Campus Placement Prediction

Mohammad Vohra
PRN : 22070243055

Problem Statement

One of the crucial goals of an academic institution is student placement. An institution's prestige and yearly admissions are inextricably linked to the placements it offers its students. Because of this, every institution works arduously to enhance their placement department in order to advance the institution as a whole. The capacity of an institution to promote its students would be positively impacted by any help in this specific area. Both the institution and the students will always benefit from this.
The primary objective is to forecast, using the data provided in the dataset, whether or not the students will be selected for campus placements.

Approach: The traditional machine learning activities, including model building, model testing, feature engineering, data exploration, and data cleaning.

Description

- This data set consists of Placement data of students in our campus. It includes secondary and higher secondary school percentage and specialization. It also includes degree specialization, type and Work experience and salary offers to the placed students

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	0	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	0	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	0	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	0	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	0	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

How does data looks like?

```
#How does data looks like? /sample 5 rows
df.sample(5)
```

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
152	153	1	75.4	Others	60.5	Central	Science	84.0	Sci&Tech	No	98.0	Mkt&Fin	65.25	Placed	240000.0
27	28	0	63.0	Others	67.0	Others	Commerce	66.0	Comm&Mgmt	No	68.0	Mkt&HR	57.69	Placed	265000.0
159	160	0	52.0	Central	49.0	Others	Commerce	58.0	Comm&Mgmt	No	62.0	Mkt&HR	60.59	Not Placed	NaN
84	85	0	70.0	Central	63.0	Others	Science	70.0	Sci&Tech	Yes	55.0	Mkt&Fin	62.00	Placed	300000.0
115	116	1	73.0	Others	63.0	Others	Science	66.0	Comm&Mgmt	No	89.0	Mkt&Fin	60.50	Placed	216000.0

2. What is size of Data?

```
#What is size of Data? /shape of data
df.shape
#Observation-->rows=215, columns=15
```

```
(215, 15)
```

3. What are the features in data?

```
#What are the features in data? /names of columns
df.columns
#Observation-->Column names aren't understandable, will change in data cleaning
```

```
Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary'],
      dtype='object')
```

4. Basic infromation about data

(null values and data types)

```
]: #basic information about data
df.info()
#Observation-->some null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   sl_no           215 non-null    int64  
 1   gender          215 non-null    int64  
 2   ssc_p           215 non-null    float64
 3   ssc_b           215 non-null    object 
 4   hsc_p           215 non-null    float64
 5   hsc_b           215 non-null    object 
 6   hsc_s           215 non-null    object 
 7   degree_p        215 non-null    float64
 8   degree_t        215 non-null    object 
 9   workex          215 non-null    object 
 10  etest_p         215 non-null    float64
 11  specialisation  215 non-null    object 
 12  mba_p           215 non-null    float64
 13  status          215 non-null    object 
 14  salary          148 non-null    float64
dtypes: float64(6), int64(2), object(7)
memory usage: 25.3+ KB
```

```
#percent of null values
np.round((df.isnull().sum()/df.shape[0])*100, 2)
#Observation-->Salary column seems to have 30% null values,
#will handle missing values in Feuture Engineering
```

```
sl_no              0.00
gender             0.00
ssc_p              0.00
ssc_b              0.00
hsc_p              0.00
hsc_b              0.00
hsc_s              0.00
degree_p           0.00
degree_t           0.00
workex             0.00
etest_p            0.00
specialisation     0.00
mba_p              0.00
status             0.00
salary            31.16
dtype: float64
```

Statistical Description about Data

```
#Statistical Description(only for numerical features)
df.describe()
#Observation-->Salary column seems to have right outliers and missing values
```

	sl_no	gender	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	0.353488	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	0.479168	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	0.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	0.000000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	0.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	1.000000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	1.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

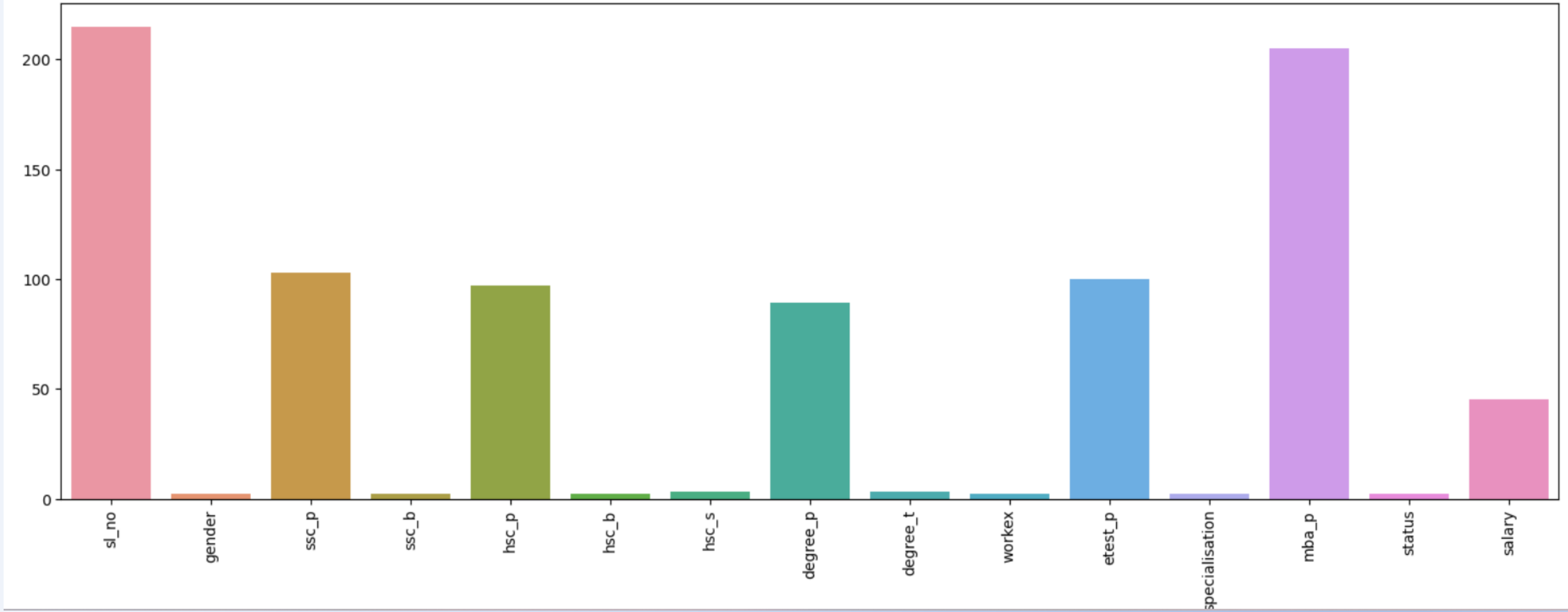
Correlation among features

```
df.corr()
```

	sl_no	gender	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.074306	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.063764
gender	-0.074306	1.000000	0.068969	0.021334	0.173217	-0.084294	0.300531	-0.158912
ssc_p	-0.078155	0.068969	1.000000	0.511472	0.538404	0.261993	0.388478	0.035330
hsc_p	-0.085711	0.021334	0.511472	1.000000	0.434206	0.245113	0.354823	0.076819
degree_p	-0.088281	0.173217	0.538404	0.434206	1.000000	0.224470	0.402364	-0.019272
etest_p	0.063636	-0.084294	0.261993	0.245113	0.224470	1.000000	0.218055	0.178307
mba_p	0.022327	0.300531	0.388478	0.354823	0.402364	0.218055	1.000000	0.175013
salary	0.063764	-0.158912	0.035330	0.076819	-0.019272	0.178307	0.175013	1.000000

Unique number of values in a particular feature

```
plt.figure(figsize=(18,6))
sns.barplot(x = df.nunique().index,y=df.nunique().values)
plt.xticks(rotation='vertical')
plt.show()
```



Data Exploration & Cleaning

Value Counts

```
#percent of all categories in categorical feature
for fea in cat_fea:
    print(f'{fea}\n{(df[fea].value_counts()/df.shape[0])*100}\n')
#Observation-->Imbalanced 'Gender','12th_board','12th_specialization'(Arts),'graduation_deg'(Others),'Work_experience','Placed/Nc
#Observation-->Approximately balanced Data on '10th_board', 'MBA_specialization'
```

```
Gender
0    64.651163
1    35.348837
Name: Gender, dtype: float64

10th_board
Central    53.953488
Others     46.046512
Name: 10th_board, dtype: float64

12th_board
Others     60.930233
Central    39.069767
Name: 12th_board, dtype: float64

12th_specialization
Commerce    52.558140
Science     42.325581
Arts         5.116279
Name: 12th_specialization, dtype: float64
```

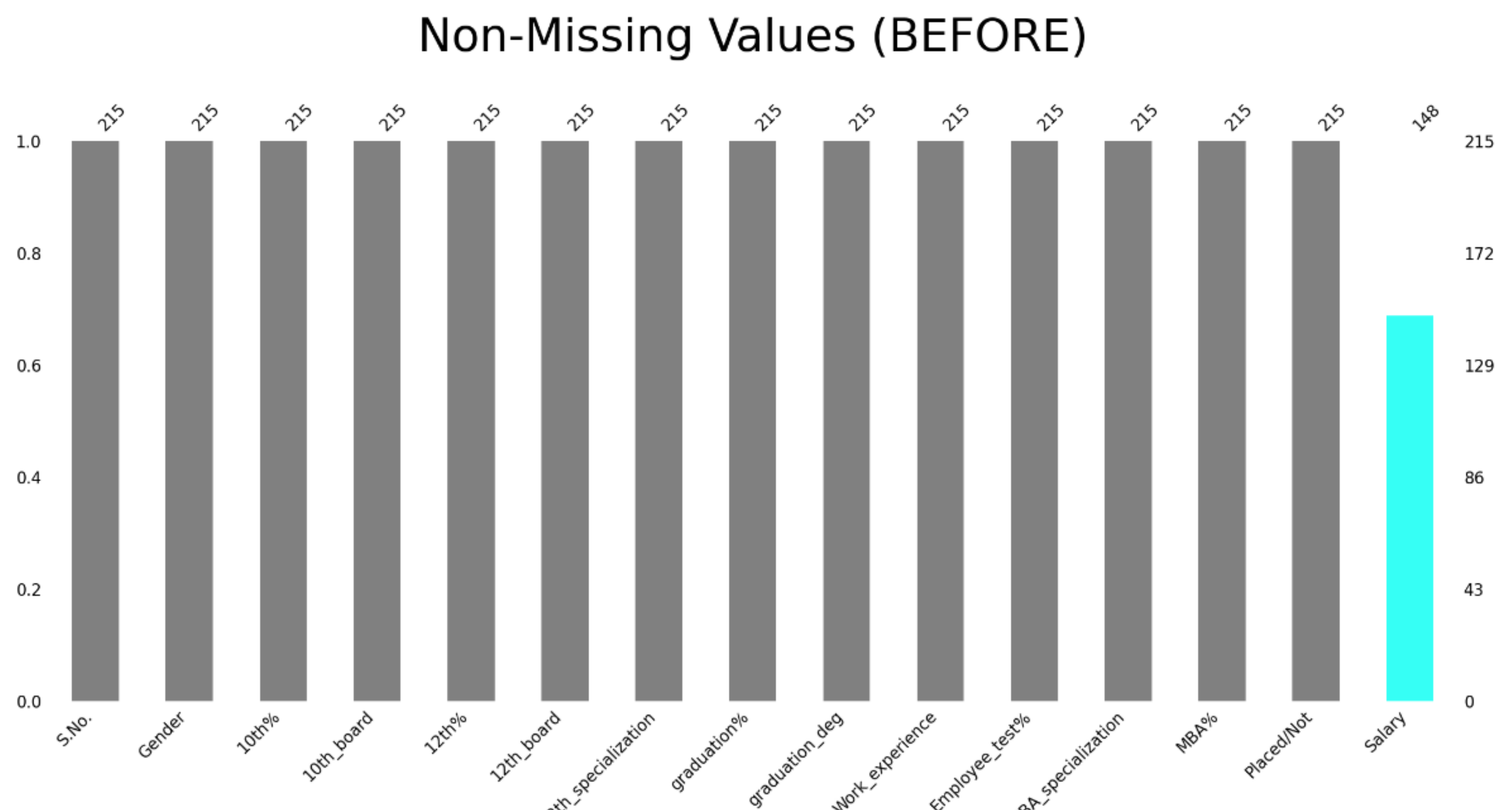
```
graduation_deg
Comm&Mgmt    67.441860
Sci&Tech     27.441860
Others        5.116279
Name: graduation_deg, dtype: float64

Work_experience
No     65.581395
Yes    34.418605
Name: Work_experience, dtype: float64

MBA_specialization
Mkt&Fin    55.813953
Mkt&HR     44.186047
Name: MBA_specialization, dtype: float64

Placed/Not
Placed        68.837209
Not Placed    31.162791
Name: Placed/Not, dtype: float64
```

Exploratory Data Analysis (EDA)

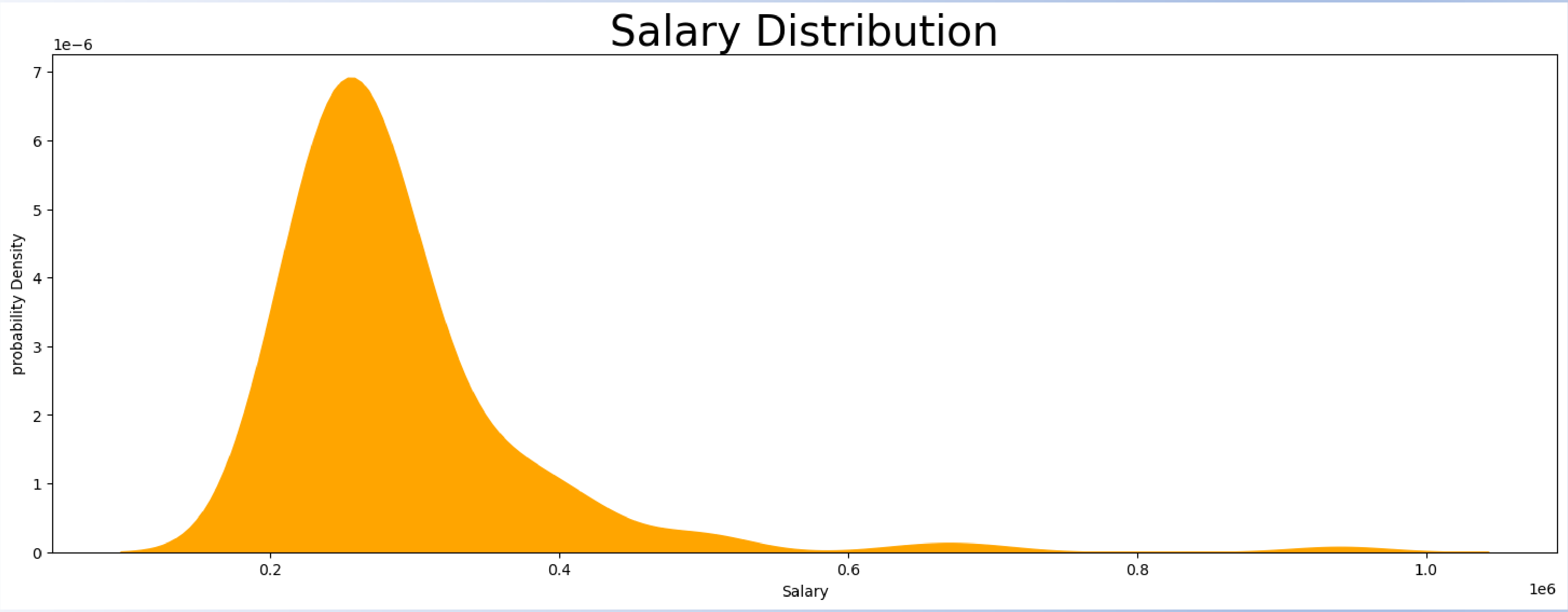
```
mis_color = []

for col in df.columns:
    if df[col].isna().sum() != 0:
        mis_color.append('#36FFF5')
    else:
        mis_color.append('gray')
```

```
msn.bar(df, color=mis_color)
plt.title('Non-Missing Values (BEFORE)', size=45, y=1.15)
#Observation-->Some missing values in 'Salary'
```

```
Text(0.5, 1.15, 'Non-Missing Values (BEFORE)')
```



Salary Distribution(Kernel Density Estimator - KDE Plot)

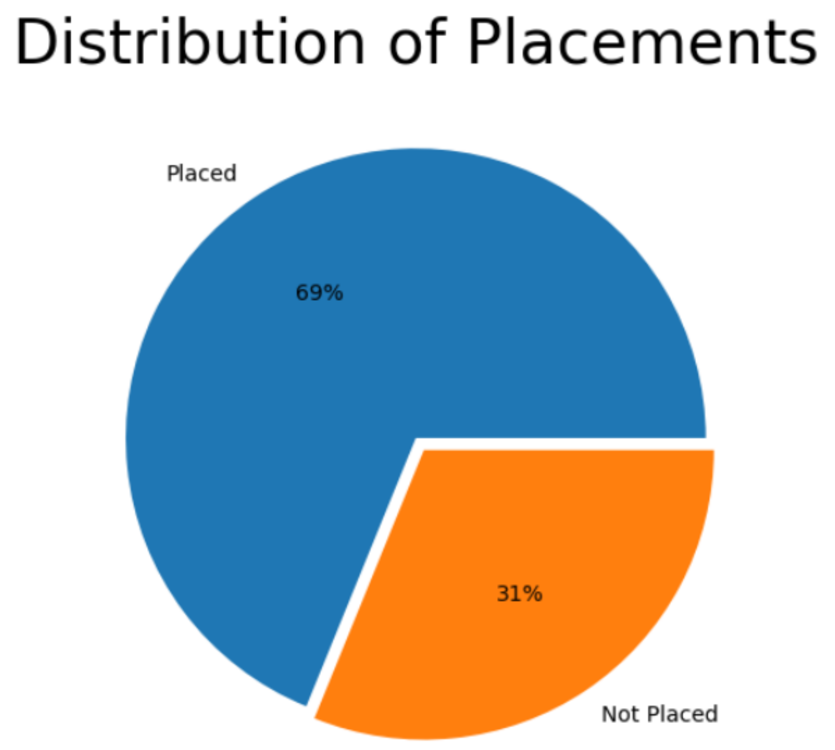



```
#Most frequent Salary
temp_df['Salary'].mode().iloc[0]
```

```
300000.0
```

Placed/Not ratio

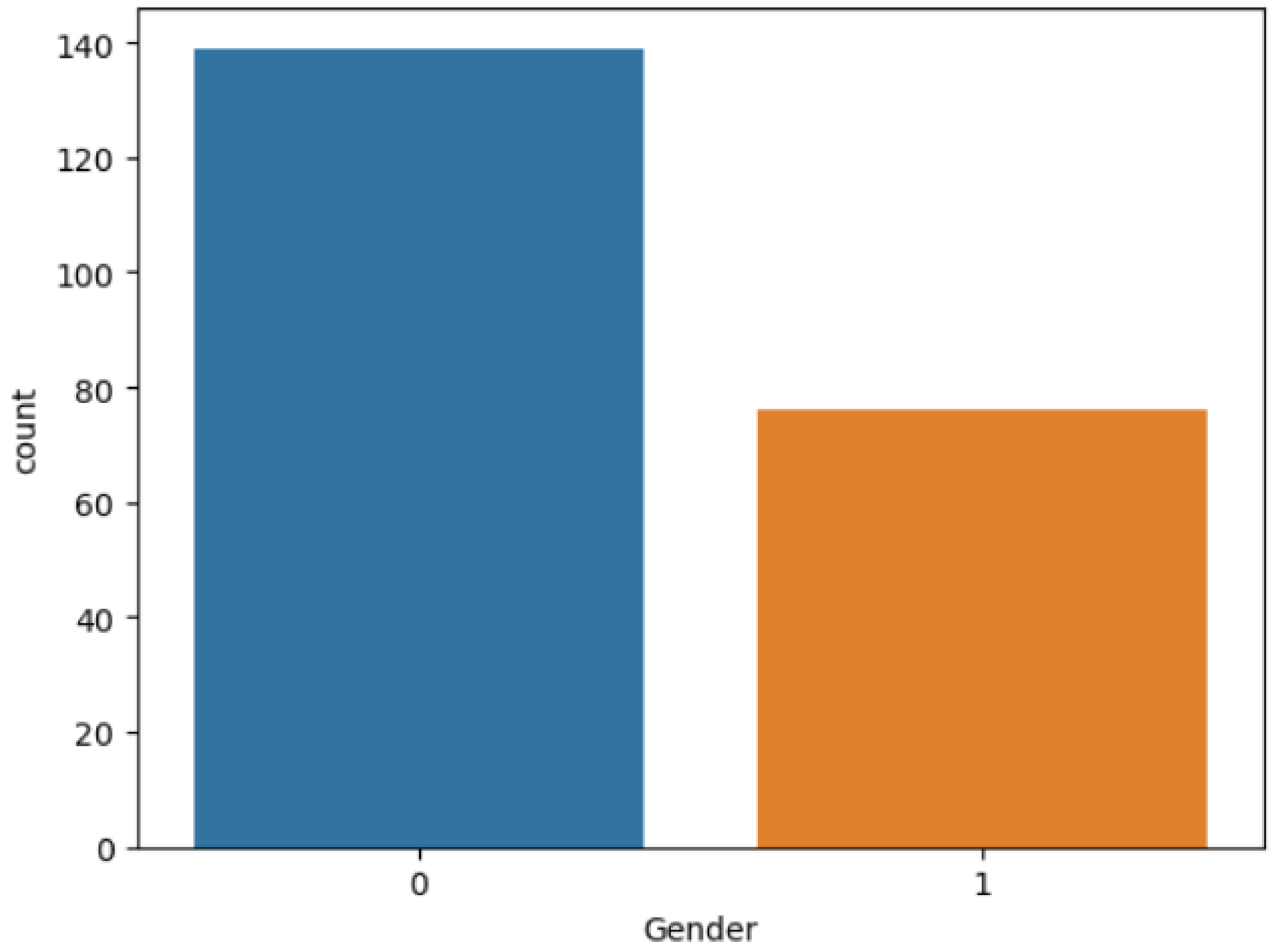
```
# pie chart for Placed/Not

fig = plt.figure(figsize=(18, 6))
ax = fig.add_subplot(111)
plt.title('Distribution of Placements', size=28)

explode = [0, 0.05]
values=df['Placed/Not'].value_counts()
labels=df['Placed/Not'].unique().tolist()
plt.pie(values, labels=labels, explode=explode, autopct='%.0f%%')

# displaying chart
plt.show()
print(values)
```




```
Placed        148
Not Placed     67
```

Gender ratio

```
]: sns.countplot(x='Gender',data=df)

]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



BIVARIATE Analysis

2.1 Salary Vs Placement

```
#Status of Placed/Not wrt NaN Salary
df[['Placed/Not', 'Salary']][np.isnan(df.Salary)]
```

	Placed/Not	Salary
3	Not Placed	NaN
5	Not Placed	NaN
6	Not Placed	NaN
9	Not Placed	NaN
12	Not Placed	NaN
...	...	...
198	Not Placed	NaN
201	Not Placed	NaN
206	Not Placed	NaN
208	Not Placed	NaN
214	Not Placed	NaN

67 rows × 2 columns

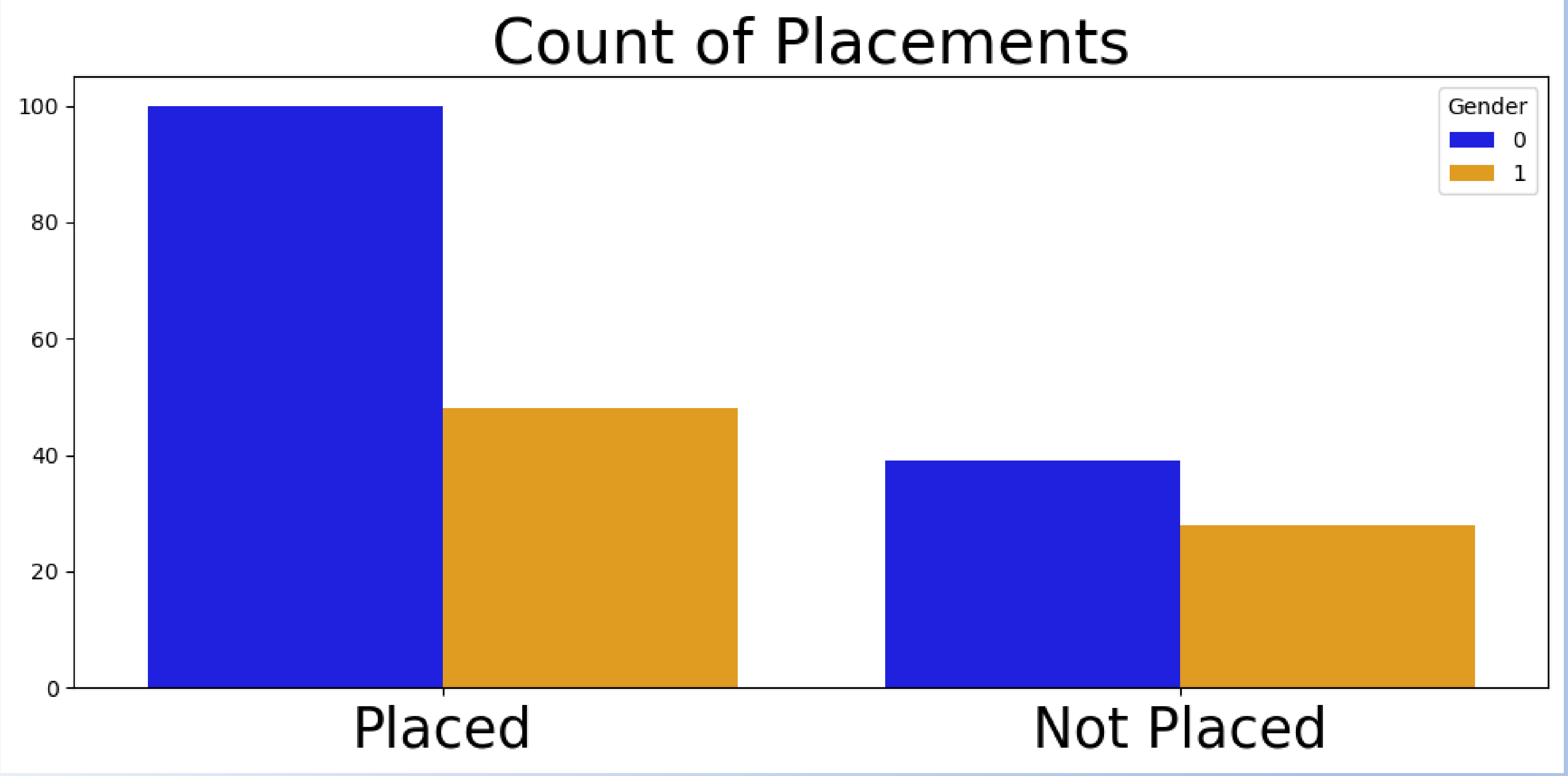
If we look at Salary of students Not got placed, it becomes clearer that all the students who haven't got Placed have 0 salary

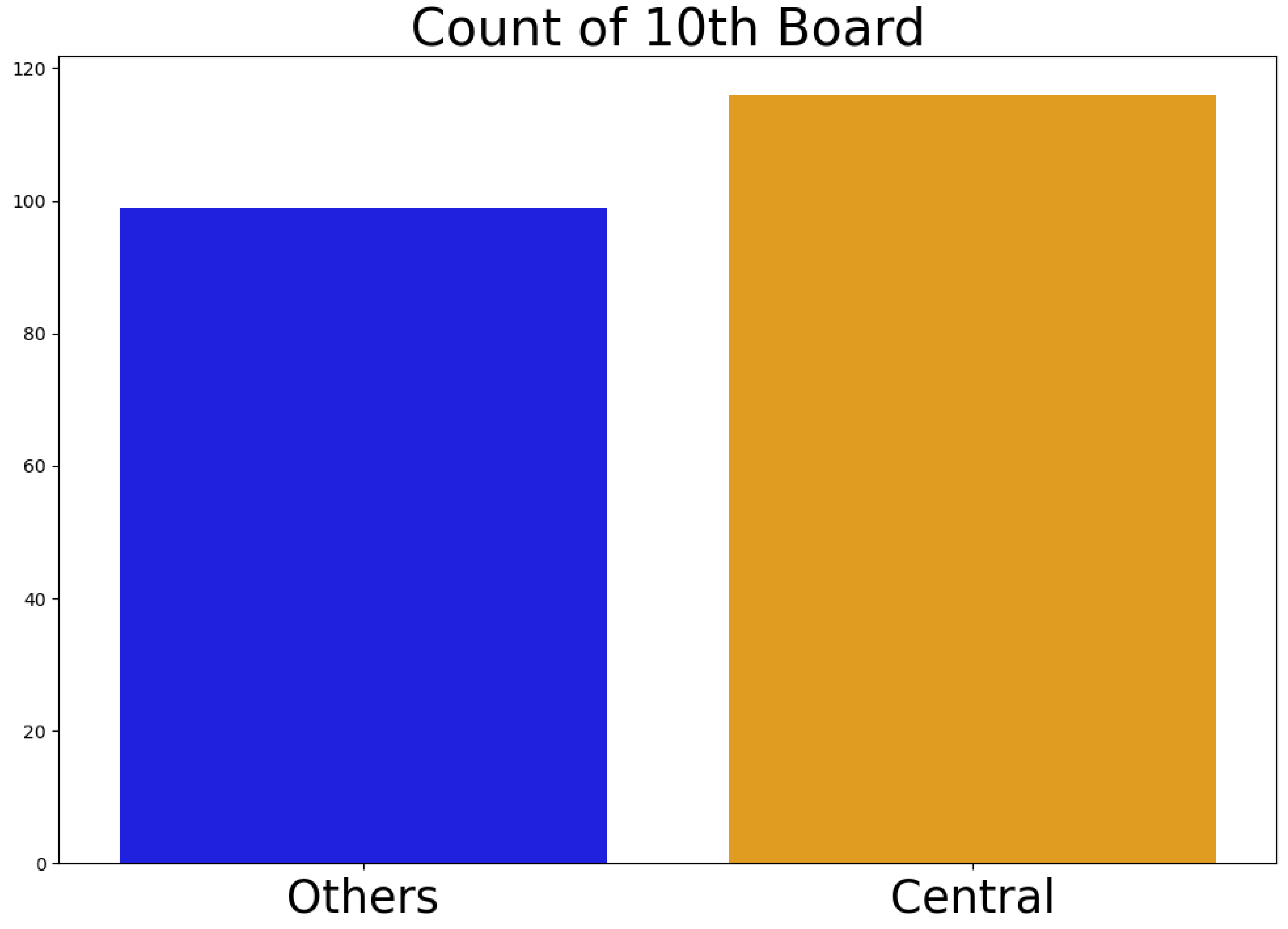
In this case, we can replace the missing values with a variable of "0", but then we will have a direct link with the predicted status, so you should delete this column¶

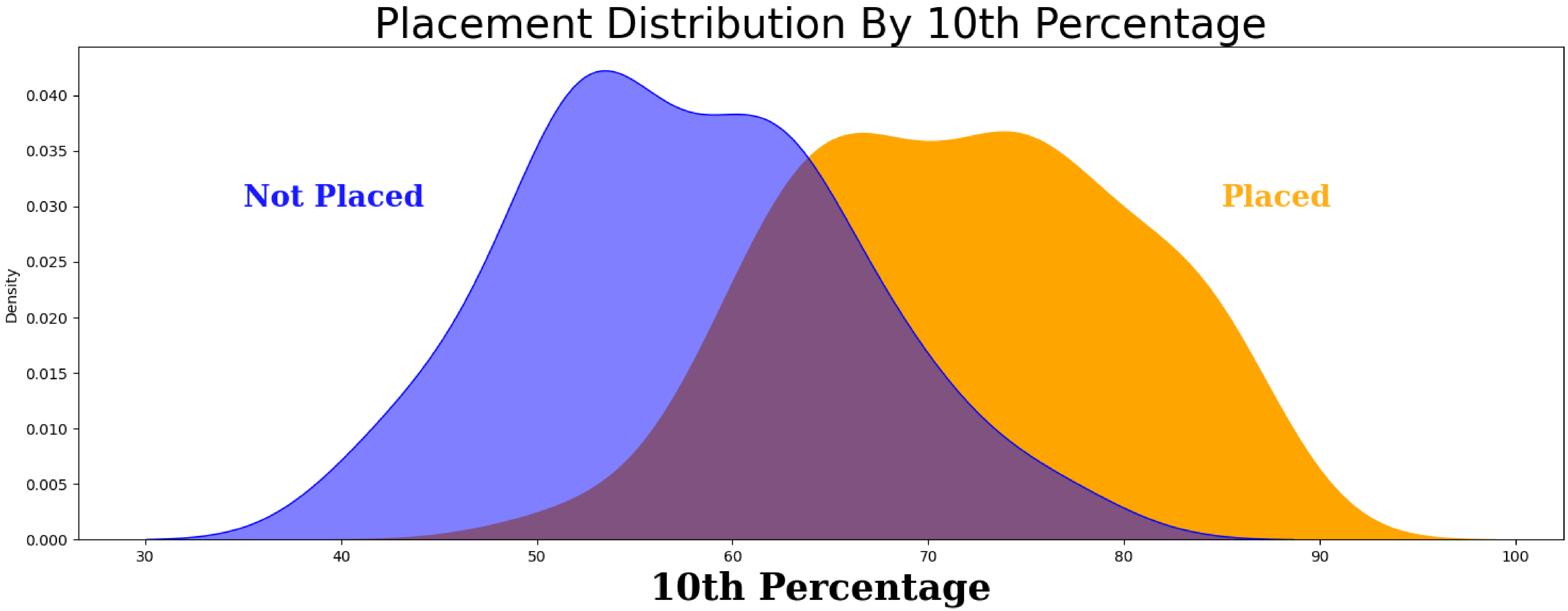
```
#drop Salary and S.No.(as its of no use)
df=df.drop(['Salary', 'S.No.'], axis=1)
#sample row of df
df.sample()
```

	Gender	10th%	10th_board	12th%	12th_board	12th_specialization	graduation%	graduation_deg	Work_experience	Employee_test%	MBA_specialization
150	0	71.0	Central	58.66	Central	Science	58.0	Sci&Tech	Yes	56.0	Mkt&Fin

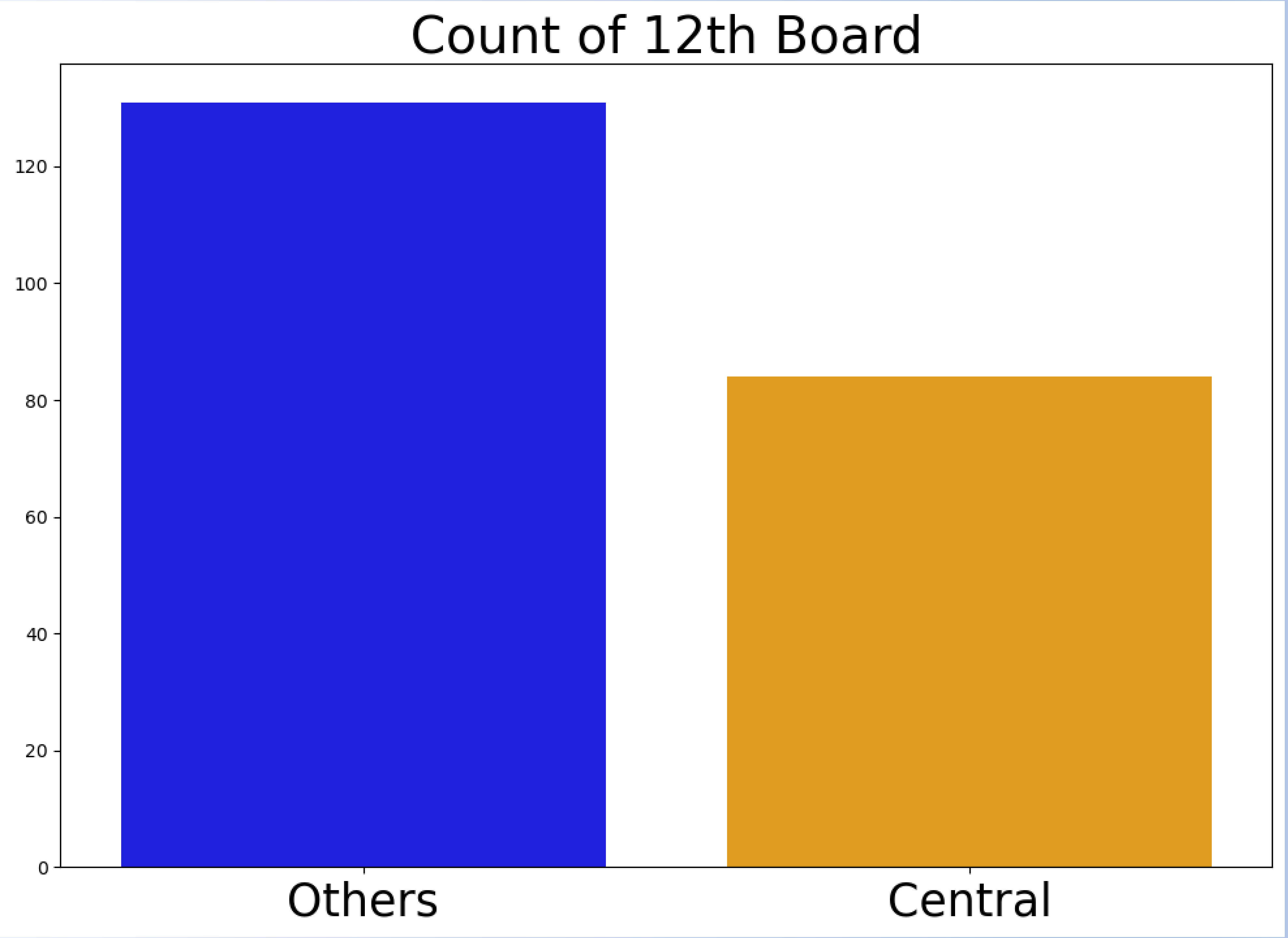
Gender influence on Placement

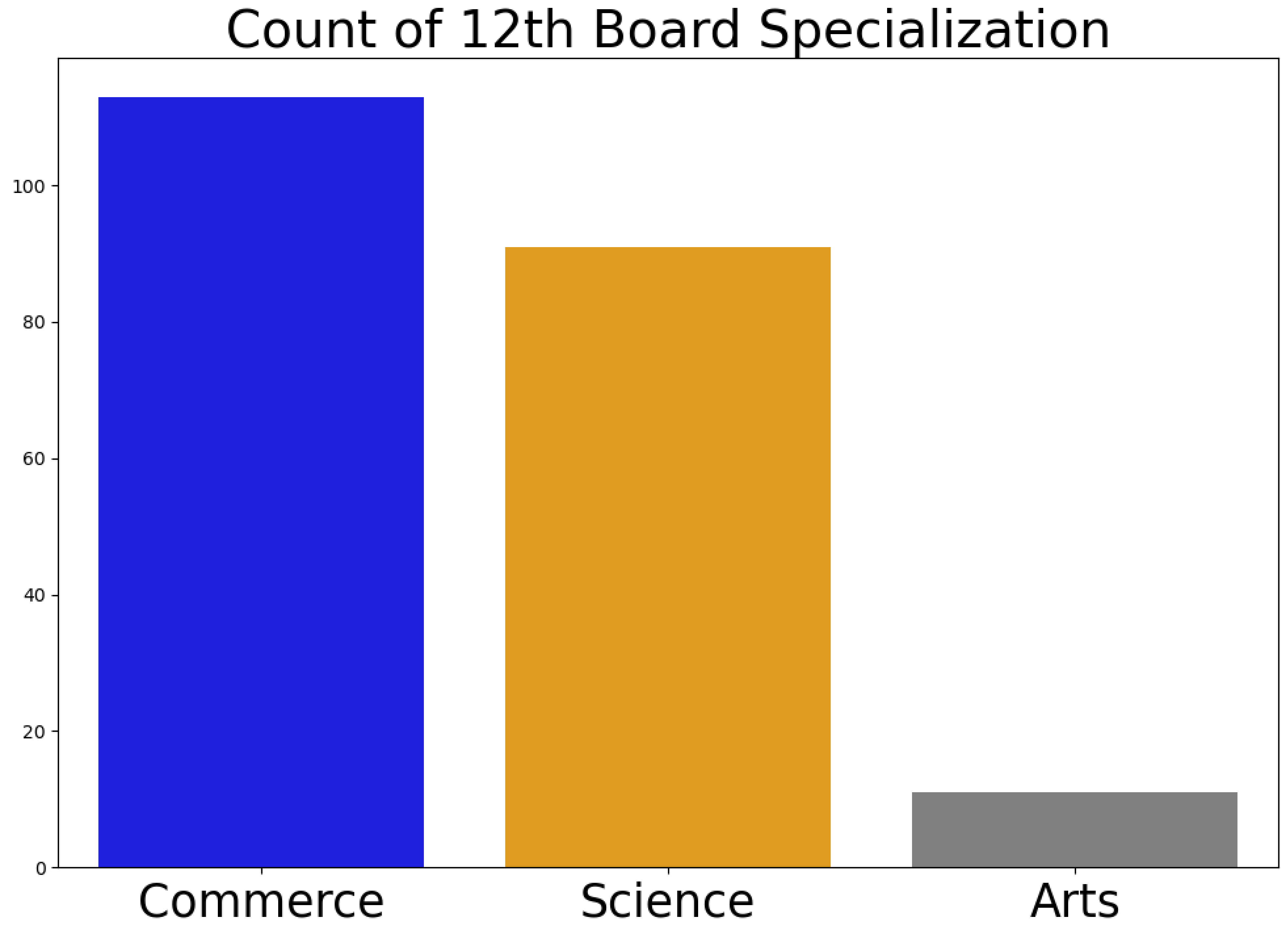



Count of 10th Board
120
100
80
60
40
20
0
Others
Central

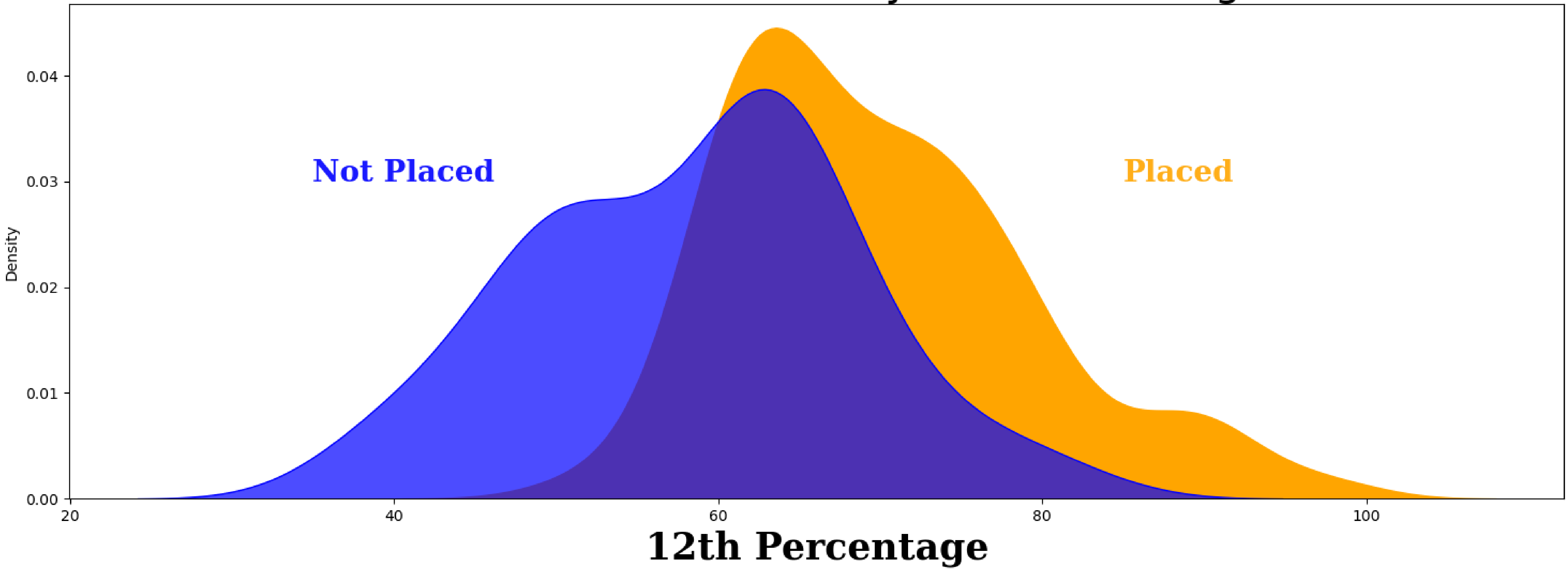


Placement Distribution By 10th Percentage
Not Placed
Placed
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
30
40
50
60
70
80
90
100
10th Percentage

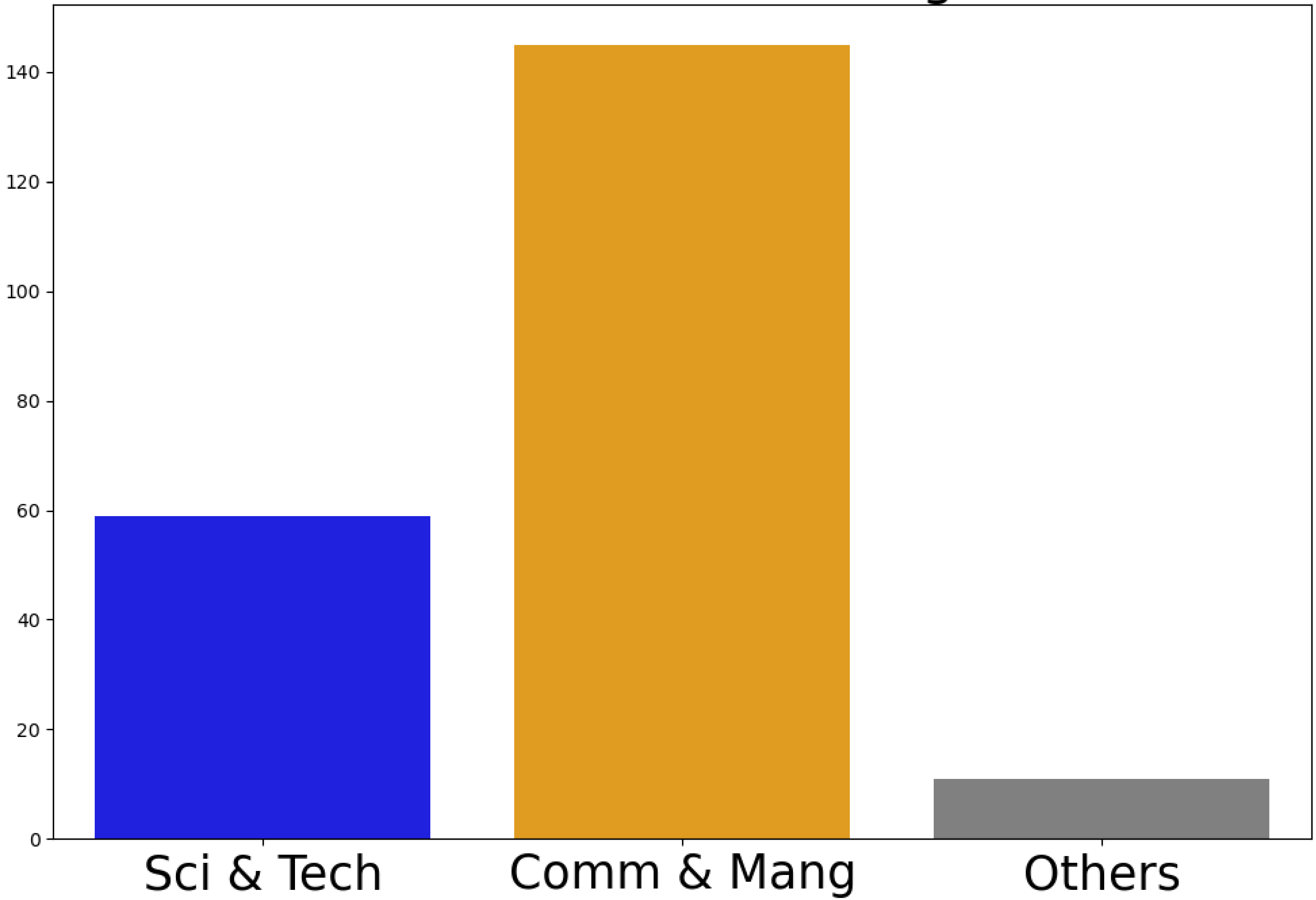

Count of 12th Board
120
100
80
60
40
20
0
Others
Central

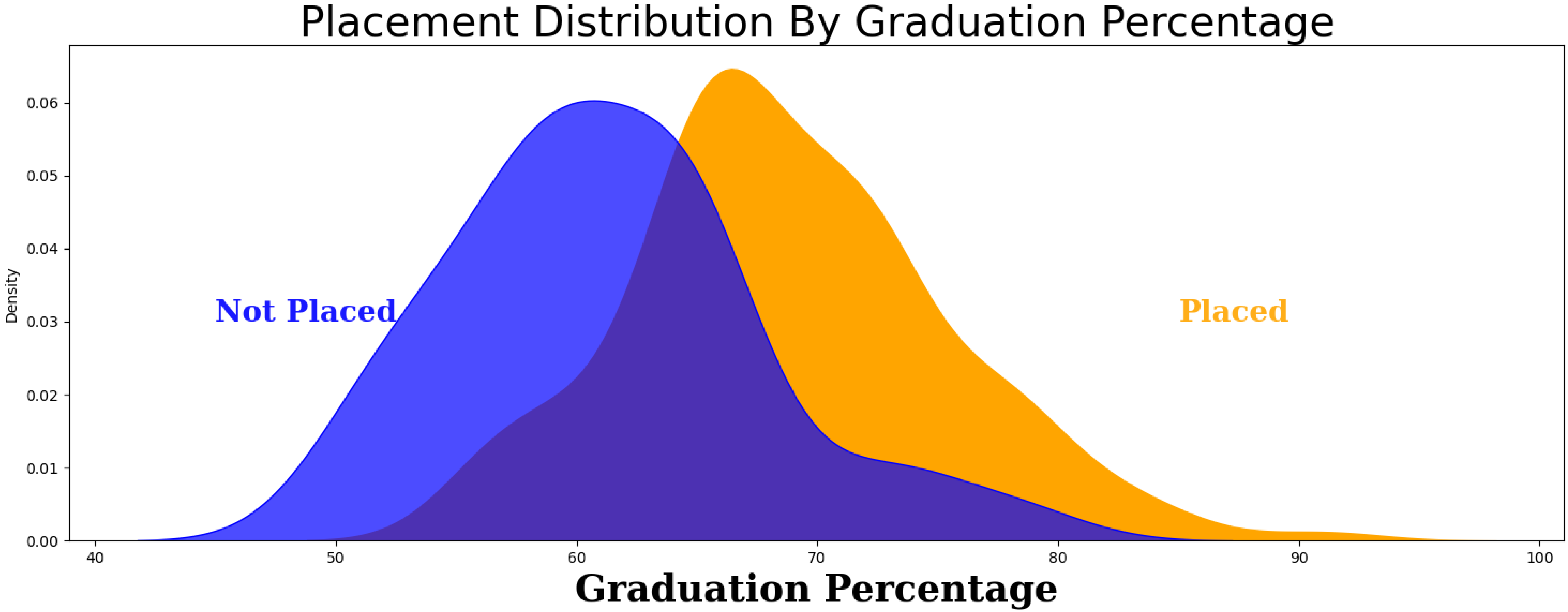

Count of 12th Board Specialization
100
80
60
40
20
0
Commerce
Science
Arts

Placement Distribution By 12th Percentage

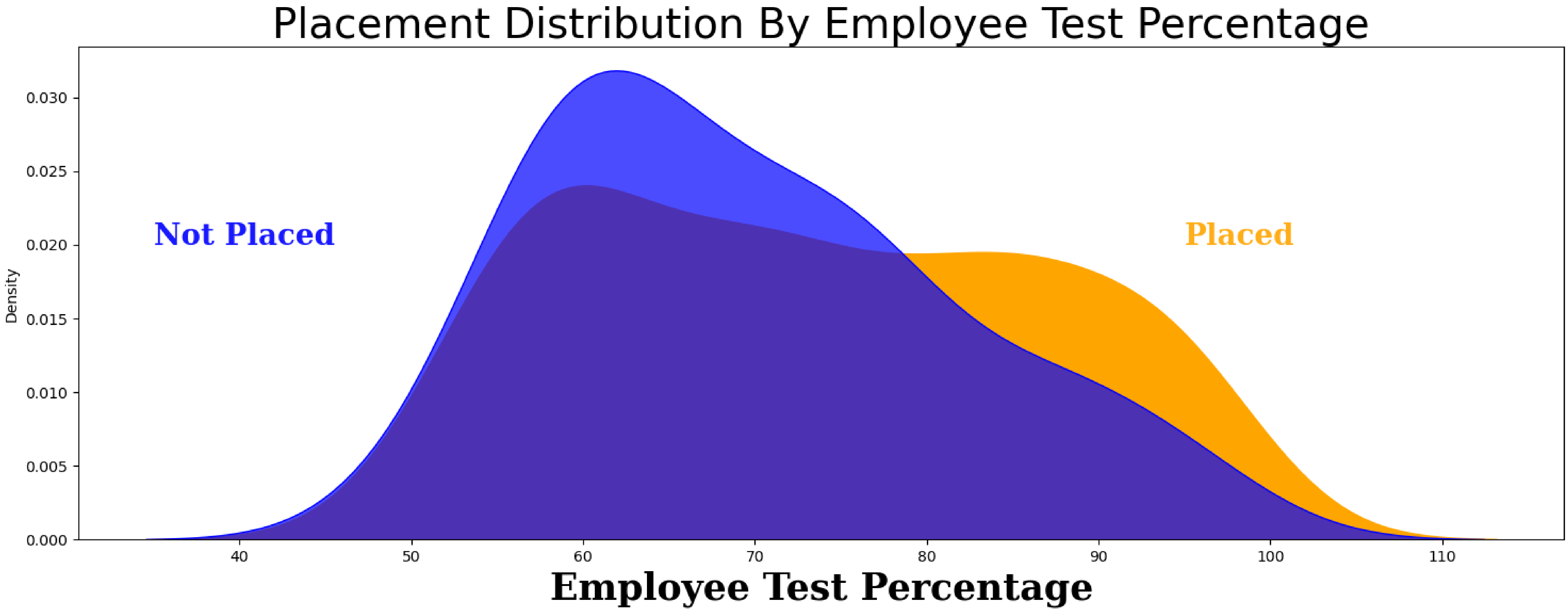


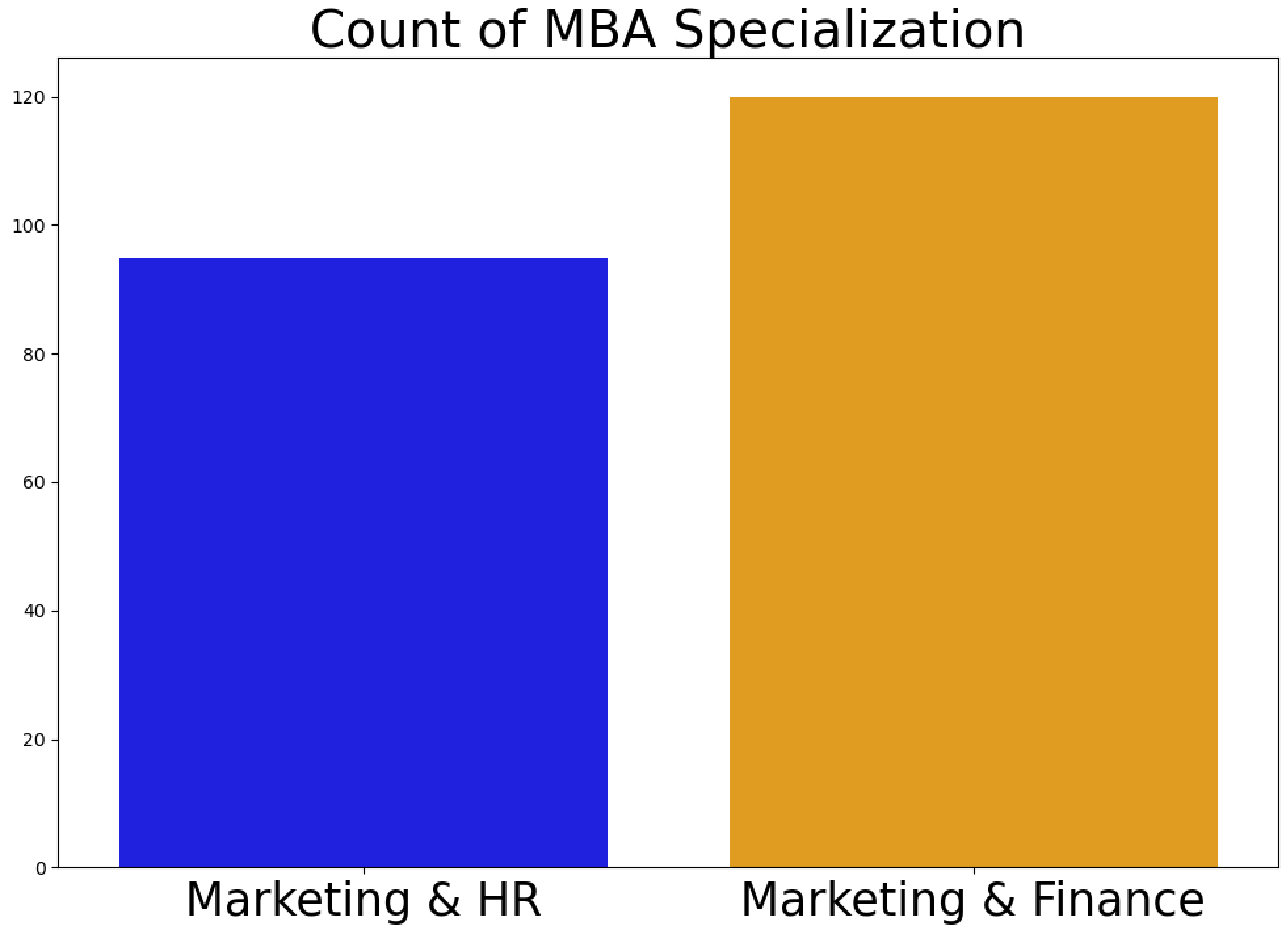
Count of Graduation Degrees

