Final Project Report

Project Title: Identifying Patterns and Trends in Campus Placement Data using Machine Learning
Project ID: SPS_PRO_4013
Project By: Vijaya Chandra Harshith Gamini, Avinash Yarabothu, L S N V Satish Pulleti

1. Introduction

1.1 Overview

Campus placement data includes details regarding students' academic standing, abilities, internship experiences, and placement outcomes. From this data in order to comprehend the variables affecting placement success and create plans for streamlining the placement procedure is very useful. By applying machine learning techniques we patterns and trends in campus placement data.

We have used various visualization techniques and in-built libraries in python and performed the exploratory data analysis to identify the pattern. Then we have implemented famous machine learning classification algorithms and based on our results we identified the important factors affecting the Placement trends.

1.2 Purpose

Use our project many educational institutions can be benifitted. They can customize their curriculum, career counselling, and placement techniques by understanding the variables that affect placement success. Higher placement rates, a better reputation, and greater student satisfaction can result from this. Institutions can make strategic decisions for budget allocation, program development, and industry alliances using the insights generated through EDA and ML, helping them to remain competitive and relevant.

2. Literature Survey

2.1 Existing Problem

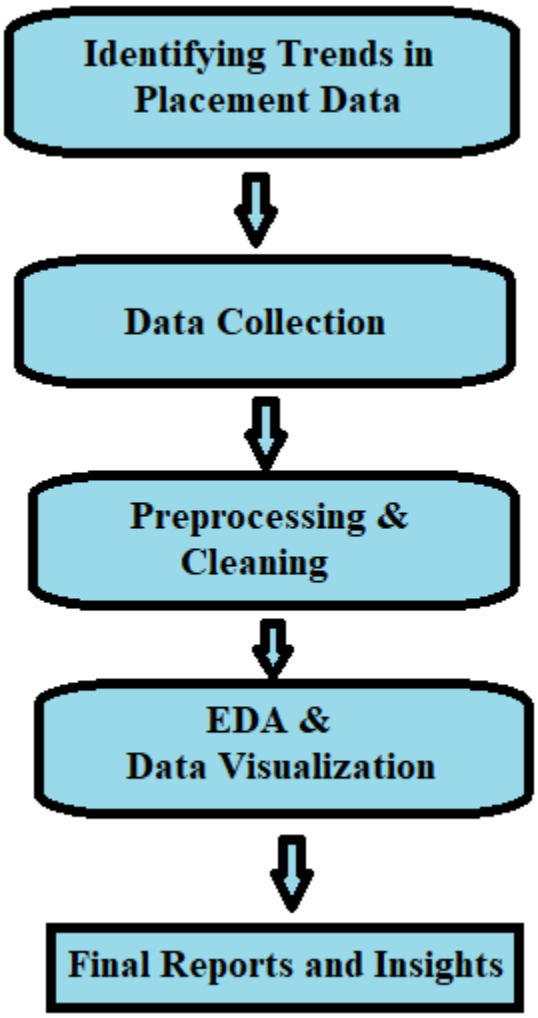
Many solutions are available which identify the trends in campus placements. But most of them just identify the patterns without considering the quality of the dataset they use. This leads to so many irregularities and wrong insights which may not be useful for everyone.

2.2 Proposed Solution

Our solution will stand out among these because of our unique approach to identifying the patterns. Because our approach doesn't involve direct identification of patterns. We are going to perform various correlation analysis techniques and then proceed to final insights based on the various visualization techniques like correlation matrix, heat maps, and scatter plots supported with Machine Learning models.

3. Theoritical Analysis

3.1 Block diagram



The block diagram depicts the various steps involved in building our solution.

