"],hue=df["PlacedOrNot"], palette="colorblind")
```

```
plt.xlabel("Stream in countplot")
plt.ylabel("PlacedOrNot")
plt.show()
sns.swarmplot(x=df['Age'],y=df['CGPA'],hue=df['PlacedOrNot'])
#column define
data=df.drop('PlacedOrNot',axis=1)
column=[column for column in data.columns if  df[column].dtype!='PlacedOrNot']
column
x=df.drop('PlacedOrNot',axis=1
#Splitting the data into train and test
y=df['PlacedOrNot']
y
#scaling data
names=data[column]
sc=StandardScaler()
x_bal=sc.fit_transform(x)
x_bal=pd.DataFrame(x_bal,columns=data.columns)
#independent variables
x=df.iloc[:,0:6]
x.head()
#split& train test data
from   sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=45)

print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
```

svm model

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier,RandomForest
Classifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

model_accuracy={}
cv=KFold(n_splits=15,random_state=13,shuffle=True)

from sklearn.svm import SVC

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)
```

knn model

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

Artificial neural network 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)
```

LogisticRegression model

```
logr=LogisticRegression(solver ='liblinear')
logr.fit(x_train,y_train)
pred_train=logr.predict(x_train)
pred_train
pred_test=logr.predict(x_test)
pred_test
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,recall_score,f1
_score
print("Train confusion matrix:\n",confusion_matrix(pred_train,y_train))
print("Train confusion matrix:\n",confusion_matrix(pred_test,y_test))
print("test accuracy:",accuracy_score(pred_test,y_test)*100)
```

test accuracy: 71.54882154882155

KNeighborsClassifier model

```
kn_model=KNeighborsClassifier(n_neighbors=3)
kn_model.fit(x_train,y_train)
kn_score=kn_model.score(x_test,y_test)
model_accuracy['Knn']=kn_score*100
kn_score*100
```

accuracy: 82.49158249158249

RandomForestClassifier model

```
ran_model=RandomForestClassifier(n_estimators=5)
ran_model.fit(x_train,y_train)
ran_score=ran_model.score(x_test,y_test)
```

```
model_accuracy['RanForest']=ran_score*100
ran_score*100
```

accuracy: *88.04713804713805*

AdaBoostClassifier model

```
ada_model = AdaBoostClassifier()
ada_model.fit(x_train,y_train)
ada_score=ada_model.score(x_test,y_test)
model_accuracy['AdaBoost']=ada_score*100
ada_score*100
```

accuracy: *87.54208754208754*

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

template

placement. html

```
<!DOCTYPE html>
<html>
<head>
<title> Identifying campus plament predition</title>
</head>
<body background="sh.png" height="1"width="5" >
<body bgcolor='pink' text='red'>

<h1>
<b>
<i>
<font size="950px" color=" white">
<center> Identifying campus placement</center>
</font></i>
```

```
<hr></div>
<center><h2><font size="150px" color="yellow"> Enter the details to check whether
placement or not!</h2></center>
<h4>
<form action="{{url_for('predict')}}" method="post">

<p> Age : <input type='text' name='age' placeholder='Enter  age' Enter Numerical part
required='required' />
        <p> Gender : <input type='text' name='Gender' size= "25" placeholder='Enter Gender'
Enter 0 for Male 1 for Female required='required' />
         
         <p>   Stream: <input type='text' name='stream' placeholder='enter your stream'
required='required' /></p>
          <p>      Internship : <input type='text' name='internship' placeholder='enter'
required='required' /></p>
           <p   >GGPA  : <input type='text' name='GGPA ' placeholder='enter GGPA '
required='required' /></p>
           <p> Back log : <input type='text' name='back log' placeholder='enter history back
log' required='required' /></p></center>

     <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>
</h4>
<h2>
<b>
<p align="right">{{ prediction is __}}</p>
</b>
</h2>
</body>
</html>
```

app.py file

```
import flask
from flask import Flask, render_template, request
import pickle
import numpy as np
import sklearn
from flask_ngrok import run_with_ngrok
import warnings
```

```
warnings.filterwarnings('ignore')

app = Flask(__name__)
run_with_ngrok(app)

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


@app.route('/', methods=['GET'])
def home():
    return render_template('index.html')


@app.route('/', methods=['GET', "POST"])
def predict():
    input_values = [float(x) for x in request.form.values()]
    inp_features = [input_values]
    print(inp_features )
    prediction = model.predict(inp_features)
    if prediction == 1:
        return render_template('index.html', prediction_text='Eligible to job, you will be placed
in campus plcement')
    else:
        return render_template('index.html', prediction_text='Not eligible to placement')

app.run()
```