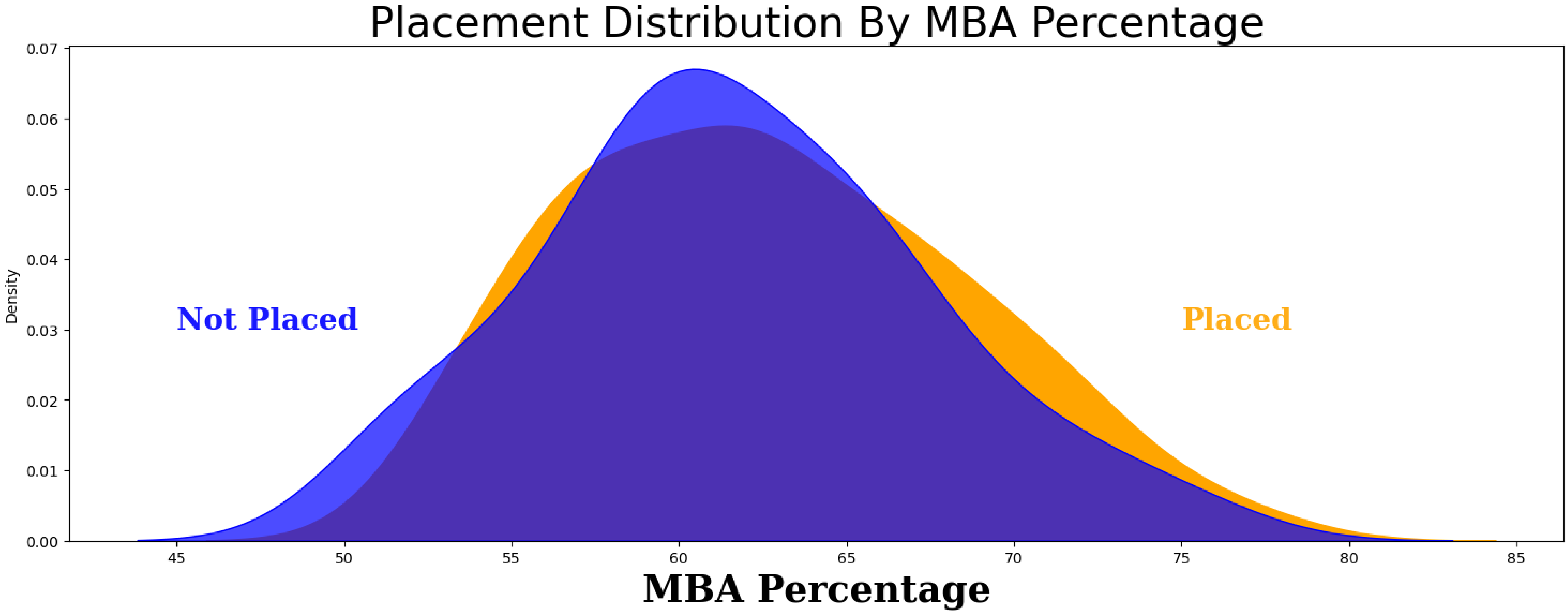


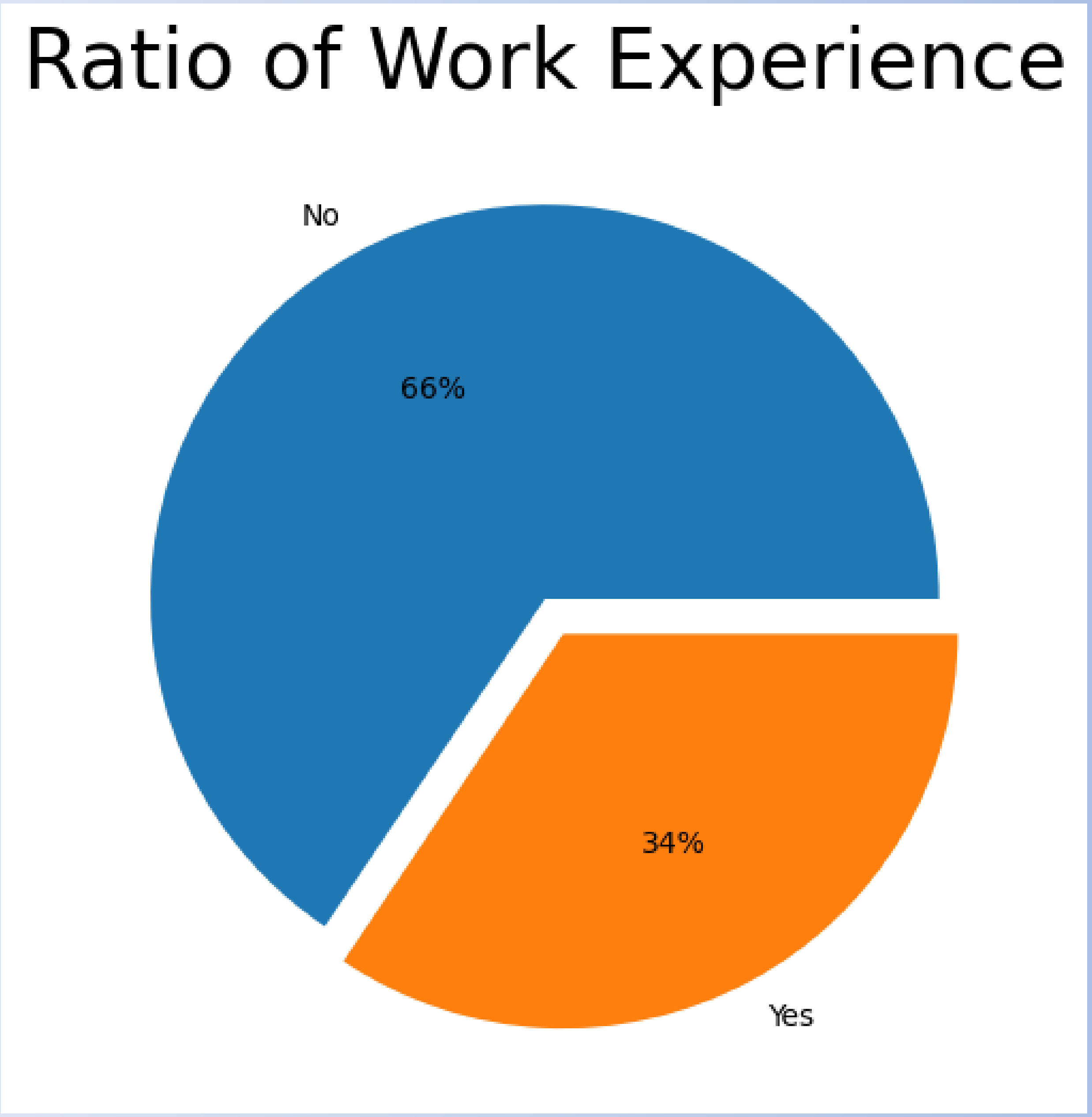


Placement Distribution By Graduation Percentage
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Not Placed
Placed
40
50
60
70
80
90
100
Graduation Percentage


Placement Distribution By Employee Test Percentage
Not Placed
Placed
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
40
50
60
70
80
90
100
110
Employee Test Percentage

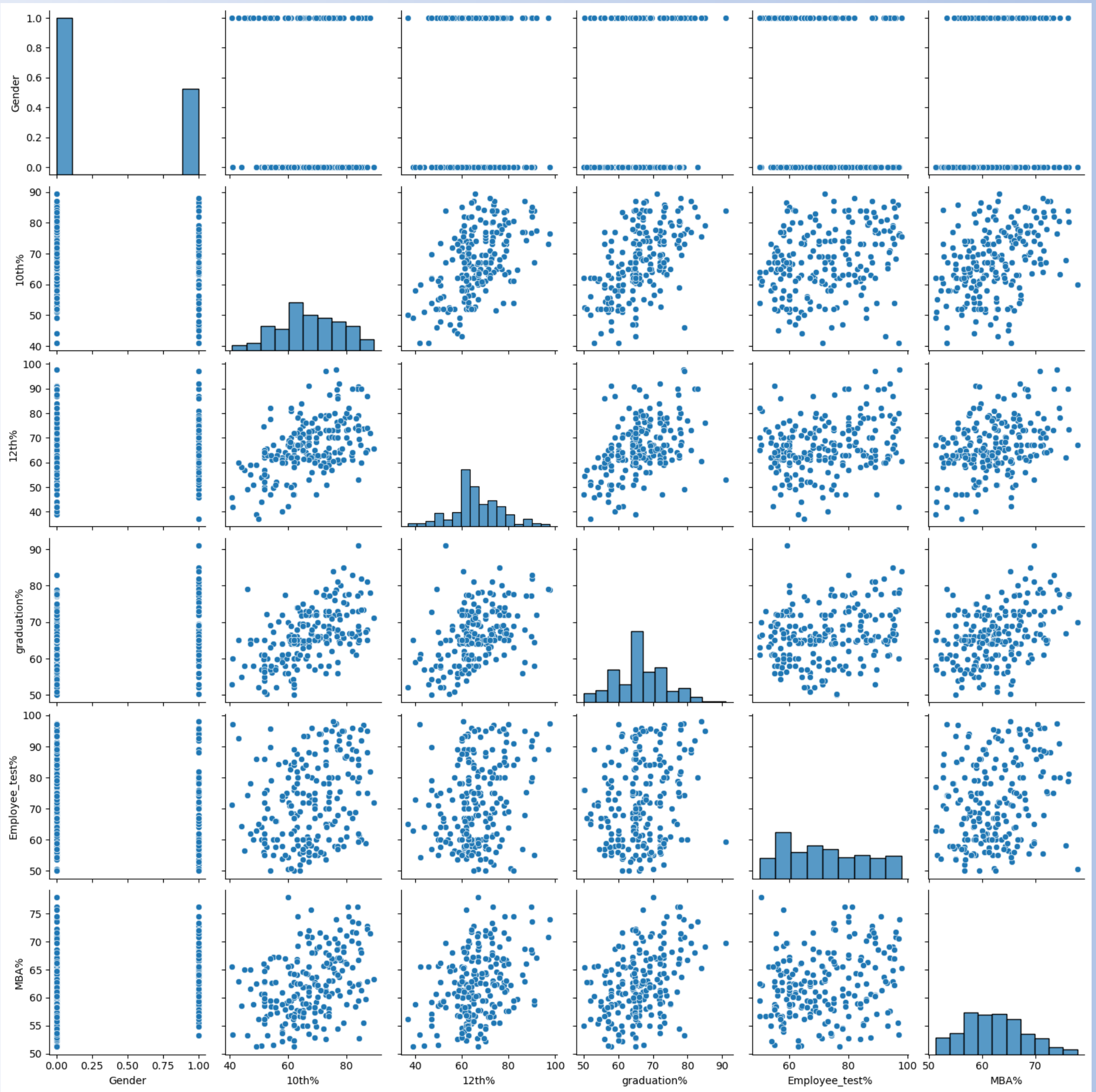

Count of MBA Specialization
120
100
80
60
40
20
0
Marketing & HR
Marketing & Finance


Placement Distribution By MBA Percentage
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Not Placed
Placed
45
50
55
60
65
70
75
80
85
MBA Percentage

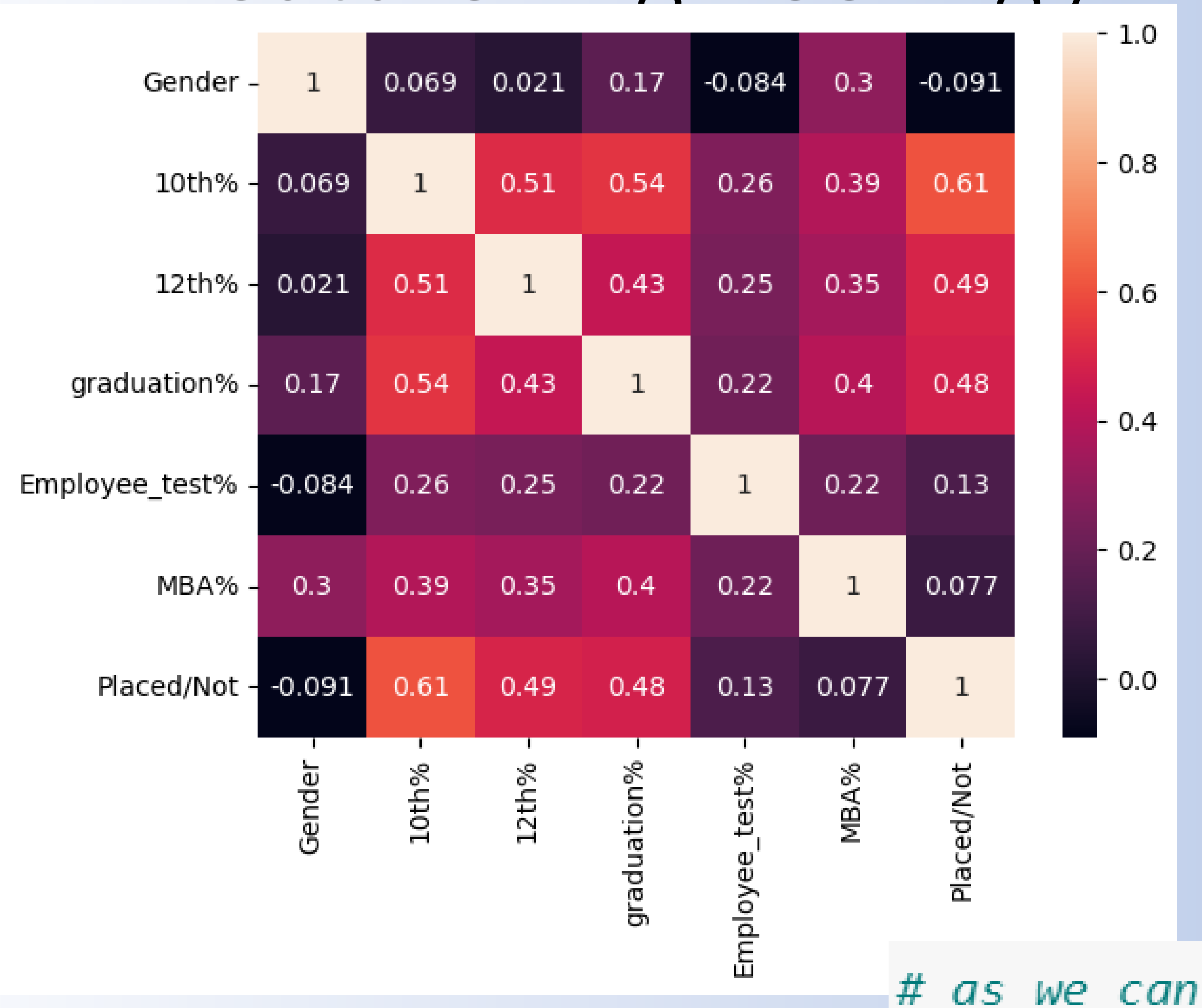

Ratio of Work Experience
No
66%
34%
Yes

Multivariate Analysis

Pairplot(relation of all the features with all others)



Feature Engineering / Preprocessing