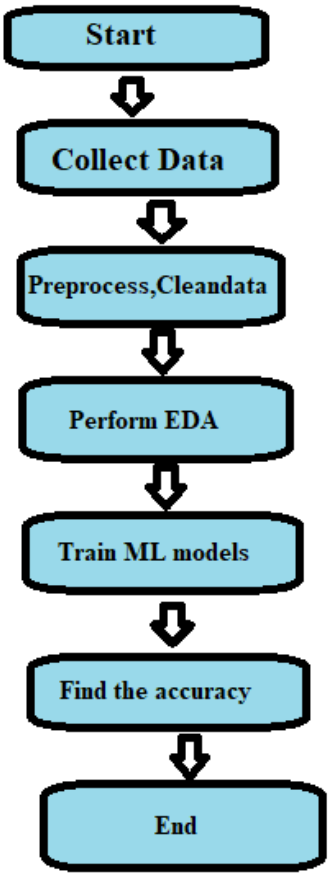
3.2 Hardware / Software Designing

For our solution there is no requirement of hardware components. Coming to software components, we need python libraries, Jupyter Notebooks, HTML, CSS, Python Flask, IBM Cloud, IBM Watson Studio for the implementation and deployement of our solution.

4. Experimental Investigations

During the implementation of our solution, we have performed many analytical tasks. We have performed cleaning of the dataset by eliminating duplicates, null values and then we have performed normalisation inorder to improve the accuracy and then we have converted the categorical data into numerical data by using the technique of label encoding. Finally we have performed the correlation analysis between the attributes of the dataset and we have visualised each and every attribute using distribution graphs, sactter plots and box plots. Finally we have visualised a Heat map between all the attributes.

