Identifying Patterns and Trends in Campus Placement Data using Machine Learning

Abstract

Placement of students is one of the most important objective of an educational institution. Reputation and yearly admissions of an institution invariably depend on the placements it provides it students with. That is why all the institutions, arduously, strive to strengthen their placement department so as to improve their institution on a whole. Any assistance in this particular area will have a positive impact on an institution’s ability to place its students. This will always be helpful to both the students, as well as the institution. In this study, the objective is to analysis previous year's student's data and use it to predict the placement chance of the current students. This model is proposed with an algorithm to predict the same. Data pertaining to the study were collected from the same institution for which the placement prediction is done, and also suitable data pre-processing methods were applied. This proposed model is also compared with other traditional classification algorithms such as Decision tree and Random forest with respect to accuracy, precision and recall. From the results obtained it is found that the proposed algorithm performs significantly better in comparison with the other algorithms mentioned. Keywords: Classification, Decision tree, Random forest

**Introduction**

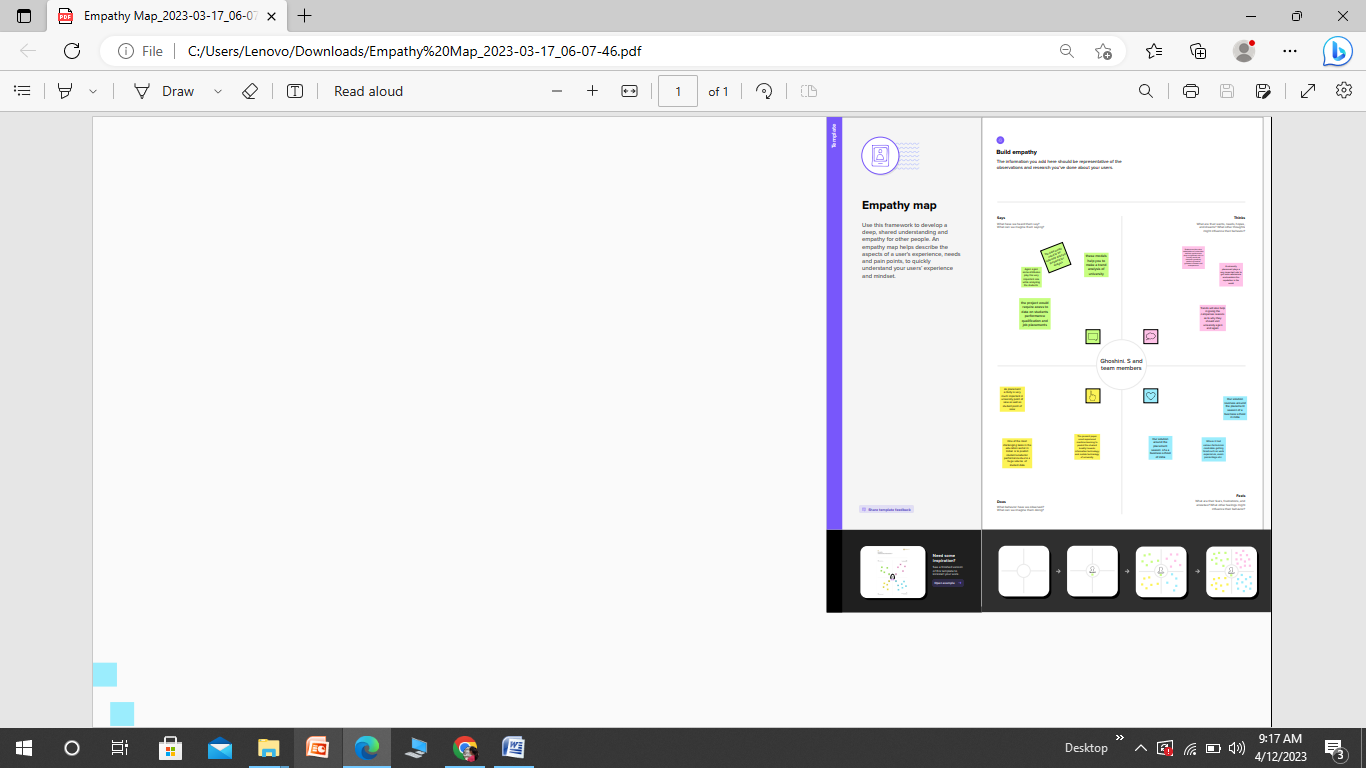
Campus recruitment is a strategy for sourcing, engaging and hiring young talent for internship and entry-level positions. College recruiting is typically a tactic for medium- to large-sized companies with high-volume recruiting needs, but can range from small efforts (like working with university career 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 graduate .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 format. We will be doing flask integration and IBM deployment.

