Introduction

This dataset consists of placement data of students in xyz campus. It includes secondary and higher secondary school percentage and specialization. It includes degree specialization, type,and work experience and salary offers to the placed students. The purpose of this analysis is to analyze the data and build a classifier using machine learning techniques such as decision tree and random forest, logstic regression to classify the student status into being placed a job or not being placed a job. The analysis will use different classification techniques and compare which classifier make the best prediction for this dataset.

The second part of the analysis will perform a regression analysis using only the students that get job to find out some of the key factors that influence the salary of an offer. The analysis are performed in Python using Jupyter Notebook. V

Loading data and libraries

The analysis starts with loading data and necessary library.

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-py
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(color_codes=True)
```

```
C:\Users\Aishwarya\anaconda3\lib\site-packages\pandas\core\computation\expressions.p
y:21: UserWarning: Pandas requires version '2.8.4' or newer of 'numexpr' (version '2.
8.3' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
C:\Users\Aishwarya\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:60: UserW
arning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' cur
rently installed).
  from pandas.core import (
```

In [2]:
```
df = pd.read_csv("/Users/Aishwarya/Desktop/data scientist/Placement_Data_Full_Class.cs
df.head()
```

Out[2]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8

In [3]:
```
df
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96
...	...	...	...	...	...	...	...	...	...	...	
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89

215 rows × 15 columns

In [4]:
```
df.describe()
```

Out[4]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [5]:
```
#checking null values
df.isnull().sum()
```

Out[5]:
```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary           67
dtype: int64
```

In [6]:
```
df.shape
```

Out[6]:
```
(215, 15)
```

Data Preprocessing Part 1

In [7]:
```
#Check the number of unique value from all of the object datatype
df.select_dtypes(include='object').nunique()
```

Out[7]:
```
gender            2
ssc_b             2
hsc_b             2
hsc_s             3
degree_t          3
workex            2
specialisation    2
status            2
dtype: int64
```

In [8]:
```
# Drop sl_no column because its only identifier column
df.drop(columns='sl_no', inplace=True)
```

```
df.head()
```

Out[8]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specia
0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	I
1	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	I
2	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	I
3	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	I
4	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	I

Data Preprocessing Part 2

In [9]:
```
#Check missing value
check_missing = df.isnull().sum() * 100 / df.shape[0]
check_missing[check_missing > 0].sort_values(ascending=False)
```

Out[9]:
```
salary    31.162791
dtype: float64
```

In [10]:
```
df.shape
```

Out[10]:
```
(215, 14)
```

In [11]:
```
# fill null value in 'salary' column with median because the outlier is pretty ba
df['salary'] = df['salary'].fillna(df['salary'].mean())
df.head()
```

Out[11]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specia
0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	I
1	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	I
2	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	I
3	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	I
4	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	I

In [12]:
```
#Check missing value
check_missing = df.isnull().sum() * 100 / df.shape[0]
check_missing[check_missing > 0].sort_values(ascending=False)
```

Out[12]:
```
Series([], dtype: float64)
```

Label Encoding for each Object datatype

In [13]:
```
# Loop over each column in the DataFrame where dtype is 'object'
for col in df.select_dtypes(include=['object']).columns:

    # Print the column name and the unique values
    print(f"{col}: {df[col].unique()}")
```

```
gender: ['M' 'F']
ssc_b: ['Others' 'Central']
hsc_b: ['Others' 'Central']
hsc_s: ['Commerce' 'Science' 'Arts']
degree_t: ['Sci&Tech' 'Comm&Mgmt' 'Others']
workex: ['No' 'Yes']
specialisation: ['Mkt&HR' 'Mkt&Fin']
status: ['Placed' 'Not Placed']
```

In [14]:
```
from sklearn import preprocessing

# Loop over each column in the DataFrame where dtype is 'object'
for col in df.select_dtypes(include=['object']).columns:

    # Initialize a LabelEncoder object
    label_encoder = preprocessing.LabelEncoder()

    # Fit the encoder to the unique values in the column
    label_encoder.fit(df[col].unique())

    # Transform the column using the encoder
    df[col] = label_encoder.transform(df[col])

    # Print the column name and the unique encoded values
    print(f"{col}: {df[col].unique()}")
```

```
gender: [1 0]
ssc_b: [1 0]
hsc_b: [1 0]
hsc_s: [1 2 0]
degree_t: [2 0 1]
workex: [0 1]
specialisation: [1 0]
status: [1 0]
```

Train Test Split

In [15]:
```
from sklearn.model_selection import train_test_split
# Select the features (X) and the target variable (y)
X = df.drop('status', axis=1)
y = df['status']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=
```

In [16]:
```
from sklearn.linear_model import LogisticRegression
logreg_model=LogisticRegression(solver='lbfgs', max_iter=4000)
logreg_model.fit(X_train,y_train)
y_pre=logreg_model.predict(X_train)
y_pred=logreg_model.predict(X_test)
y_pred1=logreg_model.predict_proba(X_test)
```

In [17]:
```
from sklearn.metrics import accuracy_score,roc_auc_score
from sklearn.model_selection import cross_val_score
print("Training Accuracy_score: {}".format(accuracy_score(y_train,y_pre)*100))
print("Testing Accuracy_score: {}".format(accuracy_score(y_test,y_pred)*100))
print("roc_auc_score: {}".format(roc_auc_score(y_test,y_pred1[:,1])*100))
print("CV_score: {}".format(cross_val_score(logreg_model, X, y, cv=10, scoring='accura
```

```
Training Accuracy_score: 89.7196261682243
Testing Accuracy_score: 77.77777777777779
roc_auc_score: 84.30047694753577
CV_score: 79.06926406926408
```

In [ ]: