PREDICTING STUDENT RECRUITMENT DURING CAMPUS PLACEMENTS

Assignment 2

Professor Ishant Gupta
Lambton College

Laura Osorio Bermudez
C0917325

Repository Link:

https://github.com/Lmosoriob/CampusPlacementPrediction_Assignment2

Overview

The dataset used for this analysis has been sourced from Kaggle. It can be found at the following link: Campus Recruitment Prediction (Course Project) | Kaggle.

The dataset, specifically the train.csv file, includes various attributes related to student demographics, academic performance, and placement outcomes. This information is crucial for analyzing factors that influence campus recruitment success.

1. Dataset Description

The train.csv file contains the following important columns:

Variable	Type	Description	Importance
ssc_p	Numerical	Percentage score obtained in the secondary school examination.	High
hsc_p	Numerical	Percentage score obtained in the higher secondary examination.	High
degree_p	Numerical	Percentage score obtained in the degree program.	High
etest_p	Numerical	Percentage score obtained in the entrance test.	High
workex	Categorical	Indicates whether the student has prior work experience (e.g., Yes, No).	High
specialisation	Categorical	Refers to the specialization pursued during the degree (e.g., Mkt&Fin, Mkt&HR).	Medium to High
status	Categorical	Represents the placement status of the student (e.g., Placed, Not Placed).	Essential for analysis
salary	Numerical	Indicates the salary offered to students who are placed; may contain null values for those not placed.	Important for insights

2. Data Preprocessing Steps

To optimize model compatibility and performance, several preprocessing steps were conducted, as detailed below

1. EDA:
   - **Data Inspection**: The dataset was loaded using pandas, and initial exploration involved functions like head(), info(), and describe() to understand its structure, summary statistics, and general characteristics.

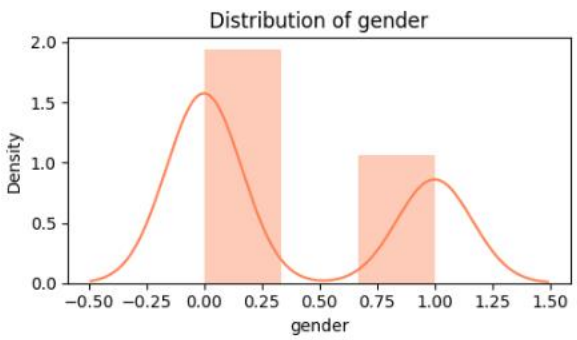
	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	0	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	0	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	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

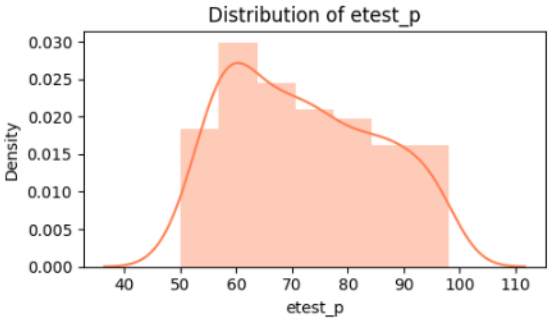
   - **Data Types**: Columns were found to include both numerical data (e.g., scores and salary) and categorical data (e.g., gender, degree type), essential for determining appropriate preprocessing strategies.