**Purpose**

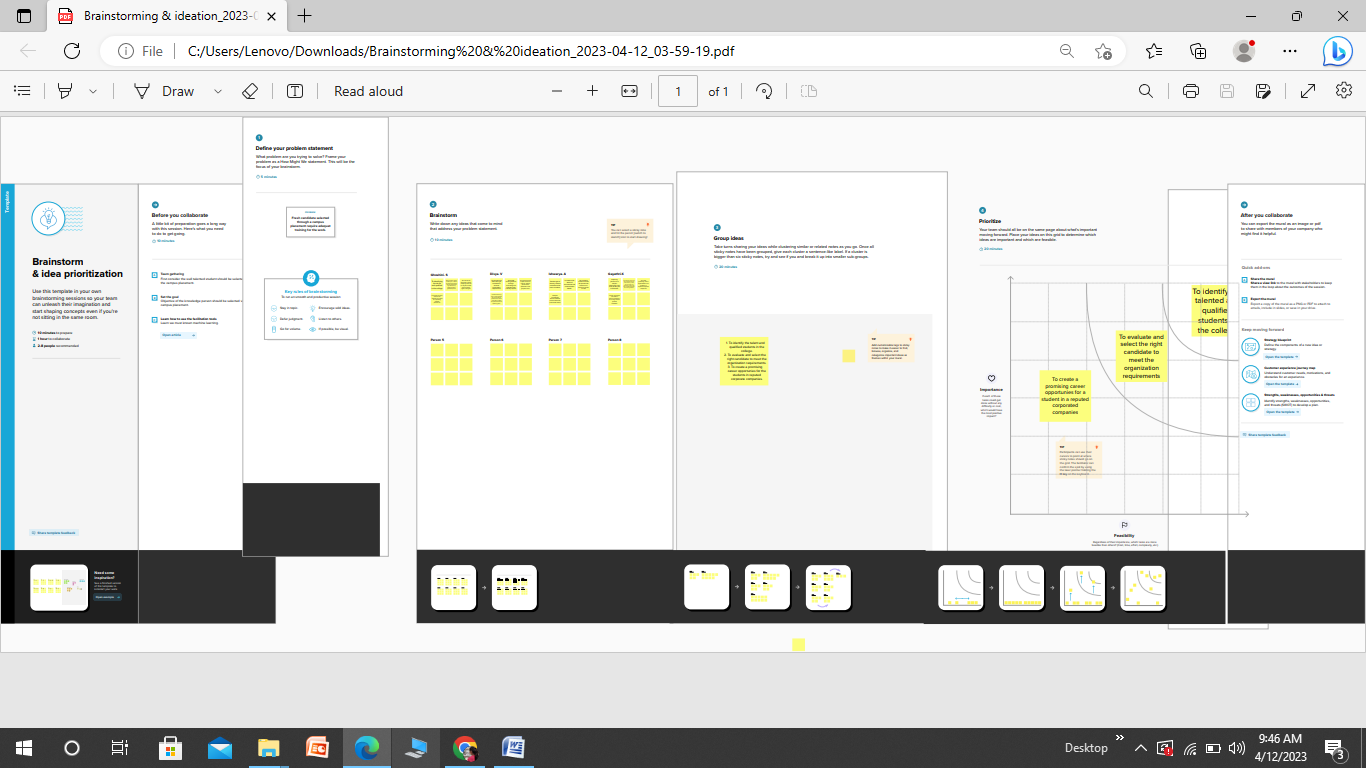
Campus placement or campus recruiting is a program conducted within universities or other educational institutions **to provide jobs to students nearing completion of their studies**. In this type of program, the educational institutions partner with corporations who wish to recruit from the student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