```
# as we can see, MBA% is not that much
#correlated with our data we are going to drop it
df.drop("MBA%",axis=1,inplace=True)
```

Encoding

```
df.head()
```

	Gender	10th%	10th_board	12th%	12th_board	12th_specialization	graduation%	graduation_deg	Work_experience	Employee_test%	MBA_specialization	MI
0	0	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	5
1	0	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	6
2	0	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	5
3	0	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	5
4	0	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	5

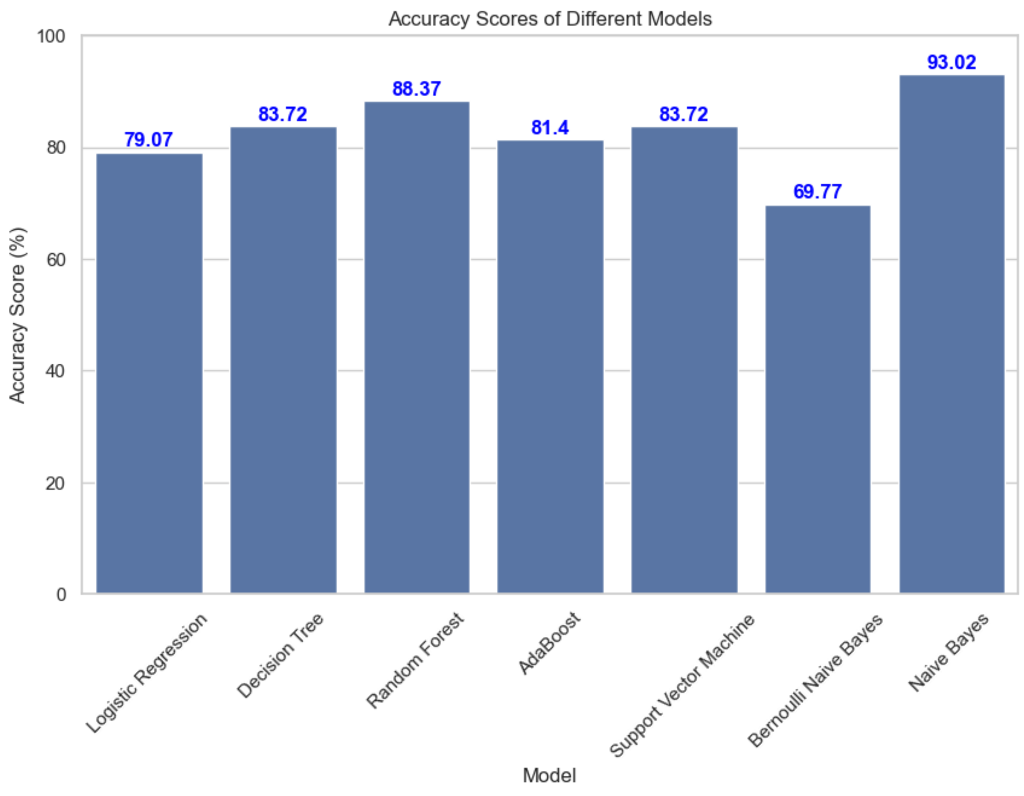
```python
#Encoding into numerical values
from sklearn import preprocessing
encoder = preprocessing.LabelEncoder()
for i in df.columns:
 if isinstance(df[i][0], str):
    df[i] = encoder.fit_transform(df[i])
```