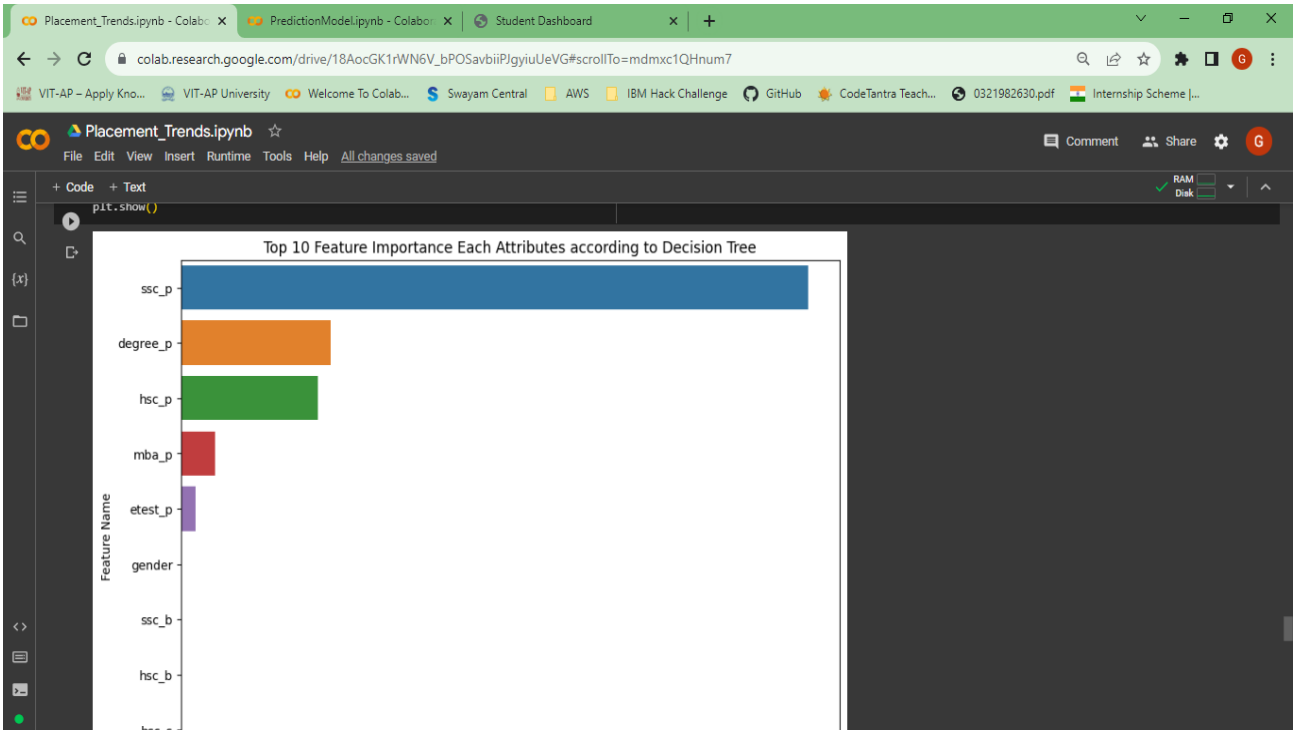
5. Flowchart



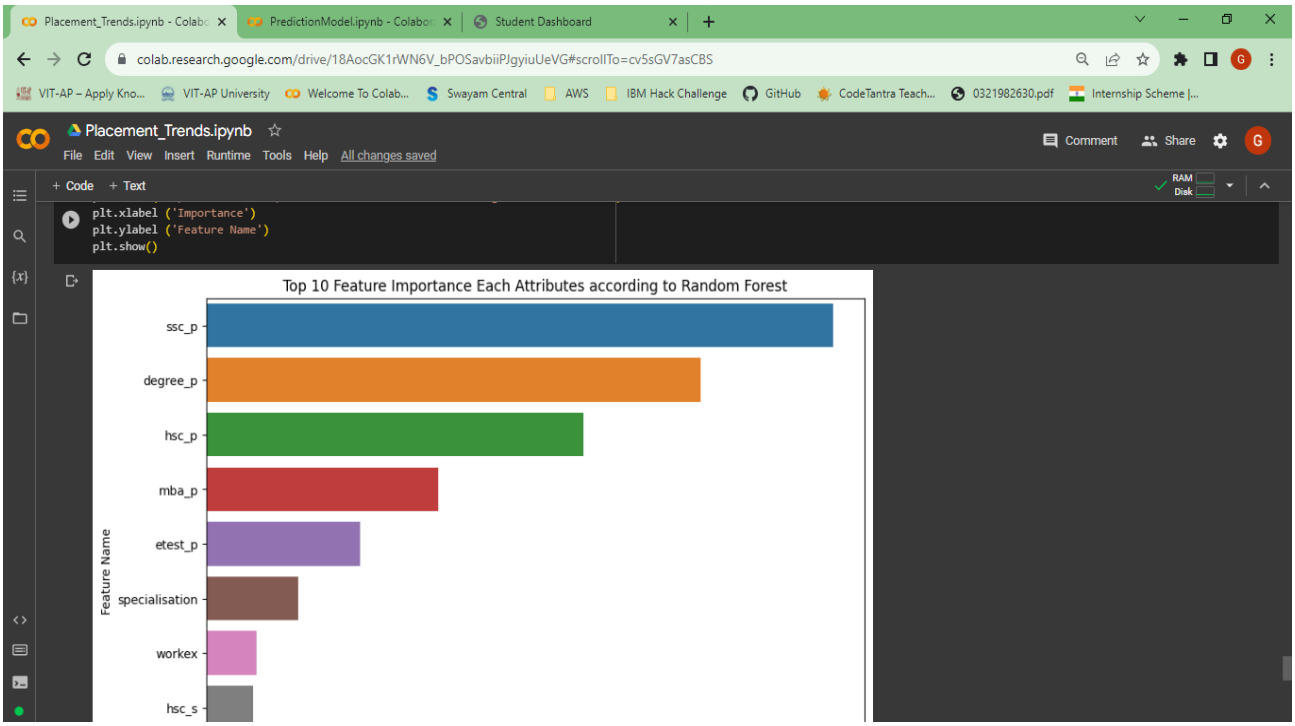
6. Result

We have observed various trends after completion of our project. We observed the relation between the various attributes in the dataset. It has been observed that the average mean and probability distribution for the Target attribute 'Placed' is comparatively greater than the attribute 'Not placed' in the case of every independent attribute. Coming to the trained Machine learning models, out of the three models Decision Tree, Random Forest and SVC classifiers, we have observed that the dataset has been sufficiently predicted with good accuracy score by the Random Forest Model.