Problem Definition & Design Thinking

Empathy Map



Ideation & Brainstorming Map



**Result**

The algorithms of machine learning we have discussed are can used to find the trend of placement, which will be helpful for university to get more admission in future

**Advantages & Disadvantages**

Advantages

Campus placements **smooth the overall process of getting that first job**. Without it, looking for a job as a fresher is like going into the wilderness unprepared. If you a student from a reputable institution. You will benefit from the institution's reputation, which increases your marketability to employers.

Disadvantages

**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.

**Applications**

Campus placement or campus recruiting 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.

**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 using the classification algorithms Decision Tree and the Random forest algorithm to validate the approaches. The algorithms are applied on the data set and attributes used to build the model. The accuracy obtained after analysis for Decision tree is 84% and for the Random Forest is 86%. Hence, from the above said analysis and prediction it better if the Random Forest algorithm is used to predict the placement results.

**Future scope**

Campus placement is considered as an institutional obligation and merit. The ranking of the institutions is based on the number of students placed successfully and the average salary offered. Due to the ongoing conditions the campus recruitment and placement landscape has been altered and has gone through various changes and challenges. There are institutes who have had to regretfully informed students of revoked job offer and are still facing a tough time to get companies onboard for further placements. The paper talks about the recruitment / placement scenario in general for both graduate and post graduates in India based on the primary research work done. Data was collected from 39 companies and analyzed to understand the placement scenario for the students who will undergo the drive in 2021. The paper also tries to understand the recruitment drive pattern for various companies’ whether online will now be routine or are the companies still willing to come to campus for the drive. It analyzes the campus recruitment landscape for future purposes in contrast with the current happenings.

**Appendix**

SOURCE CODE

Milestone 1:

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

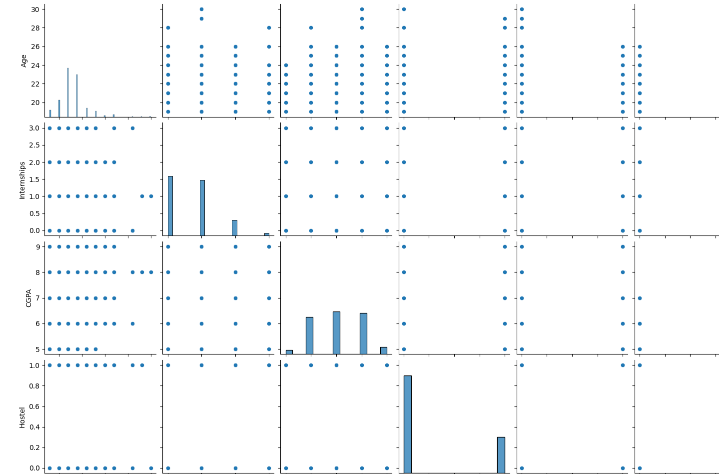
Read the data set

df = pd.read\_csv(r"/content/collegePlace.csv")

df.shape

(2966, 8)

sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x7fb5fd67f9a0>

corr = df.corr()

