Intelligent Admissions: Identifying Patterns and Trends in Campus Placement Data Using Machine Learning

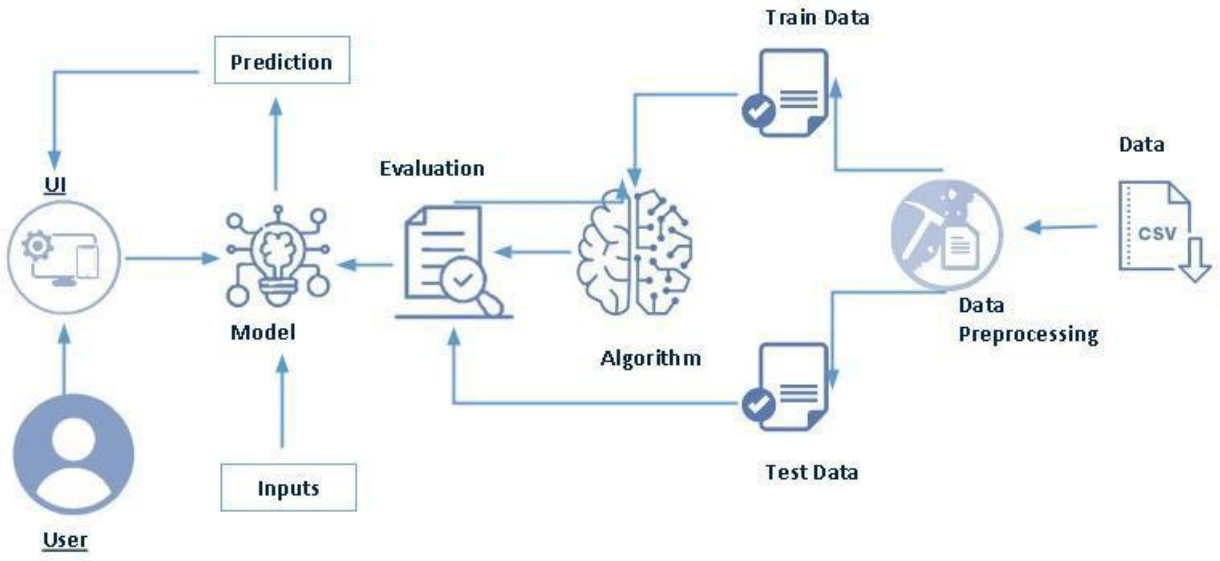
Overview:

Machine learning is a method of data analysis that automates analytical model building. These models help you to make a trend analysis of university placements data, to predict a placement rate for the students of an upcoming year which will help the university to analyze the performance during placements. Many students look at universities as a means of investment which can help them make a great future by getting placed in good companies and which will relieve their stress and unease from their lives before graduating from the university. The trend will also help in giving the companies reasons as to why they should visit university again and again.

Some attributes the very important role while analyzing the student for e.g. Student's name, Department, Company, Location and Annual package.

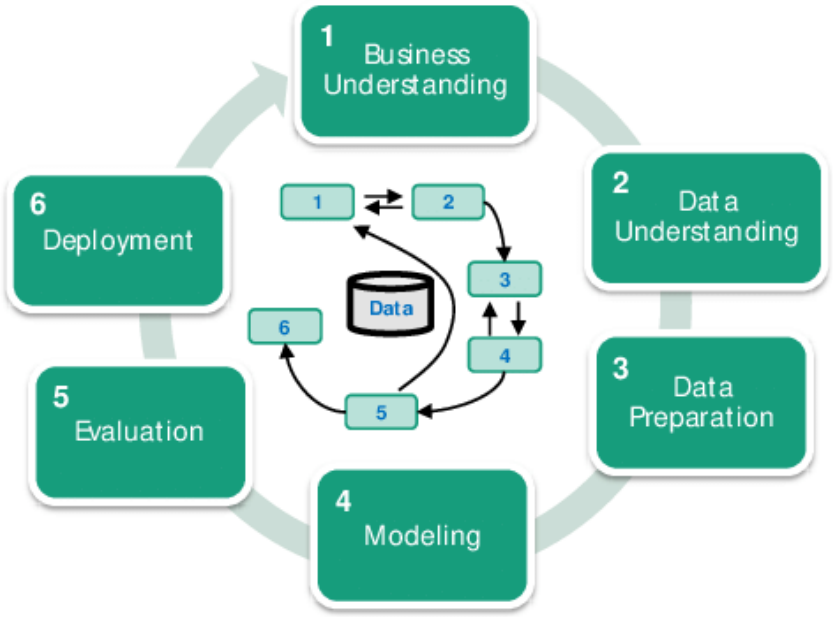
So, classification can help you to classify those data and clustering helps to make the clusters department wise. In this paper we have used neural networks to predict the upcoming student placement and got 77% of accuracy while testing Were iteration are 1000. Through extensive trend analysis of varies complex data collected from different sources, we can demonstrate that our analysis can provide a good pragmatic solution for future placement of students

