Assignment Report

Student: C0904838, Haldo Somoza
Date: Jun 26, 2024

Attached to this report comes the notebook elaborated for the Campus Placement Prediction assignment that involves several steps. Below there is a summary of the actions taken:

1. **Data Selection and Loading**

The dataset was loaded into a pandas DataFrame and then identified their columns.
The datasource was located from: https://www.kaggle.com/c/ml-with-python-course-project/data

2. **Data Preprocessing and Exploratory Data Analysis (EDA)**

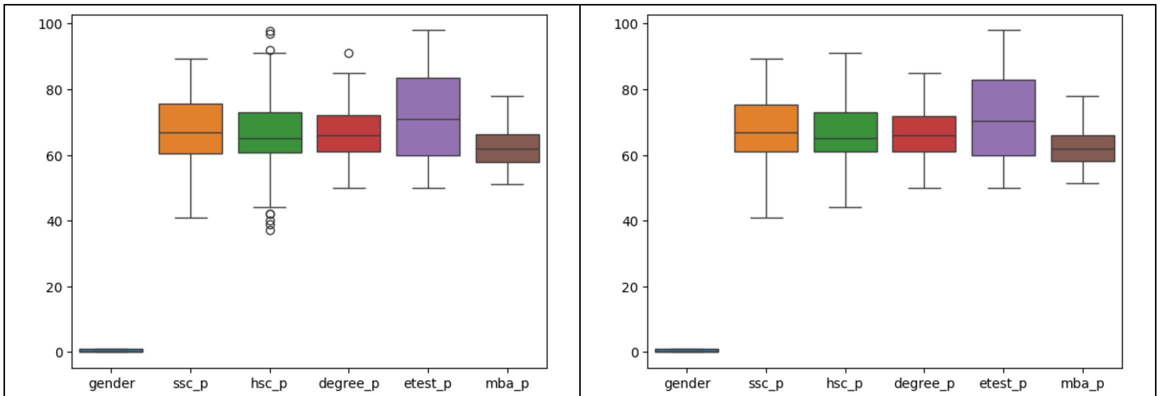
The data preprocessing and data analysis was conducted through next steps:

- Displayed data and statistics to familiarize with the dataset.

```
<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
```

- Removed unnecessary columns such sl_no (sequential index) and salary, this last one because was another target variable that was not selected by this project.
- Handled missing values: null values and duplicated. The result was not encountered null values neither duplicated record.

- Identified outliers through drawing boxplot. Two columns were identified: hsc_p and degree_p.
- Removed the outlier data for the two columns previous identified.
- Applied standard scaling for numerical columns.



- numerical columns showing outliers -

- columns after removing outliers -

```python
# Scaling numerical columns
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['gender', 'ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p']] = scaler.fit_transform(df[['gender', 'ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p']])
df
```

✓ 0.2s Open 'df' in Data Wrangler

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status
0	-0.733017	-0.045537	Others	2.528967	Others	Commerce	-1.169280	Sci&Tech	No	-1.294408	Mkt&HR	-0.603259	Placed
1	-0.733017	1.118422	Central	1.215202	Others	Science	1.585204	Sci&Tech	Yes	1.119619	Mkt&Fin	0.705320	Placed
2	-0.733017	-0.234338	Central	0.144075	Central	Arts	-0.320877	Comm&Mgmt	No	0.238307	Mkt&Fin	-0.778202	Placed

- Identified the categorical columns and their values. The columns identified were: ssc_b, hsc_b, hsc_s, degree_t, workex, specialization, and status.
- Encoded the categorical columns previously identified.

```python
# Converting not numerical columns to numerical
# First, let's check the unique values of each not numeric column
for column in df.columns:
    if df[column].dtype == 'object':
        print(f'{column}: {df[column].unique()}')
```

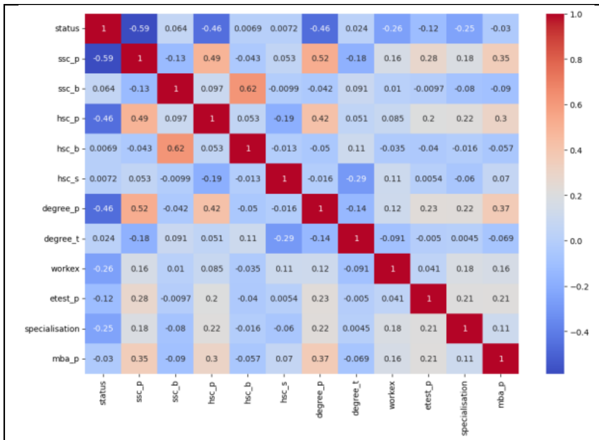
✓ 0.0s Python

```
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']
```

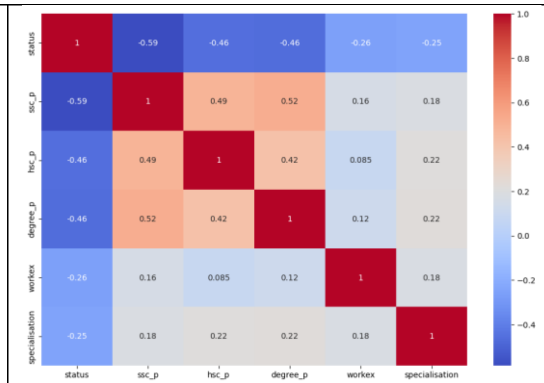
```python
# Converting not numerical columns to numerical
# Assigning the values to the categorical columns identified
df['ssc_b']          = df['ssc_b']          .map({'Others':   0, 'Central':    1})
df['hsc_b']          = df['hsc_b']          .map({'Others':   0, 'Central':    1})
df['hsc_s']          = df['hsc_s']          .map({'Commerce': 0, 'Science':    1, 'Arts':   2})
df['degree_t']       = df['degree_t']       .map({'Sci&Tech': 0, 'Comm&Mgmt':  1, 'Others': 2})
df['workex']         = df['workex']         .map({'No':       0, 'Yes':        1})
df['specialisation'] = df['specialisation'].map({'Mkt&HR':   0, 'Mkt&Fin':    1})
df['status']         = df['status']         .map({'Placed':   0, 'Not Placed': 1})

df
```

✓ 0.0s Open 'df' in Data Wrangler Python

- Showed the correlation matrix to identify columns with high similar correlation and columns with low correlation. It was no identified columns with high correlation, and many columns with low correlation: ssc_b, hsc_b, hsc_s, degree_t, etest_p, and mba_p.
- Removed the columns with low correlation.



correlation matrix with all columns



columns chosen for modeling

- Splitted the data set into training (70%) and testing (30%) datasets, both for input variables (X) and target variable (y).

4. Model Selection through Grid Search

There were chosen for models to evaluate: Naive Bayes, Support Vector Machine (SVM), Logistic Regression, and Decision Tree. Those were selected because fit for classification problems and for small datasets. GridSearchCV was used to find the optimal parameters for each model. The results were as follow:

```
Performing Grid Search for GaussianNB ...
Model evaluated: GaussianNB
Best params:     {'priors': None}
Test R2 Score:   0.3843971631205674
Test MSE:        0.11290322580645161
Test Accuracy:   0.8870967741935484

Performing Grid Search for LinearSVC ...
Model evaluated: LinearSVC
Best params:     {'loss': 'hinge', 'max_iter': 10}
Test R2 Score:   0.03262411347517735
Test MSE:        0.1774193548387097
Test Accuracy:   0.8225806451612904

Performing Grid Search for LogisticRegression ...
Model evaluated: LogisticRegression
Best params:     {'max_iter': 10, 'penalty': 'l1', 'solver': 'saga'}
Test R2 Score:   0.2085106382978723
Test MSE:        0.14516129032258066
Test Accuracy:   0.8548387096774194

Performing Grid Search for DecisionTreeClassifier ...
Model evaluated: DecisionTreeClassifier
Best params:     {'criterion': 'gini', 'max_depth': 5, 'max_features': 5}
Test R2 Score:   -0.4070921985815603
Test MSE:        0.25806451612903225
Test Accuracy:   0.7419354838709677
```

6. Model Selection through Voting Classifier

A similar evaluation model was made with Voting Classifier (with hard voting) and the result suggest the same hyperparameters than GridSearchCV, but every execution give different results.

```
Models evaluated:
GaussianNB()
LinearSVC(loss='hinge', max_iter=10)
LogisticRegression(penalty='l1', solver='saga')
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_features=4)

Scores of Voting Model selected:
Test R2 Score: 0.12056737588652489
Test MSE:      0.16129032258064516
Test Accuracy: 0.8387096774193549
```

```
Models evaluated:
GaussianNB()
LinearSVC(loss='hinge', max_iter=10)
LogisticRegression(penalty='l1', solver='saga')
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_features=4)

Scores of Voting Model selected:
Test R2 Score: 0.03262411347517735
Test MSE:      0.1774193548387097
Test Accuracy: 0.8225806451612904
```

```
Models evaluated:
GaussianNB()
LinearSVC(loss='hinge', max_iter=10)
LogisticRegression(penalty='l1', solver='saga')
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_features=4)

Scores of Voting Model selected:
Test R2 Score: -0.14326241134751783
Test MSE:      0.20967741935483872
Test Accuracy: 0.7903225806451613
```

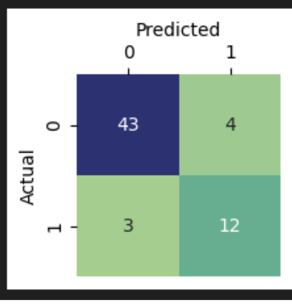
5. Model Training and Evaluation of Chosen Model

Finally, the Naïve Bayes model was chosen because the metrics given for Grid Search:
- Higher R2 Scoring: indicating a better relation between dependent variables and target variables.
- Lower MSE: indicating less error between the predicted and the actual values.
- Better Accuracy: indicating the major percentage of correct predictions.

The metrics calculated and confusion matrix were:

```
Naive Bayes Model over TRAIN Dataset:
Naive Bayes Model, R2 Score: 0.11559552533450312
Naive Bayes Model, MSE:      0.19444444444444445
Naive Bayes Model, Accuracy: 0.8055555555555556

Naive Bayes Model over TEST  Dataset:
Naive Bayes Model, R2 Score: 0.3843971631205674
Naive Bayes Model, MSE:      0.11290322580645161
Naive Bayes Model, Accuracy: 0.8870967741935484
```

```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92        47
           1       0.75      0.80      0.77        15

    accuracy                           0.89        62
   macro avg       0.84      0.86      0.85        62
weighted avg       0.89      0.89      0.89        62


Text(-4.277777777777779, 0.5, 'Actual')
```



To conclude, the notebook elaborated and attached to this delivery contains the steps from data preprocessing to perform the model evaluation for a Campus Placement Prediction, and all of the steps were documented by comments.