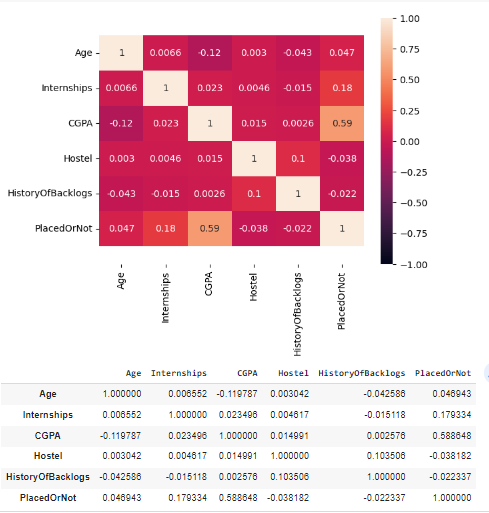
ax = sns.heatmap(corr, vmin = -1, vmax = 1, annot = True)

bottom, top = ax.get\_ylim()

ax.set\_ylim(bottom + 0.5, top - 0.5)

plt.show()

corr



plt.figure(figsize=(12,5))

plt.subplot(121)

sns.distplot(df['CGPA'],color='r')

<ipython-input-80-f92659182652>:3: UserWarning:

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

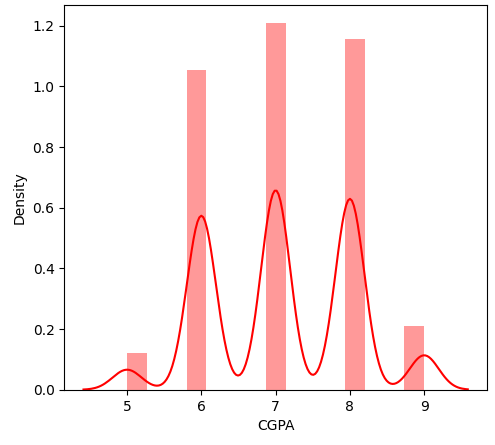
Please adapt your code to use either `displot` (a figure-level function with

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'>

plt.figure(figsize=(12,5))

plt.subplot(121)

sns.distplot(df['PlacedOrNot'],color='r')

<ipython-input-81-5e468beb8a0d>:3: UserWarning:

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

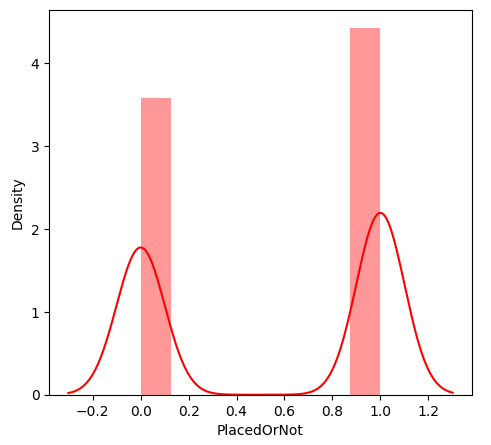
Please adapt your code to use either `displot` (a figure-level function with

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['PlacedOrNot'],color='r')

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

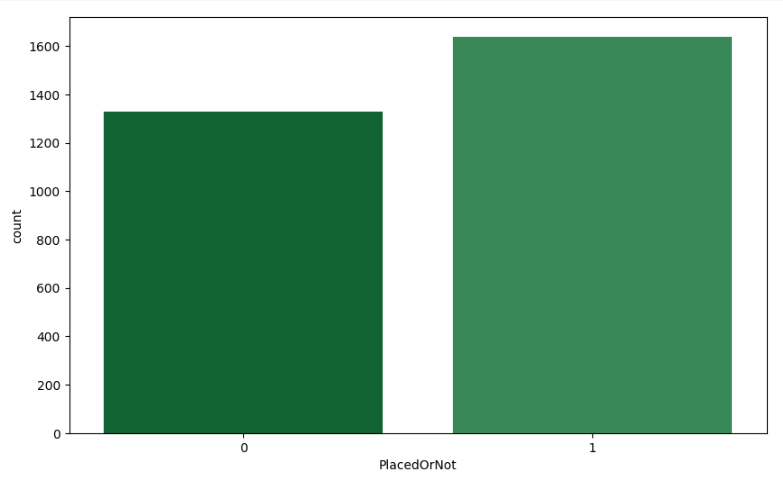
plt.figure(figsize = (10, 6), dpi = 100)

color\_palette = sns.color\_palette("BuGn\_r")

sns.set\_palette(color\_palette)

sns.countplot(x = "PlacedOrNot", data = df)

plt.show()