Technical Architecture :



Technical about the project:

A technical project is one that includes engineering elements, technical hardware components, software elements, and or data management requirements. Actually technical component management seems to be a part of almost every project.



A Project Description:

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.

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**
  - o Collect the dataset or create the dataset
- **Visualizing and analyzing data**
  - o Univariate analysis
  - o 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

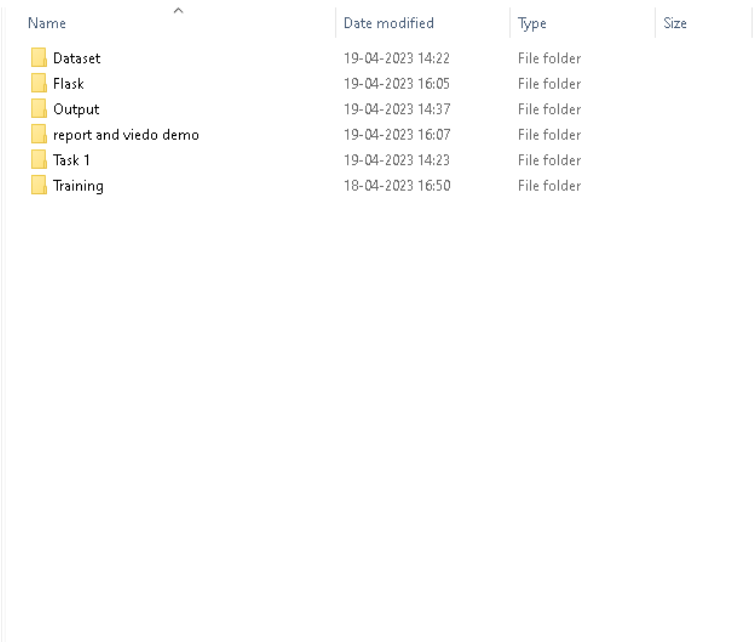
- Import the model building libraries
- Initializing the model
- Training and testing the model
- Evaluating performance of model
- Save the model

• 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.

1.2 PURPOSE:

Importance of Pattern Recognition in Machine Learning It helps in the classification of unseen data. It makes suitable predictions using learning techniques. It recognizes and identifies an object at varying distances. It not only helps in the prediction of the unseen data but also helps in making useful suggestion.