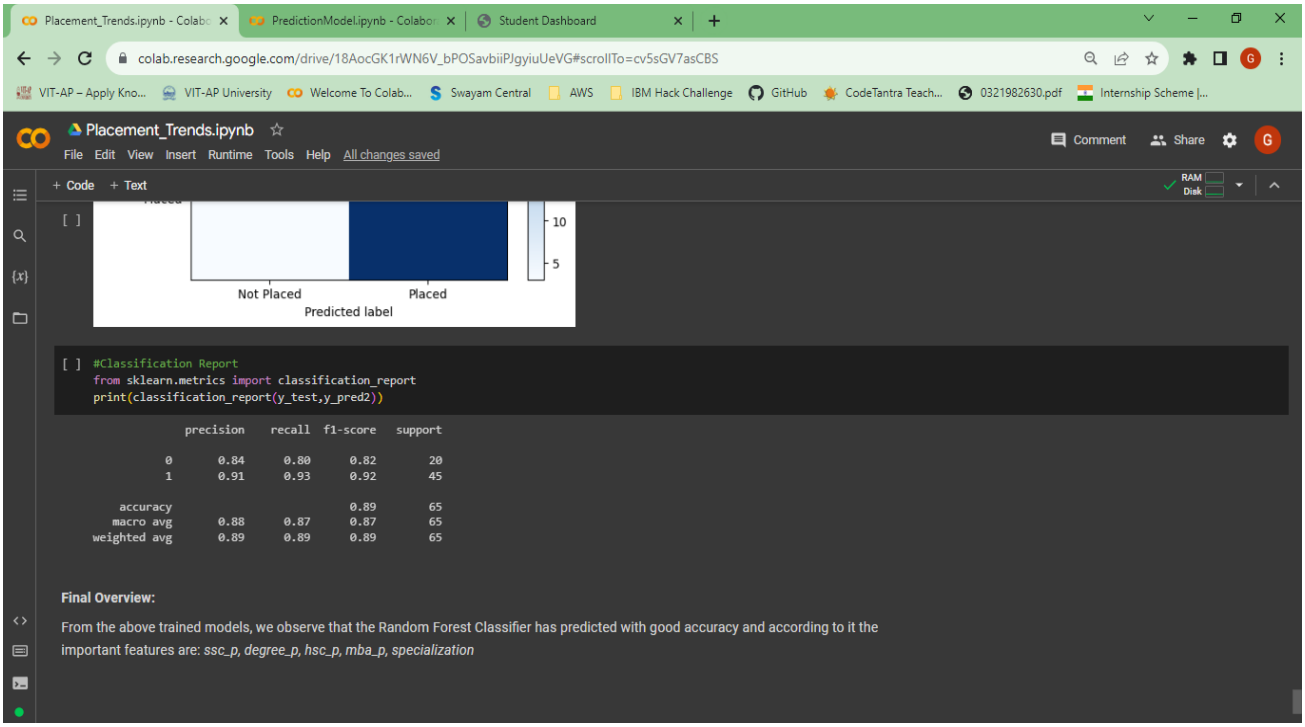
So, we have chosen the Random Forest model which will result in good prediction and according to the Random Forest Model, the important features of the dataset that contribute more in prediction of Target attribute are ***"ssc_p, hsc_p, degree_p, mba_p and the type of specialisation".*** These are our final conclusions.



Result of Decision Tree classifier



Result of Random Forest classifier



Final result based on Random Forest

7. Advantages and Disadvantages

For our developed solution, there are advantages as well as disadvantages.They are as follows:

Advantages:

- The important factors affecting placements can be easily know within fraction of seconds.
- The solution is deployed in IBM cloud, so it can be accessed by anyone for prediction at any time without any delay.
- The model is trained with good accuracy so it gives good results.
- The solution is very cost efficient as it involves softwares that are opensource and do not cost much.