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

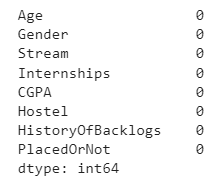
6 HistoryOfBacklogs 2966 non-null int64

7 PlacedOrNot 2966 non-null int64

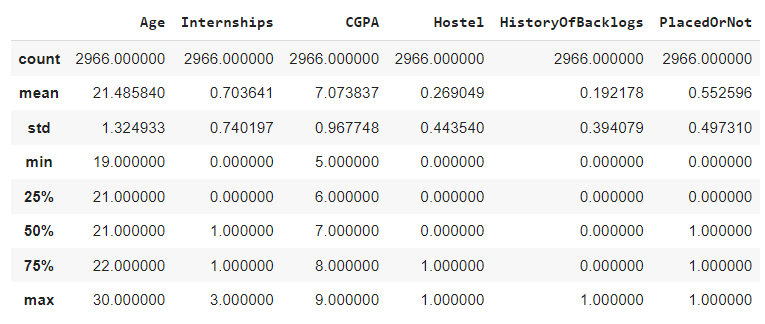
dtypes: int64(6), object(2)

memory usage: 185.5+ KB

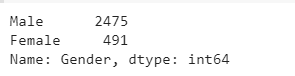
df.isnull().sum()



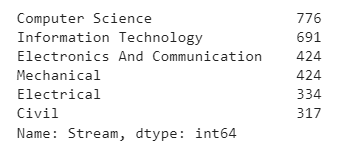
df.describe()



df['Gender'].value\_counts()



df['Stream'].value\_counts()



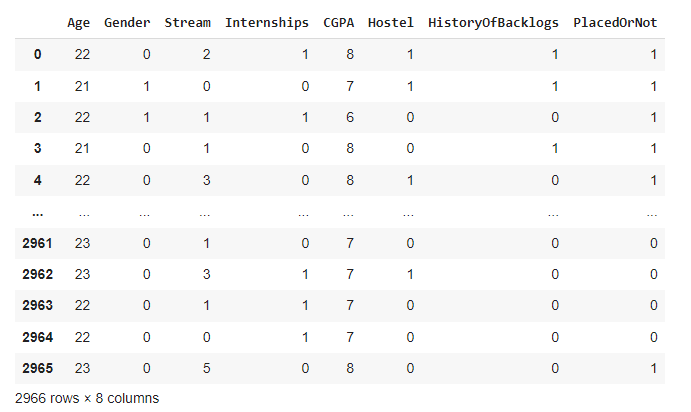
df = df.replace(['Male'], [0])

df = df.replace(['Female'], [1])

df = df.replace(['Computer Science', 'Information Technology','Electronics And Communication','Mechanical','Electrical','Civil'],

[0,1,2,3,4,5])

df



df.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

# Milestone 3:

def transformationplot(feature):

  plt.figure(figsize=(12,5))

  plt.subplot(1,2,2)

  sns.distplot(feature)

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

<ipython-input-92-572031f9942d>:4: UserWarning:

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

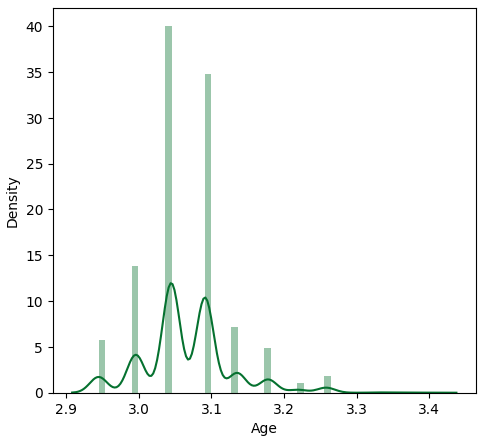
Please adapt your code to use either `displot` (a figure-level function with