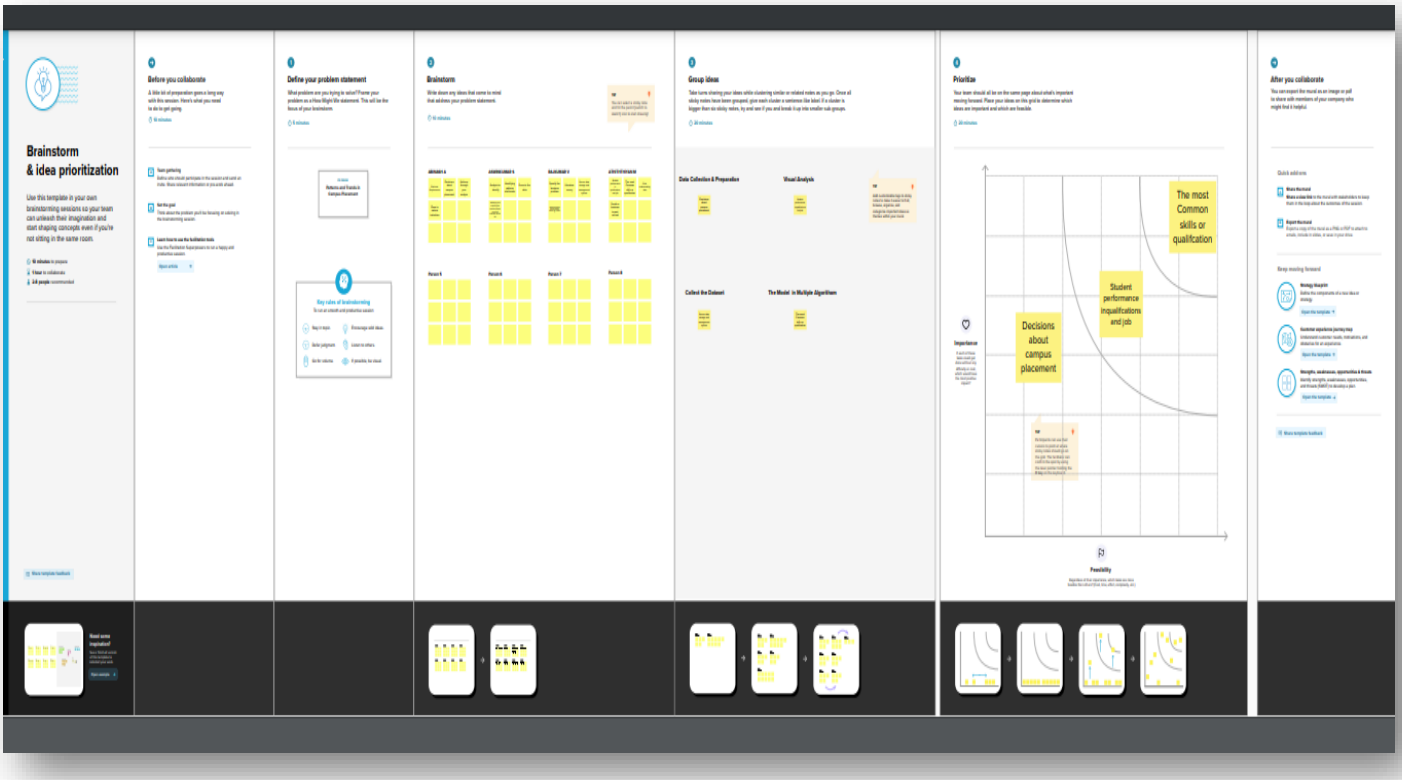
2. PROBLEM DEFINITION AND DESIGN THINKING

The campus placement process can be time consuming and overwhelming for both students and recruiters. Students often have to apply to multiple companies and attend various rounds of interviews, while recruiters have to sort through a

large pool of applicants to find the right fit for their organization. This process can be made more efficient through the use of machine learning algorithms that can match the skills and qualifications of students with the requirements of recruiters.

2.1EMPATHY MAP

2.2 IDEATION AND BRAINSTORMING MAP:



Result:

FILL THE DETAILS

Age

Gender M(0),F(0)

Stream CS(0),IT(1),ECE(2)

Internships

CGPA

Number of backlogs

Submit

4. ADVANTAGE AND DISADVANTAGE:

Advantage:

1.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.

2 .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.

3. Improving accuracy: By identifying patterns in data that humans may not be able to see, machine learning can drastically improve the accuracy of its predictions.Itcanalsocreatemodelsthatsimulatedifferentdecisionscenarios andhelpidentifythebestcourseofaction.Andasnewdatabecomesavailable, machine learning can be used to constantly update and refine decision models.

Disadvantages:

1. Data Acquisition

The whole concept of machine learning is about identifying useful data. The outcome will be incorrect if a credible data source is not provided. The quality of the data is also significant. If the user or institution needs more quality data, wait for it. It will cause delays in providing the output. So, machine learning significantly depends on the data and its quality.

2. Time and Resources

The data that machines process remains huge in quantity and differs greatly. Machines require time so that their algorithm can adjust to the environment and learn it. Trials runs are held to check the accuracy and

reliability of the machine. It requires massive and expensive resources and high-quality expertise to set up that quality of infrastructure. Trials runs are costly as they would cost in terms of time and expenses.