Disadvantages:

- The placement status doesn't always depend on the marks scored by the student, it may sometimes depend on the skill based on different companies.
- Giving improper input data may be lead to wrong results.

8. Applications

Our solution has a wide range of applications in many fields. Colleges and universities can improve their curriculum by better aligning it with market demands by analyzing placement trends. Career counselors can use placement trends to give students individualized information about appropriate career pathways.

Businesses may forecast their future talent demands and make strategic hiring decisions by analyzing previous placement data. In order to improve employability, government organizations might use placement trends to inform education programs and efforts. Researchers can examine the long-term job outcomes of graduates from various universities or programs using placement data.

Overall, the solution's insights into placement trends can aid in closing the employment-education gap by giving students, organizations, and employers useful data to help them choose their educational and career routes.

9. Conclusion

In conclusion, the solution effectively combined Python and EDA methods to identify patterns in placement data, enabling students, educational institutions, and companies to make decisions based on solid facts. Finally, we reached a conclusion that the main factors affecting the placements are ***ssc_p, hsc_p, degree_p, mba_p and the type of specialisation.***

10. Future Scope

The project's scope is limited to the provided dataset, and external factors may influence placement outcomes. In future there is a possibility of developing a further more efficient solution that considers not only the academics of the student but also the skills and other activites performed which may result in better prediction as these factors are also considered by companies during hiring of employees.

11. Bibiliography

The dataset that has been used to develop our solution has been obtained through the Kaggle website. For training machine learning models, we have used the documentation available in internet.

- Reference link: https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement

APPENDIX

A. Source code

- Placement_Trends.ipynb

```
import numpy as np
import pandas as pd
df = pd.read_csv("PlacementData.csv")
df.head()
#sl_no column is not necessary, we drop it
df = df.drop(columns = ['sl_no'])
df.head()
df['salary'].isnull().sum()
#Salary column contains many null values, so we drop it
df = df.drop(columns = ['salary'])
df.head()
#Size of dataset
df.shape
df.info()
df.describe()
#Checking for Null values in dataset
df.isnull().sum()
#Checking for duplicated values
```

```
df.duplicated().value_counts()
#Target Attribute categories
df.status.value_counts()
#Visualizing the Target attribute
import matplotlib.pyplot as plt
import seaborn as sns
#Visualizing count
count = sns.countplot(x = df.status)
for i in count.containers:
  count.bar_label(i)
plt.title("Placement - Count")
plt.show()
#Visualizing percentage
values = df.status.value_counts()
plt.pie(values,labels=['Placed','Not Placed'],autopct="%1.2f%%")
plt.title("Placement - Percentage")
plt.show()
#Relation of each Numerical attribute with Target attribute
df.groupby(['status']).mean().round(3)
#Visualizing the pattern of SSC percentage(ssc_p) attribute with Target
attribute
sns.kdeplot(data=df, x='ssc_p', hue="status")
plt.title('ssc_p per Status')
plt.show()
#Visualizing the pattern of HighSchool percentage(hsc_p) attribute with
Target attribute
sns.kdeplot(data=df, x='hsc_p', hue="status", palette="Set1")
plt.title('hsc_p per Status')
plt.show()
#Visualizing the pattern of Degree percentage(degree_p) attribute with
Target attribute
sns.kdeplot(data=df, x='degree_p', hue="status", palette="Set2")
plt.title('degree_p per Status')
plt.show()
#Visualizing the pattern of EntranceTest percentage(etest_p) attribute
with Target attribute
```

```
sns.kdeplot(data=df, x='etest_p', hue="status")
plt.title('etest_p per Status')
plt.show()
#Visualizing the pattern of MBA percentage(mba_p) attribute with Target
attribute
sns.kdeplot(data=df, x='mba_p', hue="status", palette="Set1")
plt.title('mba_p per Status')
plt.show()
#Visualizing the pattern of Gender and etest_p attribute with Target
attribute
sns.boxplot(data=df, x="etest_p", y="gender", hue="status")
plt.title("gender with etest_p")
plt.show()
#Visualizing the pattern of ssc_b and ssc_p attribute with Target
attribute
sns.boxplot(data=df, x="ssc_p", y="ssc_b", hue="status", palette="Set1")
plt.title("ssc_b with ssc_p" , fontsize = 18)
plt.show()
#Visualizing the pattern of hsc_p and hsc_b attribute with Target
attribute
sns.boxplot(data=df, x="hsc_p", y="hsc_b", hue="status", palette="Set2")
plt.title("hsc_b with hsc_p")
plt.show()
#Visualizing the pattern of hsc_p and hsc_s attribute with Target
attribute
sns.boxplot(data=df, x="hsc_p", y="hsc_s", hue="status", palette="Set3")
plt.title("hsc_s with hsc_p")
plt.show()
#Visualizing the pattern of degree_p and degree_t attribute with Target
attribute
sns.boxplot(data=df, x="degree_p", y="degree_t", hue="status")
plt.title("degree_t with degree_p")
plt.show()
#Visualizing the pattern of degree_p and workex attribute with Target
attribute
sns.boxplot(data=df, x="degree_p", y="workex", hue="status",
```

```
palette="Set1")
plt.title("workex with degree_p")
plt.show()
#Visualizing the pattern of mba_p and specialisation attribute with Target
attribute
sns.boxplot(data=df, x="mba_p", y="specialisation", hue="status",
palette="Set2")
plt.title("specialisation with mba_p")
plt.show()
sns.pairplot(df,hue='status')
plt.show()
sns.heatmap(df.corr(),annot=True)
X = df.drop(columns=['status'])
X.head()
Y = pd.DataFrame(df.status)
Y.head()
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
#Encoding Gender Attribute
X.gender = Le.fit_transform(X.gender)
X.head(10)
#Male-1 Female-0
#Encoding ssc_b Attribute
X.ssc_b = Le.fit_transform(X.ssc_b)
X.head(10)
#Central-0 Others-1
#Encoding hsc_b Attribute
X.hsc_b = Le.fit_transform(X.hsc_b)
X.head(10)
#Central-0 Others-1
#Encoding hsc_s Attribute
X.hsc_s = Le.fit_transform(X.hsc_s)
X.head(10)
#Arts-0 Commerce-1 Science-2
#Encoding degree_t Attribute
X.degree_t = Le.fit_transform(X.degree_t)
```

```
X.head(10)
#Sci&Tech-2 Comm&Mgmt-0 Others-1
#Encoding workex Attribute
X.workex = Le.fit_transform(X.workex)
X.head(10)
#No-0 Yes-1
#Encoding specialisation Attribute
X.specialisation = Le.fit_transform(X.specialisation)
X.head(10)
#Mkt&HR-1 Mkt&Fin-0
#Encoding Target Attribute
Y.status = Le.fit_transform(Y.status)
Y.head()
# #Placed-1 Not Placed-0
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
X_scaled = pd.DataFrame(scale.fit_transform(X),columns=X.columns)
X_scaled.head()
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(X_scaled,Y,test_size=0.3,random_state=5)
x_train.head()
#Size of training dataset
x_train.shape
#Size of testing dataset
x_test.shape
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
dt = DecisionTreeClassifier(class_weight='balanced')
param_grid = {
    'max_depth': [3, 4, 5, 6, 7, 8],
    'min_samples_split': [2, 3, 4],
    'min_samples_leaf': [1, 2, 3, 4],
    'random_state': [0, 42]
}
#Finding Best parameters
```

```
grid_search = GridSearchCV(dt, param_grid, cv=5)
grid_search.fit(x_train, y_train)
# Print the best hyperparameters
print(grid_search.best_params_)
dt = DecisionTreeClassifier(random_state=0, max_depth=4,
min_samples_leaf=4, min_samples_split=2, class_weight='balanced')
dt.fit(x_train, y_train)
from sklearn.metrics import accuracy_score
y_pred = dt.predict(x_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2),
"%")
#Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test,y_pred, cmap =
plt.cm.Blues, normalize = None, display_labels = ['Not Placed', 'Placed'])
#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
#Important Features according to Decision Tree Classifier
imp_df = pd.DataFrame({
    "Feature Name": x_train.columns,
    "Importance": dt.feature_importances_
})
fig = imp_df.sort_values(by="Importance", ascending=False)
fig2 = fig.head(10)
plt.figure(figsize=(10,8))
sns.barplot(data=fig2, x='Importance', y='Feature Name')
plt.title('Top 10 Feature Importance Each Attributes according to Decision
Tree')
plt.xlabel ('Importance')
plt.ylabel ('Feature Name')
plt.show()
from sklearn.ensemble import RandomForestClassifier
rftree = RandomForestClassifier(criterion="gini", min_samples_split=2)
rftree.fit(x_train,y_train)
```

```
from sklearn.metrics import accuracy_score
y_pred1 = rftree.predict(x_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred1)*100 ,2),
"%")
#Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test,y_pred1, cmap =
plt.cm.Blues, normalize = None, display_labels = ['Not Placed', 'Placed'])
#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred1))
#Important Features according to Decision Tree Classifier
imp_df = pd.DataFrame({
    "Feature Name": x_train.columns,
    "Importance": rftree.feature_importances_
})
fig = imp_df.sort_values(by="Importance", ascending=False)
fig2 = fig.head(10)
plt.figure(figsize=(10,8))
sns.barplot(data=fig2, x='Importance', y='Feature Name')
plt.title('Top 10 Feature Importance Each Attributes according to Random
Forest')
plt.xlabel ('Importance')
plt.ylabel ('Feature Name')
plt.show()
from sklearn.svm import SVC
sclf = SVC(kernel='linear')
sclf.fit(x_train, y_train)
from sklearn.metrics import accuracy_score
y_pred2 = sclf.predict(x_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred2)*100 ,2),
"%")
#Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred2, cmap =
plt.cm.Blues, normalize = None, display_labels = ['Not Placed', 'Placed'])
#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred2))
```