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)



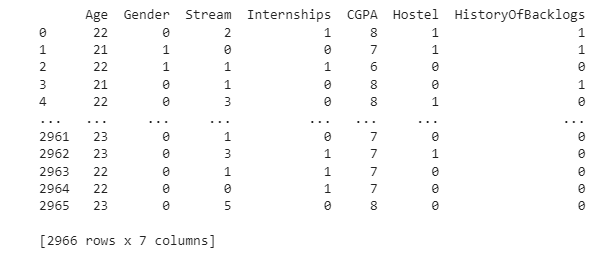
X = df.drop(columns = 'PlacedOrNot', axis=1)

Y = df['PlacedOrNot']

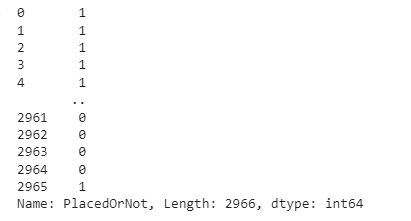
import joblib

joblib.dump(X,"placement")

['placement']

print(X) 

print(Y)



scaler = StandardScaler()

scaler.fit(X)

C:\Users\Lenovo\Pictures\Screenshots\Screenshot (411).png

standardized\_data = scaler.transform(X)

print(standardized\_data)

[[ 0.38813058 -0.44540301 0.04008175 ... 0.95719068 1.64826939

2.05024603]

[-0.36675158 2.24515772 -1.14874288 ... -0.07631043 1.64826939

2.05024603]

[ 0.38813058 2.24515772 -0.55433057 ... -1.10981154 -0.60669694

-0.48774634]

...

[ 0.38813058 -0.44540301 -0.55433057 ... -0.07631043 -0.60669694

-0.48774634]

[ 0.38813058 -0.44540301 -1.14874288 ... -0.07631043 -0.60669694

-0.48774634]

[ 1.14301273 -0.44540301 1.82331869 ... 0.95719068 -0.60669694

-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, 7) (2372, 7) (594, 7)

# Milestone 4:

classifier = svm.SVC(kernel='linear')

classifier.fit(X\_train, Y\_train)

C:\Users\Lenovo\Pictures\Screenshots\Screenshot (413).png

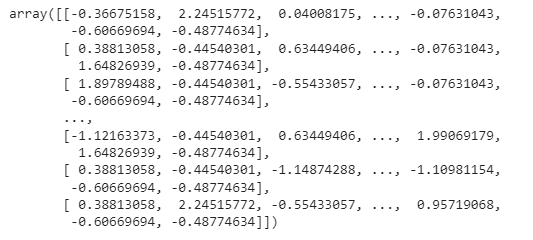
X\_test\_prediction = classifier.predict(X\_test)

y\_pred= accuracy\_score(X\_test\_prediction, Y\_test)

y\_pred

0.7794612794612794

X\_test



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 : 0.7715008431703204

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):

    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}

Score: {'Regular': 86.19528619528619}

knn\_pred

array([1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 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, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 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, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 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, 0, 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, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 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, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 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('Accuracy score of the test data using KNN : ', testd)

Accuracy score of the test data using KNN : 0.8619528619528619

knn\_pred\_1 = knn.predict(X\_train)

traind = accuracy\_score(knn\_pred\_1, Y\_train)

traind

0.8646711635750421

knn\_pred\_1

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

X\_train.shape

(2372, 7)