3. Results Interpretations

One of the biggest advantages of Machine learning is that interpreted data that we get from that cannot be hundred percent accurate. It will have some degree of inaccuracy. For a high degree of accuracy, algorithms should be developed so that they give reliable results.

4. High Error Chances

The error committed during the initial stages is huge, and if not corrected at that time, it creates havoc. Biasness and wrongness have to be dealt with separately; they are not interconnected. Machine learning depends on two factors, **i.e., data and algorithm**. All the errors are dependent on the two variables. Any incorrectness in any variables would have huge repercussions on the output.

5. APPLICATION:

1. Image Recognition

One of the most notable machine learning applications is image recognition, which is a method for cataloging and detecting an object or feature in a digital image. In addition, this technique is used for further analysis, such as pattern recognition, face detection, and face recognition.

2. Speech Recognition

ML software can make measurements of words spoken using a collection of numbers that represent the speech signal. Popular applications that employ speech recognition include Amazon's Alexa, Apple's and Google Maps.

3. Predict Traffic Patterns

To explain this, let's consider the example of Google maps. When we enter our location on the map, the application collects massive amounts of data about the present traffic to generate predictions regarding the upcoming traffic and identify the fastest route to our destination.

4. E-commerce Product Recommendations

One of the prominent elements of typically any e-commerce website is product recommendation, which involves the sophisticated use of machine learning algorithms. Websites track customer behavior based on past purchases, browsing habits, and cart history and then recommend products using machine learning AI.

5. Catching Malware

The process of using machine learning (ML) to detect malware consists of two basic stages. First, analyzing suspicious activities in an Android environment to generate a suitable collection of features; second, training the system to use the machine and deep learning (DL) techniques on the generated features to detect future cyber attacks in such environments.

6. Virtual Personal Assistant

Virtual personal assistants help people access relevant information via text or voice. When a query is put into the system, the personal assistant gathers information by searching for it or recalling similar questions an individual has asked in the past. Some popular ML techniques involved in virtual assistants include speech recognition, speech-to-text conversion, natural language processing, and text-to-speech conversion.

6. CONCLUSION:

Machine learning approaches applied in systematic reviews of complex research fields such as quality improvement may assist in the title and abstract inclusion screening process. Machine learning approaches are of particular interest considering steadily increasing search outputs and accessibility of the existing evidence is a particular challenge of the research field quality improvement. Increased reviewer agreement appeared to be associated with improved predictive performance.

This tutorial has introduced you to Machine Learning. Now, you know that Machine Learning is a technique of training machines to perform the activities a human brain can do, albeit bit faster and better than an average human-being. Today we have seen that the machines can beat human champions in games such as Chess, Alpha GO, which are considered very complex. You have seen that machines can be trained to perform human activities in several areas and can aid humans in living better lives.

Machine Learning can be a Supervised or Unsupervised. If you have lesser amount of data and clearly labeled data for training, opt for Supervised Learning. Unsupervised Learning would generally give better performance and results for large data sets. If you have a huge data set easily available, go for deep learning techniques. You also have learned Reinforcement Learning and Deep Reinforcement Learning. You now know what Neural Networks are, their applications and limitations.

Finally, when it comes to the development of machine learning models of your own, you looked at the choices of various development languages, IDEs and Platforms. Next thing that you need to do is start learning and practicing each machine learning technique. The subject is vast, it means that there is width, but if you consider the depth, each topic can be learned in a few hours. Each topic is independent of each other. You need to take into consideration one topic at a

time, learn it, practice it and implement the algorithm/s in it using a language choice of yours. This is the best way to start studying Machine Learning. Practicing one topic at a time, very soon you would acquire the width that is eventually required of a Machine Learning expert.

7. FUTURE SCOPE:

Future scope of Machine Learning (ML) and is high, as it provides the machine a capability to acquire knowledge, which makes it a more human like machine. Machine learning is currently in use, maybe in so many domains as one might imagine. Overall, this article will provide a basic overview of the timeline, types of machine learning, and present industrial applications of this technology.