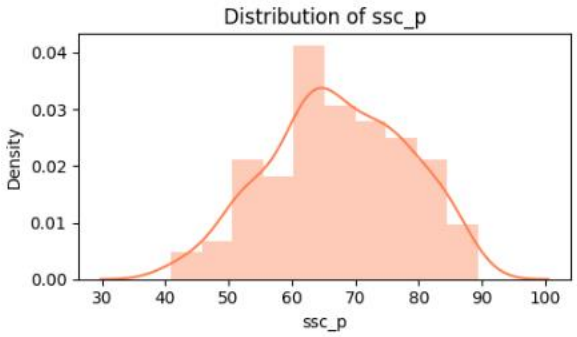
```
[7]  df.info()
```

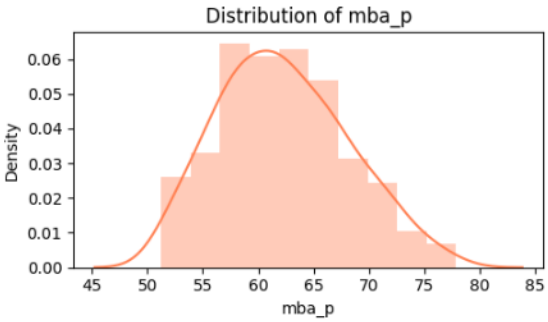
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   gender          215 non-null    int64
 1   ssc_p           215 non-null    float64
 2   ssc_b           215 non-null    object
 3   hsc_p           215 non-null    float64
 4   hsc_b           215 non-null    object
 5   hsc_s           215 non-null    object
 6   degree_p        215 non-null    float64
 7   degree_t        215 non-null    object
 8   workex          215 non-null    object
 9   etest_p         215 non-null    float64
 10  specialisation  215 non-null    object
 11  mba_p           215 non-null    float64
 12  salary          148 non-null    float64
 13  status          215 non-null    object
dtypes: float64(6), int64(1), object(7)
memory usage: 23.6+ KB
```

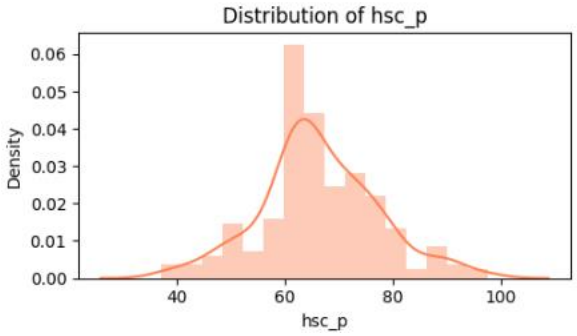
   - **Feature Distribution**: Histograms and box plots were utilized to visualize the distribution of numerical features (such as ssc_p, hsc_p, and salary). This helped identify skewness, outliers, and the overall range of values.

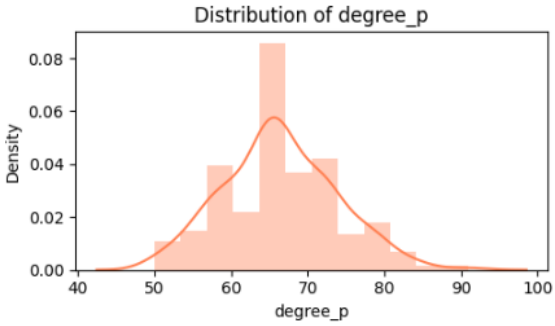










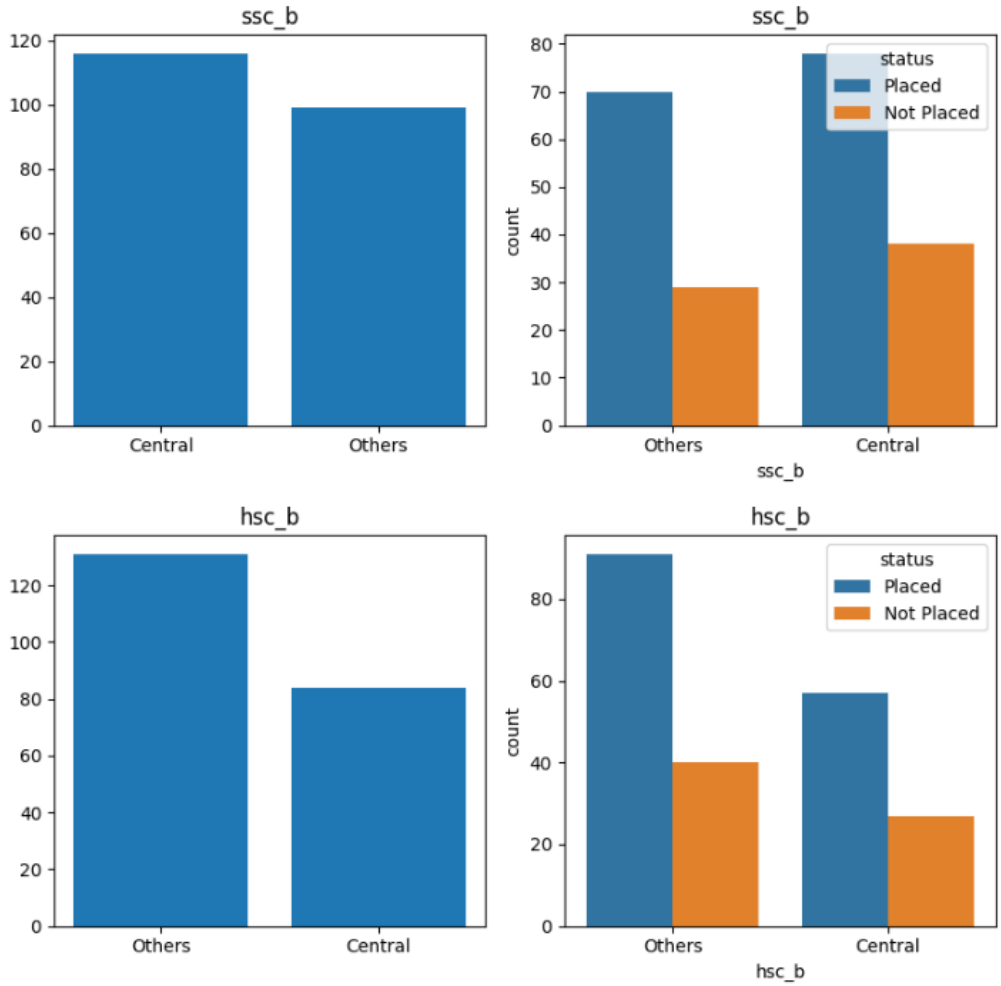


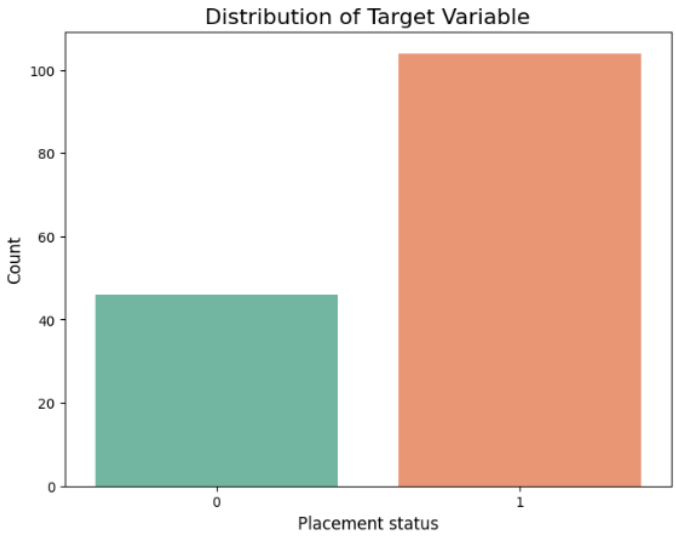
Most of the columns shows a bell-shaped distribution, indicating that the data is symmetrically distributed around a central value. However, the salary column stands out as it is right-skewed. This means that while most salaries are concentrated on the lower end, there are some higher salaries that pull the average up, resulting in an asymmetric distribution.

The categorical variables indicate a clear trend where the ratio of placed to not placed students is typically 2:1. This means that for every 2 students who secure placement, there is 1 who does not. However, there are a few notable exceptions:

- In the specialization category, the Mkt&HR specialization has a nearly equal distribution between placed and not placed students. On the other hand, the Mkt&Fin specialization shows a ratio of about 4:1, indicating that for every 4 students placed, there is only 1 who is not.
- Interestingly, students with other degrees show a higher number of those not placed compared to those who are placed, which is unusual for this dataset. A similar situation occurs within the Arts category, where the number of placed and not placed students is about the same.

Overall, the dataset displays a significant imbalance, with many more students placed than those who are not.





The distribution reveals a significant disparity between the number of "Placed" students and "Not Placed" students, indicating a moderate class imbalance. This imbalance suggests that the model may be biased toward predicting "Placed" more frequently, as it has a greater number of examples to learn from. Such a tendency could affect the model's performance, particularly in accurately predicting the minority class, "Not Placed."

1. **Encoding Categorical Variables**