```
df.head()
```

	Gender	10th%	10th_board	12th%	12th_board	12th_specialization	graduation%	graduation_deg	Work_experience	Employee_test%	MBA_specialization	Pl
0	0	67.00	1	91.00	1	1	58.00	2	0	55.0	1	
1	0	79.33	0	78.33	1	2	77.48	2	1	86.5	0	
2	0	65.00	0	68.00	0	0	64.00	0	0	75.0	0	
3	0	56.00	0	52.00	0	2	52.00	2	0	66.0	1	
4	0	85.80	0	73.60	0	1	73.30	0	0	96.8	0	

Model Evaluation

```
Logistic Regression : 79.06976744186046 %
Decision Tree : 81.3953488372093 %
Random Forest : 83.72093023255815 %
AdaBoost : 81.3953488372093 %
Support Vector Machine : 83.72093023255815 %
Bernoulli Naive Bayes : 69.76744186046511 %
Naive Bayes : 93.02325581395348 %
```




```python
import sklearn
from sklearn.model_selection import train_test_split,GridSearchCV,ShuffleSplit
from sklearn.preprocessing import StandardScaler,OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from sklearn.naive_bayes import BernoulliNB,GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,confusion_matrix,classification_report

    model_list = [("Logistic Regression",LogisticRegression()),("Decision Tree",DecisionTreeClassifier()),("Random Forest",Random
              ,("Support Vector Machine",SVC()),("Bernoulli Naive Bayes",BernoulliNB()),("Naive Bayes",GaussianNB())]


plot = {}
# accuracy score on test dataset for all models
for model_name,model in model_list:
    m = model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    plot[model_name] = accuracy_score(y_test,y_pred)*100
    print(f'{model_name} : {accuracy_score(y_test,y_pred)*100} %')
```

Model Evaluation : Naive Bayes

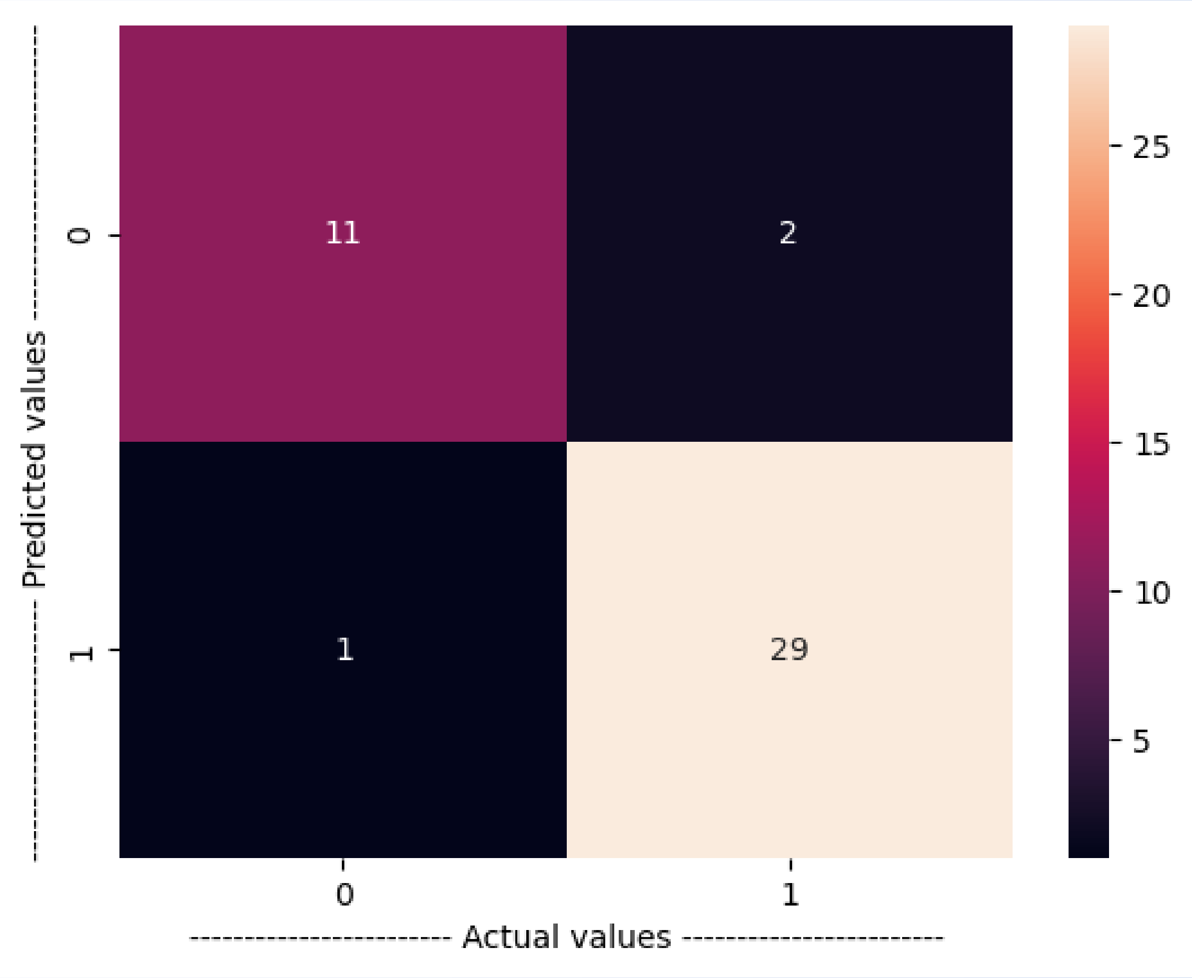
```
classifier = GaussianNB()
classifier.fit(X_train,y_train)
```

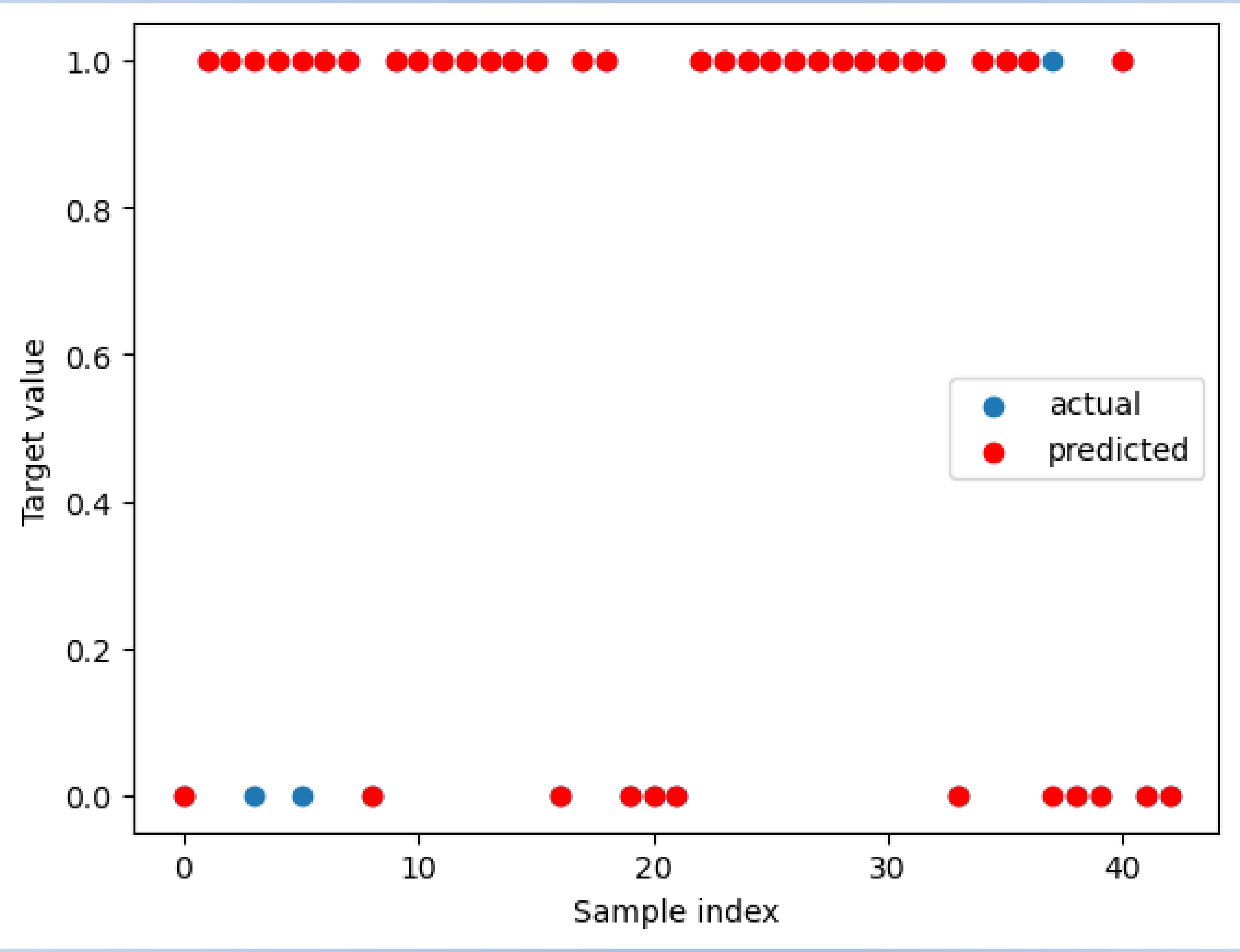
```
GaussianNB()
```