8. APPENDIX

Source Code:

```
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.metrics  import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib
from sklearn.metrics import accuracy_score
```

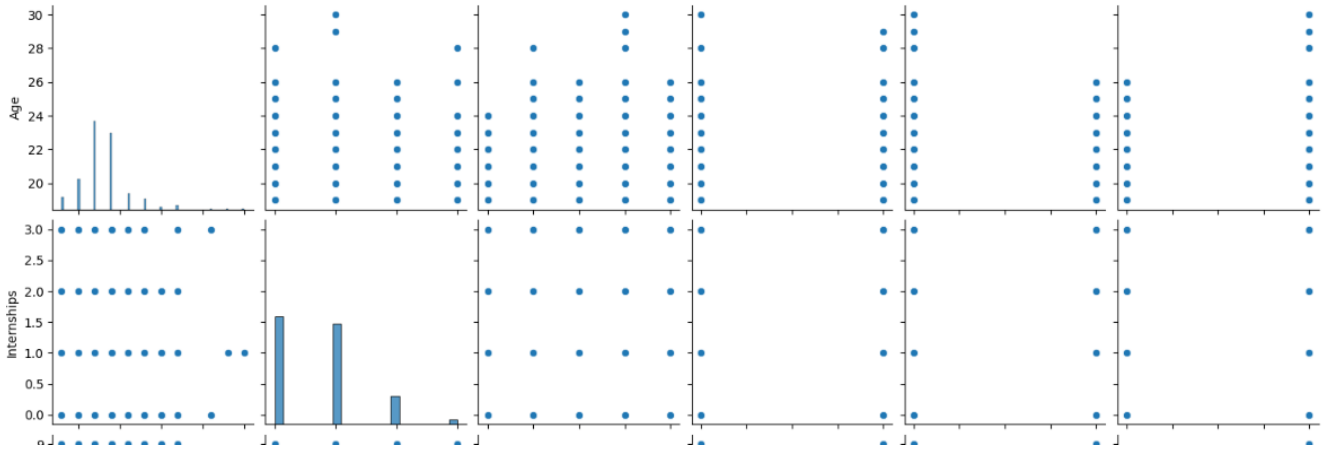
```
[ ] df = pd.read_csv("/content/collegePlace.csv")
```

```
[ ] df.shape
```

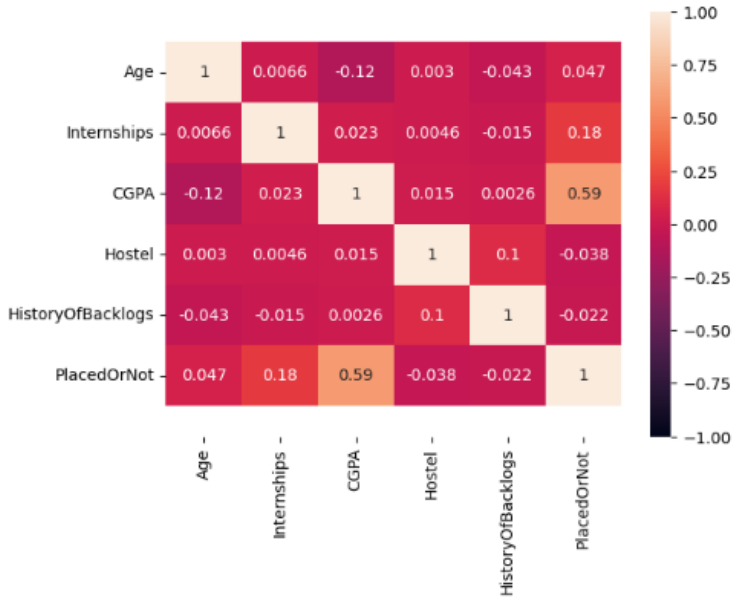
```
(2966, 8)
```

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f9e37bb1fa0>
```



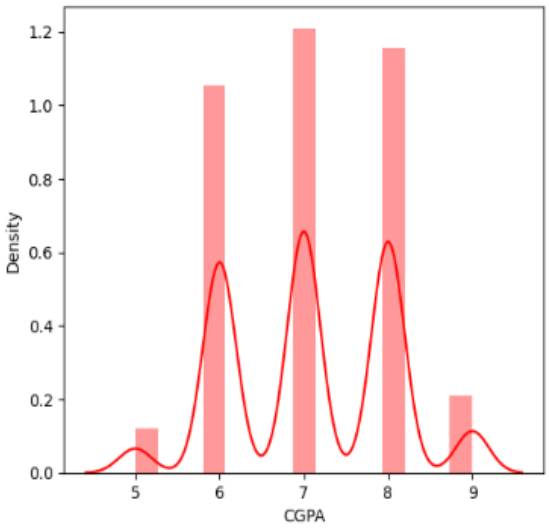