Y\_train.shape

(2372,)

import tensorflow as tf

from tensorflow import keras

from keras.models import Sequential

from tensorflow.keras import layers

from keras.optimizers import Adam

classifier = Sequential()

classifier.add(keras.layers.Dense(6,activation = 'relu', input\_dim = 6))

classifier.add(keras.layers.Dropout(0.50))

classifier.add(keras.layers.Dense(6,activation = 'relu'))

classifier.add(keras.layers.Dropout(0.50))

classifier.add(keras.layers.Dense(1, activation = 'sigmoid'))

loss\_1 = tf. keras.losses.BinaryCrossentropy()

classifier.compile(optimizer = 'Adam', loss = loss\_1 , metrics = ['accuracy'])

classifier.fit(X\_train, Y\_train, batch\_size = 10, epochs = 100)

pred = classifier.predict(X\_test)

pred = (pred > 0.5)

pred

from sklearn.matrics import confusion\_matrix

cm = confusion\_matrix(Y\_test, pred)

cm

# Milestone 5:

import pickle

pickle.dump(knn,open("placement.pkl",'wb'))

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

#input\_data = (0.0,60.0,0.0,11.0,67349.0,0.0,0.0,0.0,1.0,3.0,44.0.1,14230.0,11.0,1.0,4461.0,2.0,0.0,3.0,0.0,18.0,1.0)

input\_data = [[22,0,2,1,8,1]]

\*\*\*# changing the input\_data to numpy array

input\_data\_as\_numpy\_array = np.as array(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 = scaler.transform(input\_data\_reshaped)

print(std\_data)\*\*\*

prediction = knn.predict(input\_data)

print(prediction)

if (prediction[0] == 0):

  print(' not placed')

else:

  print('placed')

<section id="hero" class="d-flex-column justify-content-center">

  <div class="container">

    <div class="row justify-content-center">

      <div class="col-xl-8">

        <h1>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h1>

      </div>

    </div>

  </div>

</section>

<section id="about" class="about">

  <div class="container">

    <div class="section-title">

      <h2>Fill the details</h2>

    </div>

    <div class="row content">

      <div class="first">

        <form action="{{ url\_for('y\_predict)}}" method="POST">

            <input type="number" id="sen1" name="sen1" placeholder="Age">

            <input type="number" id="sen2" name="sen2" placeholder="Gender M(0),F(0)">

            <input type="number" id="sen3" name="sen3" placeholder="Stream cs(0),IT(1),ECE(2),Mech(3),EEE(4),Civil">

            <input type="number" id="sen4" name="sen4" placeholder="Internships">

            <input type="number" id="sen5" name="sen5" placeholder="CGPA">

            <input type="number" id="sen6" name="sen6" placeholder="Number of backlogs">

            <input type="submit" value="submit">

        </form>

      </div>

    </div>

  </div>

</section>

<section id="hero" class="d-flex flex-column justify-content-center">

  <div class="container">

    <div class="row justify-content-center">

      <div class="col-xl-8">

        <h1>The prediction is : {{y}}</h1>

        <h3> 0 represents Not-Placed </h3>

        <h3> 1 represents Placed<h2>

      </div>

    </div>

   </div>

</section>

from flask import Flask, render\_template , Request

app=Flask(\_\_name\_\_)

import pickle

import joblib

model=pickle.load(open("placement123.pkl",'rb'))

ct=joblib.load('placement')

@app.route('/')

def hello():

return render\_template("index.html")

@app.route('/guest' , methods = ["post"])

def Guest():

    sen1=request.form["sen1"]

    sen2=request.form["sen2"]

    sen3=request.form["sen3"]

    sen4=request.form["sen4"]

    sen5=request.form["sen5"]

    sen6=request.form["sen6"]

@app.route('/y\_predict' , methods = ["POST"])

def y\_predict():

    X\_test = [[(yo) for yo in request.form.values()]]

    prediction =model.predict(X\_test)

    prediction = prediction[0]

    return render\_template("secondpage.html",y=prediction)

app.run(debug=True)