```
# classification report like precision, recall, f1-score  of our model
from sklearn import metrics

y_predict=classifier.predict(X_test)
print("Model accuracy:",accuracy_score(y_test, y_predict)*100,"%")

from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict))
```

```
Model accuracy: 93.02325581395348 %
              precision    recall  f1-score   support

           0       0.92      0.85      0.88        13
           1       0.94      0.97      0.95        30

    accuracy                           0.93        43
   macro avg       0.93      0.91      0.92        43
weighted avg       0.93      0.93      0.93        43
```

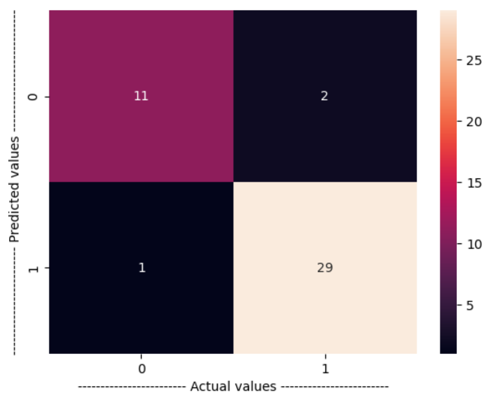
Naïve Bayes : Prediction Visualization

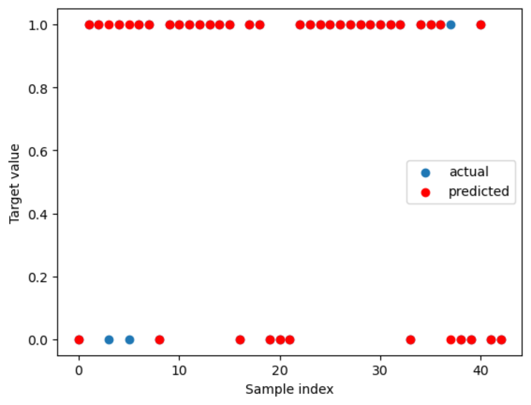




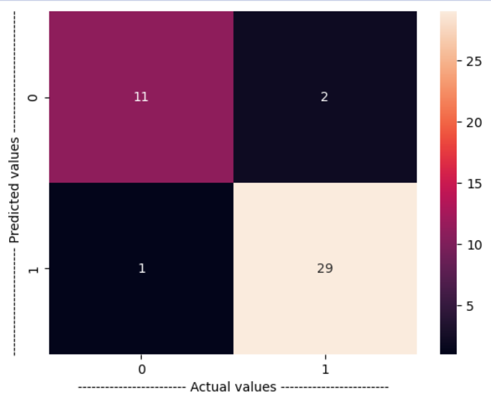
Models : With Acurracy Visualization

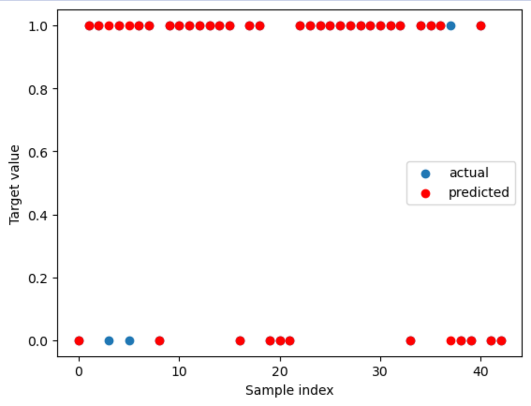
LogisticRegression

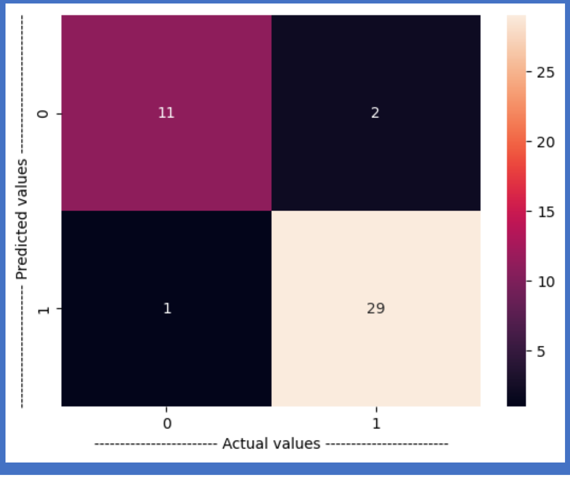
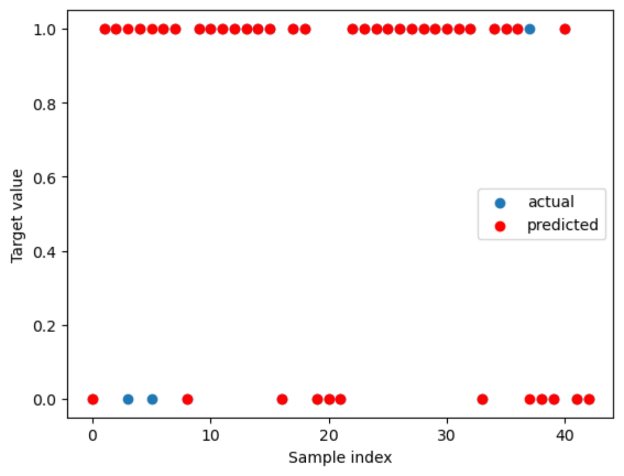
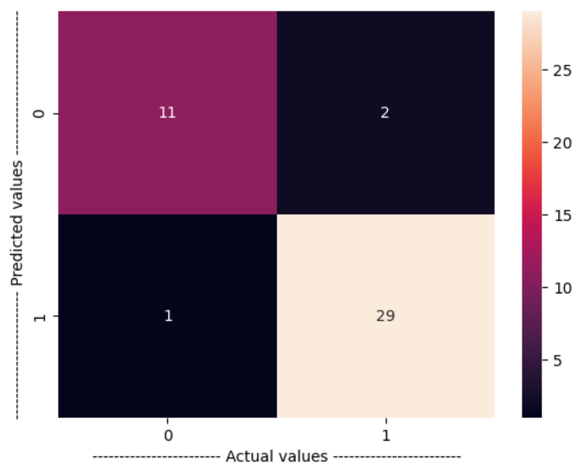
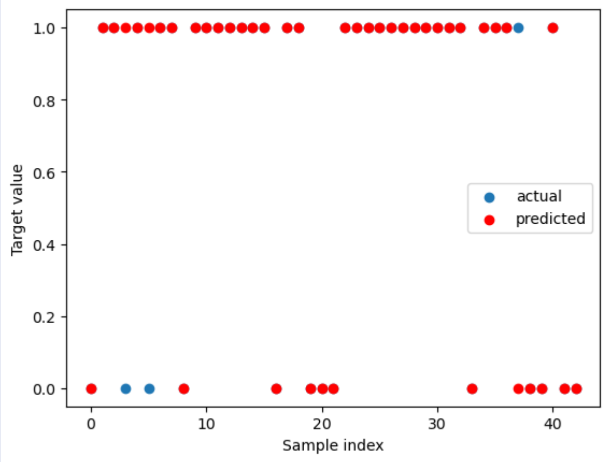


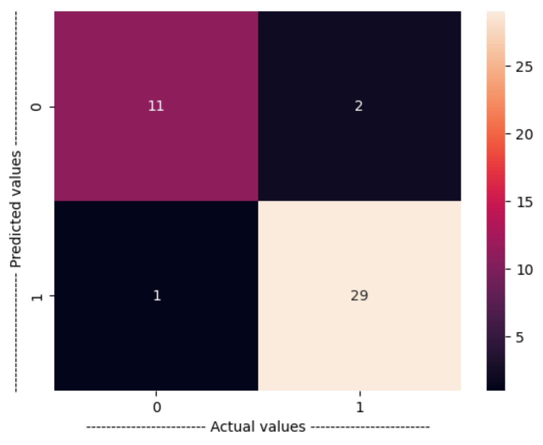
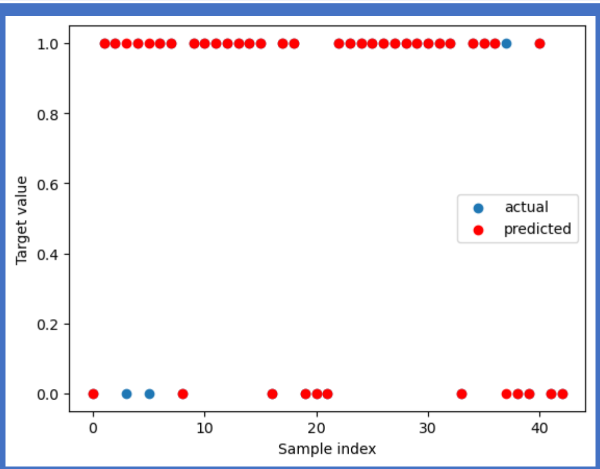
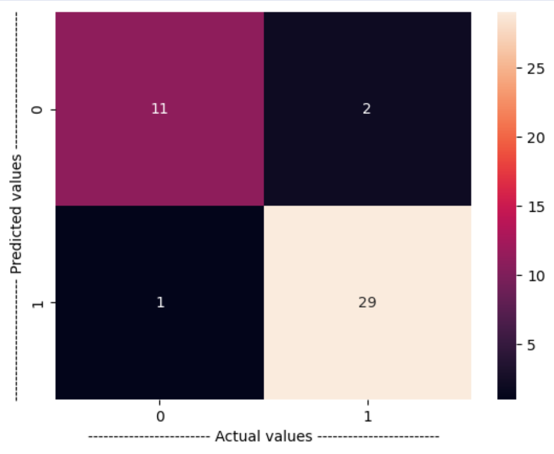
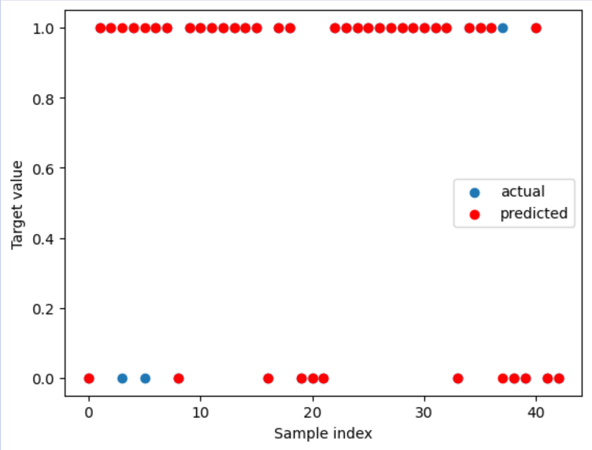


Decision Tree





Random Forest		
AdaBoost		

<table>
<tr>
<td>Support Vector Machine</td>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>Bernoulli Nave Bayes</td>
<td>
</td>
<td>
</td>
</tr>
</table>

Conclusion

- In conclusion, campus placement prediction is an important task for educational institutions and companies to effectively match candidates with job opportunities. Various machine learning models such as logistic regression, decision trees, random forests, and neural networks can be used to predict campus placements based on factors such as academic performance, technical skills, and personal traits. Feature selection and hyperparameter tuning can improve the accuracy of these models. Furthermore, the use of ensemble methods and hybrid models can lead to more robust predictions. Overall, campus placement prediction can provide valuable insights to both educational institutions and companies in their decision-making process.