```
[13] {column: list(df[column].unique()) for column in df.select_dtypes('object').columns}
```

```
{'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']}
```

This output shows the values that are in each categorical column. The dataset's categorical columns, which include two columns with three categories and others with two categories, were appropriately handled. Columns with two categories were encoded using label encoding, while those with three categories were processed through one-hot encoding. Additionally, the target variable was manually encoded first to ensure accurate label assignment, following the specified order. This thorough approach has facilitated the preparation of the dataset for further analysis and modeling.

Categorical features were transformed into numerical values to ensure compatibility with machine learning algorithms.

3. Standardization: Features were standardized using StandardScaler to ensure a mean of 0 and standard deviation of 1. This prevented features with larger scales from dominating model learning, improving efficiency and accuracy for models sensitive to feature scaling, such as Logistic Regression and XGBoost.

4.. Dataset Splitting

The dataset was divided into training and testing subsets, with 70% allocated for training and 30% for testing. This split was performed before addressing missing values or applying any resampling techniques to prevent data leakage, ensuring that preprocessing was limited to the training dataset.

```
[19] df.isnull().sum()
```

	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
salary	67
status	0
hsc_s_Arts	0
hsc_s_Commerce	0
hsc_s_Science	0
degree_t_Comm&Mgmt	0
degree_t_Others	0
degree_t_Sci&Tech	0

dtype: int64

3. Handling Missing Data

Missing values were identified exclusively in the salaries column. To address these gaps, target imputation was implemented:

- The average salary was calculated based on the degree classification within the training dataset.
- These average salary figures were subsequently used to fill the missing entries in the testing dataset, maintaining consistency by relying on characteristics derived from the training data.

There are 67 null values in the column named 'salary'.

The distribution reveals a significant disparity between the number of "Placed" students and "Not Placed" students, indicating a moderate class imbalance. This imbalance suggests that the model may be biased toward predicting "Placed" more frequently, as it has a greater number of examples to learn from. Such a tendency could affect the model's performance, particularly in accurately predicting the minority class, "Not Placed."

It is evident that all entries with null values in the salary column correspond to students who are not placed. Given the imbalance in the dataset, retaining these records is essential rather than discarding them. Therefore, target imputation will be employed to estimate the missing salary values based on the salaries associated with the degree column. This approach ensures that valuable information is preserved while addressing the missing data.

To impute missing values in the salary column based on the degree_t column, the mean salary for each unique degree_t category in the training data is calculated. These mean values are then applied to fill missing salary entries in both the training and test datasets, using the degree_t value as a reference. This method ensures that missing salary values are replaced consistently with the corresponding category's average, thereby preserving the relationship between degree_t and salary while preventing data leakage.

```
[23] degree_salary_means = X_train.groupby('degree_t')['salary'].mean()
     degree_salary_means
```

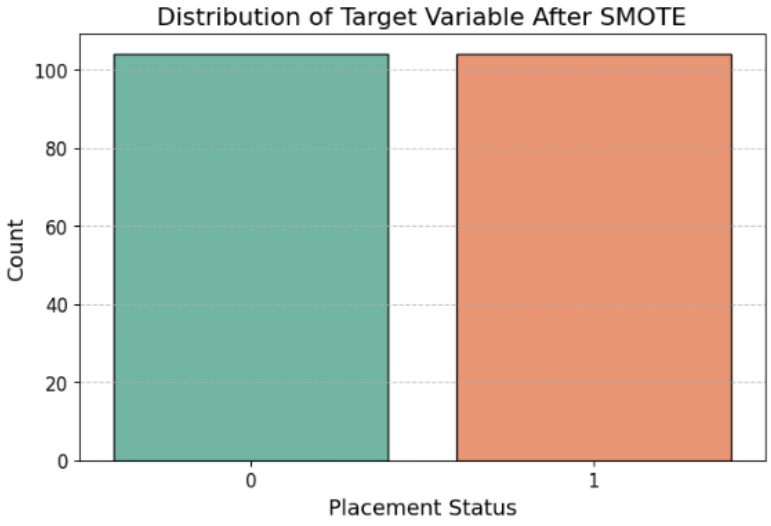
degree_t	salary
Comm&Mgmt	284041.666667
Others	300000.000000
Sci&Tech	305793.103448

dtype: float64

6. Class Imbalance Mitigation

To address the class imbalance in the target variable, the Synthetic Minority Over-sampling Technique (SMOTE) was applied solely to the training dataset:

Implementing SMOTE (Synthetic Minority Over-sampling Technique) solely in the training dataset without missing values is essential for effectively addressing class imbalance while maintaining data integrity. By generating synthetic samples for the minority class in the training data, SMOTE helps balance class distribution, which enhances the model's ability to learn from underrepresented classes. Importantly, applying SMOTE only to the training set prevents data leakage, ensuring that the test dataset remains untouched and unbiased, allowing for a more accurate evaluation of the model's performance on unseen data. Additionally, working with a complete dataset free of missing values during this process ensures the reliability of the synthetic samples generated. Overall, this approach leads to a robust and valid model evaluation while addressing the challenges of class imbalance.



Model Selection

For this task, Logistic Regression, Random Forest, and XGBoost were selected and combined using a Voting Classifier. These models were chosen for their effectiveness in binary classification and their ability to handle both linear and non-linear relationships.

Logistic Regression is a straightforward and interpretable model that is effective for binary classification. It serves as a solid baseline for performance, particularly suited for managing linear relationships. Key hyperparameters include a regularization parameter (C=1) and an L2 penalty to mitigate the risk of overfitting.

Random Forest is an ensemble method that excels at capturing non-linear relationships by constructing multiple decision trees and averaging their predictions. This approach enhances accuracy and provides insights into feature importance, making it robust in imbalanced datasets. Optimal hyperparameters include n_estimators=500, max_depth=None, and max_features='sqrt', which help create diverse trees.

XGBoost, an optimized version of gradient boosting, is renowned for its accuracy and efficiency in handling complex relationships. It constructs decision trees sequentially, correcting the shortcomings of previous trees while incorporating regularization to avoid overfitting. Effective hyperparameters for XGBoost include learning_rate=0.01, max_depth=6, and n_estimators=500, which ensure controlled learning.

The **Voting Classifier** aggregates the strengths of Logistic Regression, Random Forest, and XGBoost, improving generalization by reducing individual model errors. It employs hard voting (majority voting) to consolidate predictions, leveraging the unique advantages of each model for enhanced classification performance.

Model training and Evaluation

Random Forest Classifier Performance Analysis

The Random Forest Classifier has shown strong performance in predicting student placement, effectively identifying most placed students with minimal false negatives. However, the model does exhibit some bias, evident in the moderate number of false positives, where unplaced students are misclassified as placed. This issue is likely influenced by the class imbalance present in the dataset.

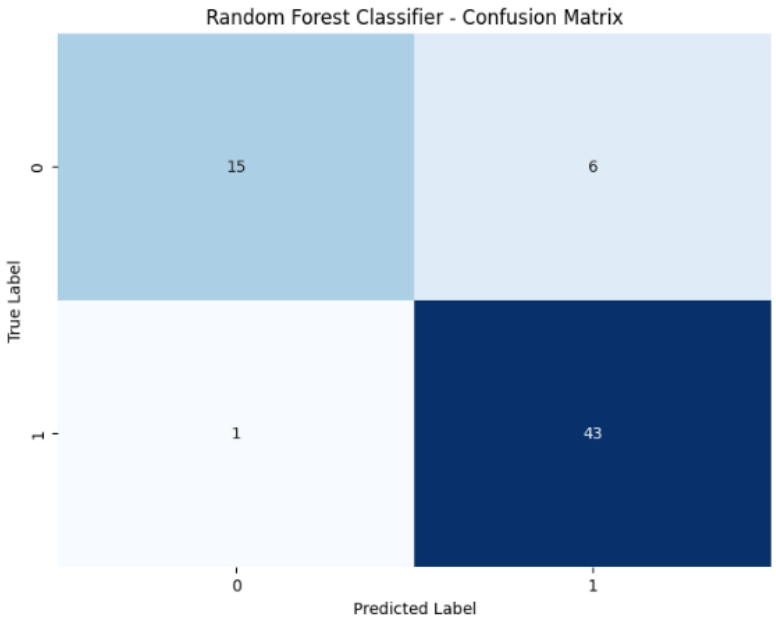
Model Metrics

The model was fine-tuned using specific hyperparameters:

- max_depth: Set to None, allowing trees to split until pure leaf nodes are reached, which enhances fit but may risk overfitting.
- max_features: Set to 'sqrt', ensuring that only the square root of features is considered for splits, which helps reduce overfitting.
- n_estimators: Set to 500, utilizing 500 decision trees to enhance stability and performance, though this increases computational time.

The model achieved an impressive F1 score of 0.9584, indicating a strong balance between precision and recall. The classification report revealed that for Class 0 (Not Placed), precision was 0.94, recall was 0.71, and the F1 score was 0.81. For Class 1 (Placed), precision was 0.88, recall was 0.98, and the F1 score was 0.92, resulting in an overall accuracy of 89%.

Confusion Matrix



When examining average metrics, the model displayed a macro average precision of 0.91 and recall of 0.85, indicating consistent performance across classes. The weighted averages showed precision at 0.90 and recall at 0.89, reflecting balanced performance considering class distribution.

Training Insights

On the training side, the model achieved perfect scores for precision, recall, F1 score, and accuracy (all 1.0). While this indicates excellent performance, it suggests overfitting, where the model memorizes the training data rather than generalizing well to unseen data. However, the test set performance indicates that the model generalizes well, particularly for Class 1, although improvements are needed for Class 0.

Overall, while the Random Forest model is effective in predicting student placements, it could benefit from further tuning to address class imbalance and reduce misclassification of unplaced students. Future work may involve techniques like cross-validation and hyperparameter adjustments to enhance recall for Class 0 without compromising performance for Class 1.

Analysis of the XGBoost Classifier Performance

The XGBoost model has shown impressive results in predicting whether students will be placed or not. It does an excellent job of accurately identifying the majority of placed students, with a low rate of misclassifying them as unplaced.

Looking at the detailed classification report, we can see that the model has a strong performance across both classes:

For the 'Not Placed' class:

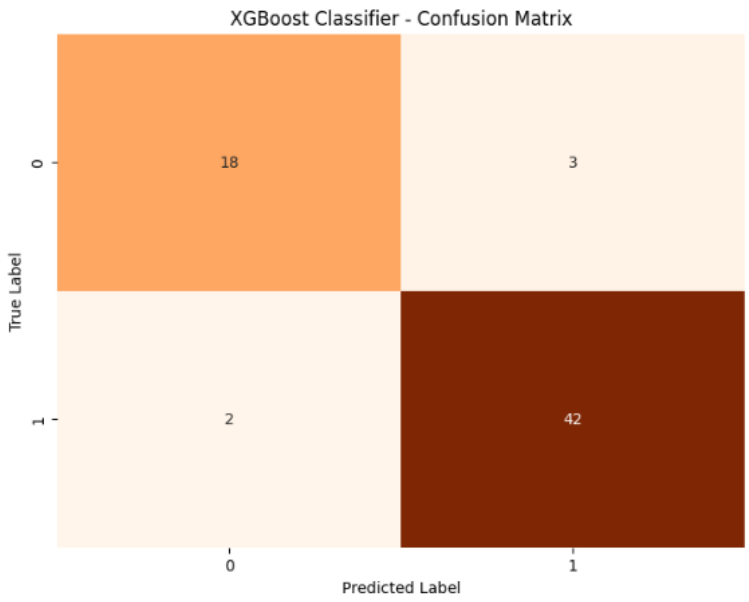
- Precision is 90%, meaning 90% of the students it predicted as 'Not Placed' were actually in that category.
- Recall is 86%, indicating the model correctly identified 86% of the actual 'Not Placed' students.
- The F1-score, which balances precision and recall, is a solid 0.88.

For the 'Placed' class:

- Precision is 93%, so 93% of the students it predicted as 'Placed' were correctly classified.
- Recall is even higher at 95%, demonstrating the model's strong ability to detect placed students.
- This results in an excellent F1-score of 0.94 for the 'Placed' class.

Overall, the model achieved an impressive accuracy of 92% on the test data. The macro average metrics (precision 0.92, recall 0.91, F1-score 0.91) and weighted averages (all 0.92) further confirm the model's balanced and robust performance across both classes.

Confusion Matrix:



It shows 18 true negatives (correctly predicted as 'Not Placed'), 42 true positives (correctly predicted as 'Placed'), 2 false positives (predicted as 'Placed' but actual 'Not Placed'), and 3 false negatives (predicted as 'Not Placed' but actual 'Placed').

This matrix reinforces the model's strength in accurately identifying placed students, with only a small number of misclassifications.

However, the model's training metrics were perfect across the board (precision, recall, F1-score, and accuracy all 1.0), indicating potential overfitting. This means the model may have memorized the training data rather than learning to generalize well to new, unseen examples.

To address this, future efforts could explore techniques like cross-validation, further hyperparameter tuning, and strategies to handle the class imbalance in the data. This would help ensure the model maintains its strong performance on real-world, unseen data.

Overall, the XGBoost classifier demonstrates a high degree of effectiveness in predicting student placements, though continued refinement could unlock even better results.

Analysis of the Logistic Regression Model

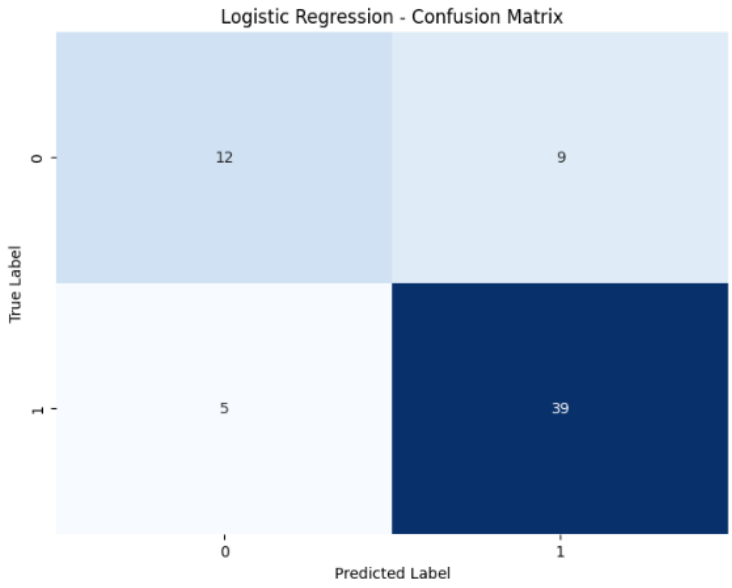
The Logistic Regression model has demonstrated reasonable performance in predicting student placement, though with some notable limitations that require further attention.

Looking at the classification report, the model appears to be stronger at identifying students who were placed, with a precision of 81% and recall of 89% for the 'Placed' class. This suggests the model is quite effective at correctly recognizing placed students.

However, the model struggles more with the 'Not Placed' class, with a precision of only 71% and a recall of 57%. This means the model is not as reliable at accurately detecting students who were not placed.

The overall accuracy of the model on the test data is 78%, which is decent but leaves room for improvement. The macro average metrics (precision 0.76, recall 0.73, F1-score 0.74) indicate the model's performance is not as consistent across both classes.

Confusion Matrix:



Examining the confusion matrix:

- True Negatives (Correctly predicted as 'Not Placed'): 12
- True Positives (Correctly predicted as 'Placed'): 39
- False Positives (Predicted as 'Placed' but Actual 'Not Placed'): 9

- False Negatives (Predicted as 'Not Placed' but Actual 'Placed'): 5

The matrix shows the model is better at correctly identifying placed students (true positives) than it is at correctly identifying not placed students (true negatives). The higher number of false positives (9) compared to false negatives (5) suggests the model may have a tendency to over-predict the 'Placed' class.

This pattern is consistent with the classification report, where the model's precision and recall are stronger for the 'Placed' class than the 'Not Placed' class.

The training metrics, which show high precision, recall, F1-score, and accuracy, raise concerns about potential overfitting. This means the model may have memorized the training data rather than learning more generalized patterns that would translate well to new, unseen examples.

To address these issues, future work could explore techniques like cross-validation, regularization, and adjusting the model's hyperparameters to improve its ability to generalize. Investigating feature engineering approaches or trying alternative models may also help enhance the Logistic Regression model's overall predictive performance and balance across both classes.