- PredictionModel.ipynb

```
x =
df.drop(columns=['sl_no','gender','ssc_b','hsc_b','hsc_s','degree_t','work
ex','etest_p','status','salary'])
x.head()
Y = pd.DataFrame(df.status)
Y.head()
#Encoding Target Attribute
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
Y.status = Le.fit_transform(Y.status)
Y.head()
# #Placed-1 Not Placed-0
#Encoding specialisation Attribute
x.specialisation = Le.fit_transform(x.specialisation)
x.head()
#Mkt&HR-1 Mkt&Fin-0
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
x_scaled = pd.DataFrame(scale.fit_transform(x),columns=x.columns)
x_scaled.head()
x_scaled = pd.DataFrame(scale.fit_transform(x),columns=x.columns)
x_scaled.head()
X_train.head()
X_train.shape
X_test.shape
RFtree = RandomForestClassifier(criterion="gini", min_samples_split=2)
RFtree.fit(X_train,Y_train)
from sklearn.metrics import accuracy_score
Final_pred = RFtree.predict(X_test)
```

```
print("Accuracy Score :", round(accuracy_score(Y_test,Final_pred)*100 ,2),
"%")
#Confusion Matrix
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(Y_test,Final_pred, cmap =
plt.cm.Blues, normalize = None, display_labels = ['Not Placed', 'Placed'])
#Classification Report
from sklearn.metrics import classification_report
print(classification_report(Y_test,Final_pred))
import pickle
pickle.dump(RFtree,open("Prediction.pkl","wb"))
```

* THE END *