```
[ ] ax = sns.heatmap(corr,vmin= -1, vmax = 1, annot = True)
    bottom, top = ax.get_ylim()
    ax .set_ylim(bottom + 0.5, top - 0.5)
    plt.show()
    corr
```



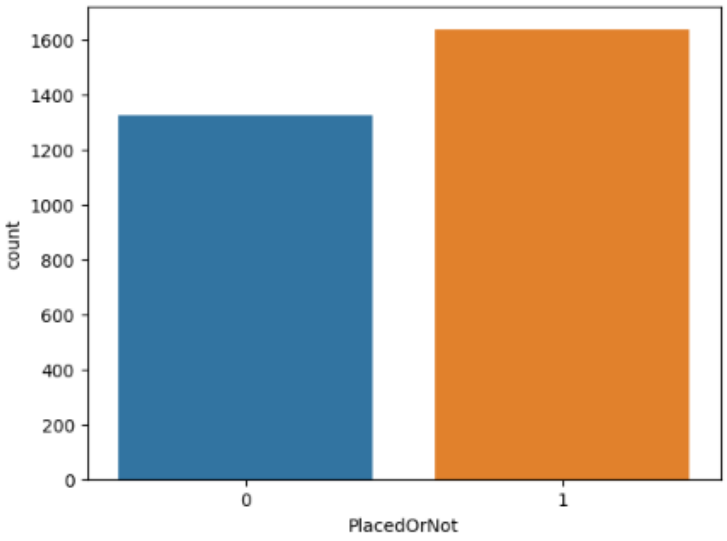
	Age	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
Age	1.000000	0.006552	-0.119787	0.003042	-0.042586	0.046943
Internships	0.006552	1.000000	0.023496	0.004617	-0.015118	0.179334

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

```
<ipython-input-9-f92659182652>:3: UserWarning:

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

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

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

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




```
from seaborn.widgets import color_palette
# setting the different color palette
color_palette = sns.color_palette("BuGn_r")
sns.countplot(x = "PlacedOrNot", data = df)
plt.show()
#as we can see data is not that imbalanced
```



```
[ ] 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
```

```
[ ] df.describe()
```

	Age	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
count	2966.000000	2966.000000	2966.000000	2966.000000	2966.000000	2966.000000
mean	21.485840	0.703641	7.073837	0.269049	0.192178	0.552596
std	1.324933	0.740197	0.967748	0.443540	0.394079	0.497310
min	19.000000	0.000000	5.000000	0.000000	0.000000	0.000000
25%	21.000000	0.000000	6.000000	0.000000	0.000000	0.000000
50%	21.000000	1.000000	7.000000	0.000000	0.000000	1.000000
75%	22.000000	1.000000	8.000000	1.000000	0.000000	1.000000
max	30.000000	3.000000	9.000000	1.000000	1.000000	1.000000

```
[ ] df['Gender'].value_counts()
```

```
Male      2475
Female     491
Name: Gender, dtype: int64
```

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

```
Computer Science                776
Information Technology          691
Electronics And Communication   424
Mechanical                      424
Electrical                      334
Civil                           317
Name: Stream, dtype: int64
```

```
[ ] 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
```