Evaluation of the Voting Classifier's Performance

The Voting Classifier has demonstrated impressive results in predicting whether students will be placed or not. It does an excellent job of accurately identifying students in both the 'Placed' and 'Not Placed' categories.

Looking at the detailed classification report, we can see the model has a strong performance across both classes:

For the 'Not Placed' class:

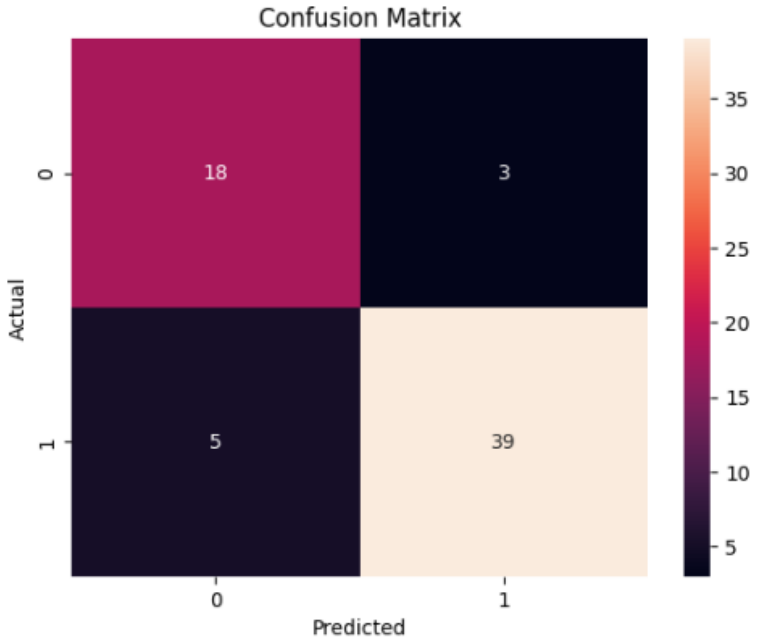
- Precision is 78%, meaning 78% of the students it predicted as 'Not Placed' were actually in that category.
- Recall is 86%, indicating the model correctly identified 86% of the actual 'Not Placed' students.
- The F1-score, balancing precision and recall, is a solid 0.82.

For the 'Placed' class:

- Precision is 93%, so 93% of the students it predicted as 'Placed' were correctly classified.
- Recall is 89%, demonstrating the model's strong ability to detect placed students.
- This results in an excellent F1-score of 0.91 for the 'Placed' class.

Overall, the model achieved an impressive accuracy of 88% on the test data. The macro average metrics (precision 0.86, recall 0.87, F1-score 0.86) and weighted averages (all 0.88) further confirm the model's balanced and robust performance across both classes.

Confusion Matrix:



Examining the confusion matrix provides additional insights:

- True Negatives (Correctly predicted as 'Not Placed'): 18
- True Positives (Correctly predicted as 'Placed'): 39
- False Positives (Predicted as 'Placed' but Actual 'Not Placed'): 3
- False Negatives (Predicted as 'Not Placed' but Actual 'Placed'): 5

The matrix shows the model is highly effective at correctly identifying both 'Placed' and 'Not Placed' students, with a low number of misclassifications in both categories. This balanced performance is a key strength of the Voting Classifier.

The training metrics for the model were also very strong, with perfect precision and high recall, F1-score, and accuracy. However, this raises some concerns about potential overfitting, where the model may have memorized the training data rather than learning more generalized patterns.

To address this, further testing and evaluation on new, unseen data would be important. Techniques like cross-validation could also help ensure the model maintains its impressive performance on real-world applications.

Overall, the Voting Classifier demonstrates excellent predictive capabilities for determining student placements. Its balanced and accurate performance across both classes makes it a highly promising model for this task.

Based on the detailed evaluation, the **Voting Classifier** clearly **stands out as the best option for predicting** student placements. It consistently delivers strong results for both 'Placed' and 'Not Placed' students, achieving high scores in precision, recall, and F1 measures.

With an accuracy of 88% on the test data, along with solid metrics for both classes, the Voting Classifier proves to be the most reliable choice among the models we examined. While some other models did well in one area, the Voting Classifier shows a balanced performance across the board.

Another notable point is that its test metrics are actually better than its training metrics. This suggests that the model has learned to generalize well, rather than just memorizing the training data. This ability makes it a great candidate for real-world use and future improvements.