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

2966 rows × 8 columns

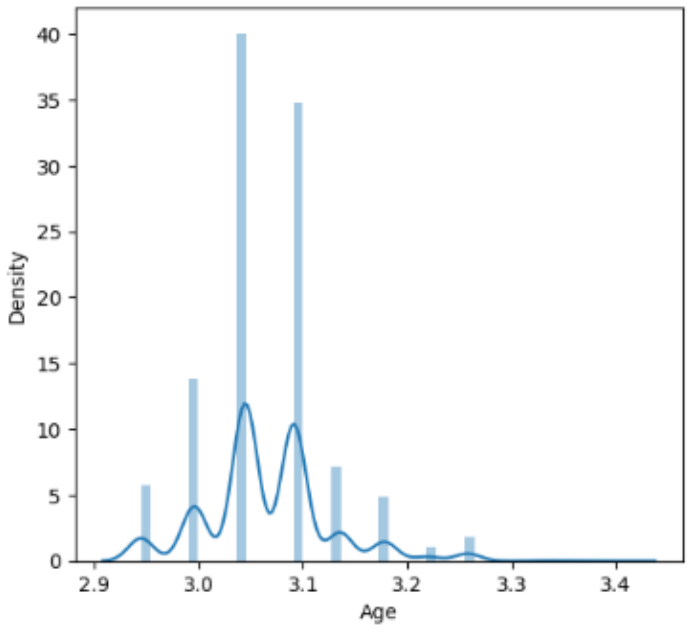
```
[ ] df.info()
```

```
<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   int64
 2   Stream             2966 non-null   int64
 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(8)
memory usage: 185.5 KB
```

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

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

```
<ipython-input-42-4e023cd5df85>:4: UserWarning:

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

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

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

  sns.distplot(feature)
```



```
[ ] # we are dropping the 'Hostel'column as it won't have much effect on our final output
    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

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

```
[ ] print(x)
```

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

[2966 rows x 6 columns]
```

```
print(y)
```

```
0       1
1       1
2       1
3       1
4       1
       ..
2961    0
2962    0
2963    0
2964    0
2965    1
Name: PlacedOrNot, Length: 2966, dtype: int64
```

```
[ ] scaler = StandardScaler()
```

```
[ ] scaler.fit(x)
```

```
StandardScaler
StandardScaler()
```

```
[ ] Standardized_data = scaler.transform(x)
```

```
[ ] print(Standardized_data)
```

```
[[ 0.38813058 -0.44540301  0.04008175  0.40044544  0.95719068  2.05024603]
 [-0.36675158  2.24515772 -1.14874288 -0.95077319 -0.07631043  2.05024603]
 [ 0.38813058  2.24515772 -0.55433057  0.40044544 -1.10981154 -0.48774634]
 ...
 [ 0.38813058 -0.44540301 -0.55433057  0.40044544 -0.07631043 -0.48774634]
 [ 0.38813058 -0.44540301 -1.14874288  0.40044544 -0.07631043 -0.48774634]
 [ 1.14301273 -0.44540301  1.82331869 -0.95077319  0.95719068 -0.48774634]]
```

```
[ ] 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)
```

```
[ ] print(X.shape, X_train.shape, X_test.shape)
```

```
(2966, 6) (2372, 6) (594, 6)
```

```
[ ] classifier = svm.SVC(kernel='linear')
```

```
[ ] classifier.fit(X_train,y_train)
```

```
SVC
SVC(kernel='linear')
```

```
[ ] #testing accuracy
    x_test_prediction = classifier.predict(x_test)
    y_pred= accuracy_score(x_test_prediction, y_test)
    y_pred
```

```
0.7794612794612794
```

```
[ ] x_test
```

```
array([[-0.36675158,  2.24515772,  0.04008175,  1.75166407, -0.07631043,
        -0.48774634],
       [ 0.38813058, -0.44540301,  0.63449406, -0.95077319, -0.07631043,
        -0.48774634],
       [ 1.89789488, -0.44540301, -0.55433057,  0.40044544, -0.07631043,
        -0.48774634],
       ...,
       [-1.12163373, -0.44540301,  0.63449406,  0.40044544,  1.99069179,
        -0.48774634],
       [ 0.38813058, -0.44540301, -1.14874288,  0.40044544, -1.10981154,
        -0.48774634],
       [ 0.38813058,  2.24515772, -0.55433057,  0.40044544,  0.95719068,
        -0.48774634]])
```

```
[ ] #training accuracy
    x_train_prediction = classifier.predict(x_train)
    training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

```
[ ] print('Accurancy score of the training data:',training_data_accuracy)
```

```
Accurancy score of the training data: 0.7685497470489039
```

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    from sklearn import metrics
    from sklearn.model_selection import cross_val_score
```

```
[ ] 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("---Result---\nk: {}".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)
```

```
---Result---
k: {'Regular': 7}
```

```
[ ] knn_pred
```

```
array([1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1])
```

```
[ ] print('Accurancy score of the test data using KNN :', testd)
```

```
Accurancy score of the test data using KNN : 0.8619528619528619
```

```
[ ] print('Accurancy score of the test data using KNN :', testd)
```

```
Accurancy score of the test data using KNN : 0.8619528619528619
```

```
[ ] #checking the train accurancy
    knn_pred_1 = knn.predict(x_train)
    traind = accuracy_score(knn_pred_1, y_train)
    traind
```

```
0.8882799325463744
```

```
[ ] knn_pred_1
```

```
array([1, 0, 0, ..., 1, 1, 1])
```

```
[ ] x_train.shape
```

```
(2372, 6)
```

```
[ ] y_train.shape
```

```
(2372,)
```

```
[ ] 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))
    #find 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)
```

```
Epoch 1/100
119/119 [==============================] - 1s 2ms/step - loss: 0.8801 - accuracy: 0.4970
Epoch 2/100
119/119 [==============================] - 0s 2ms/step - loss: 0.7641 - accuracy: 0.5308
Epoch 3/100
119/119 [==============================] - 0s 2ms/step - loss: 0.7195 - accuracy: 0.5662
Epoch 4/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6946 - accuracy: 0.5898
Epoch 5/100
119/119 [==============================] - 0s 2ms/step - loss: 0.6632 - accuracy: 0.6172
Epoch 6/100
119/119 [==============================] - 0s 3ms/step - loss: 0.6495 - accuracy: 0.6315
Epoch 7/100
119/119 [==============================] - 0s 3ms/step - loss: 0.6311 - accuracy: 0.6400
Epoch 8/100
119/119 [==============================] - 0s 3ms/step - loss: 0.6222 - accuracy: 0.6509
Epoch 9/100
119/119 [==============================] - 0s 3ms/step - loss: 0.6098 - accuracy: 0.6560
Epoch 10/100
119/119 [==============================] - 0s 3ms/step - loss: 0.5901 - accuracy: 0.6610
Epoch 11/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5761 - accuracy: 0.6868
Epoch 12/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5614 - accuracy: 0.6859
Epoch 13/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5412 - accuracy: 0.6977
Epoch 14/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5398 - accuracy: 0.7045
Epoch 15/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5460 - accuracy: 0.6863
Epoch 16/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5424 - accuracy: 0.7045
Epoch 17/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5318 - accuracy: 0.6969
Epoch 18/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5084 - accuracy: 0.7192
Epoch 19/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5195 - accuracy: 0.7133
Epoch 20/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5046 - accuracy: 0.7184
Epoch 21/100
119/119 [==============================] - 0s 2ms/step - loss: 0.5117 - accuracy: 0.7011
Epoch 22/100
```

```
# with the model fitted, we test the model on test data.use a threshold of 0.5,to turn data into True(Placed) and false(NotPlaced)
pred = classifier.predict(x_test)
pred = (pred > 0.5)
pred
```

```
19/19 [==============================] - 0s 1ms/step
array([[ True],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
```

```
[ ] from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, pred)
    cm
```

```
array([[236,  30],
       [ 68, 260]])
```

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

```
[ ] #input_data = (0.0,60.0,0.0,11.0,67349.0,0.0,0.0,0.0,0.0,1.0,3.0,44.0.1,14230.0,11.0,1.0,4461.0,2.0,0.0,3.0,0.0,18.0,1.0)
    input_data = [[22,0,2,1,8,1]]
    '''# changing the input_data to numpy array
    input_data_as_numpy_array = np.asarray(input_data)
    #reshape the array as we are predicting for one instance
    input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
    #standardize the input data
    std_data = scalar.trandform(input_data_reshaped)
    print(std_data)'''
    prediction = knn.predict(input_data)
    print(prediction)
    if (prediction[0] == 0):
      print('not placed')
    else:
        print('placed')
```

```
[1]
placed
```