Campus Recruitment

Sara Akbarzadeh

May 2025

Abstract

This report presents the process and results of applying machine learning techniques to a classification problem using a dataset with various features. We applied several models, including Logistic Regression, Naive Bayes, Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN), to predict the target variable. After performing exploratory data analysis (EDA), data cleaning, and feature engineering, we evaluated the models based on accuracy, ROC AUC, and PR AUC scores. Our findings indicate that Naive Bayes performed the best initially, but further improvements through regularization and feature engineering were applied to enhance the Logistic Regression model. Hyperparameter tuning and cross-validation were also performed to fine-tune the models for better performance.

1 Introduction

This project focuses on predicting campus placement success for students based on various factors such as academic performance, test scores, gender, specialization, and internship experience. The dataset used for this analysis is sourced from the Kaggle competition "Factors Affecting Campus Placement," which provides detailed information about students from various universities who have participated in campus recruitment drives.

By leveraging machine learning algorithms, the goal is to predict whether a student will be successfully placed in a company based on the aforementioned factors. This project explores various data preprocessing and model evaluation techniques, such as feature engineering, regularization, and cross-validation, to improve the performance and accuracy of predictive models.

2 Dataset

The dataset used in this project is sourced from Kaggle and is publicly available.

This dataset contains records of students' academic and personal backgrounds, aiming to identify factors that influence successful campus placement. It includes attributes such as gender, educational background (both secondary and higher secondary), specialization, work experience, and various performance metrics (e.g., percentages in SSC, HSC, degree, and MBA). The target variable indicates whether the student was placed or not.

The dataset consists of 215 instances and 15 features. It serves as a realistic and practical example for evaluating classification algorithms in the context of educational and employment data. Prior to model training, the dataset underwent preprocessing steps such as handling missing values, encoding categorical variables, and normalizing numeric features where necessary.

3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) plays a vital role in understanding the dataset before applying machine learning models. It helps identify patterns, detect anomalies, and uncover relationships between variables. EDA also aids in cleaning the data, identifying features for further analysis, and ensuring that assumptions for machine learning models are met.

In this project, EDA was conducted to explore the dataset and gain insights into its structure. Various techniques were used, such as summary statistics, visualizations, and correlation analysis. We performed multiple visualizations to examine the distribution of features and identify any potential outliers or relationships among them.

We performed several EDA techniques, as shown in Figures 1 to 7, to better understand the dataset. These visualizations helped us make informed decisions regarding data cleaning, feature selection, and model preparation.

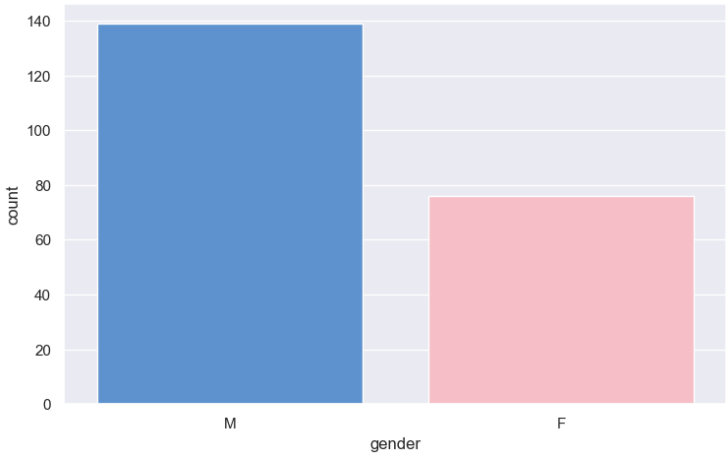

Figure 1: EDA

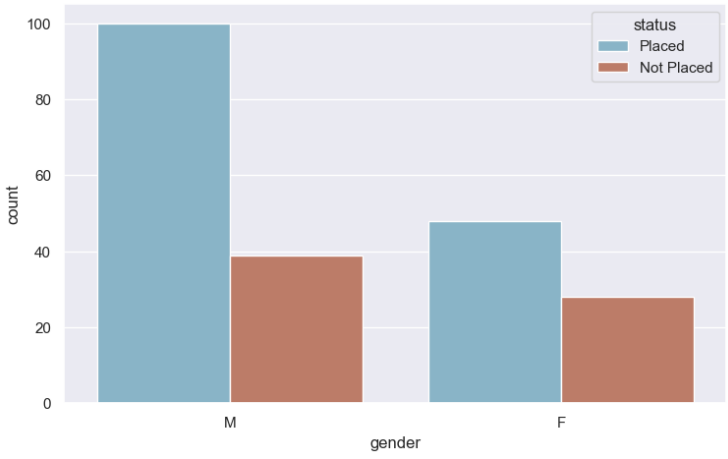

Figure 2: EDA

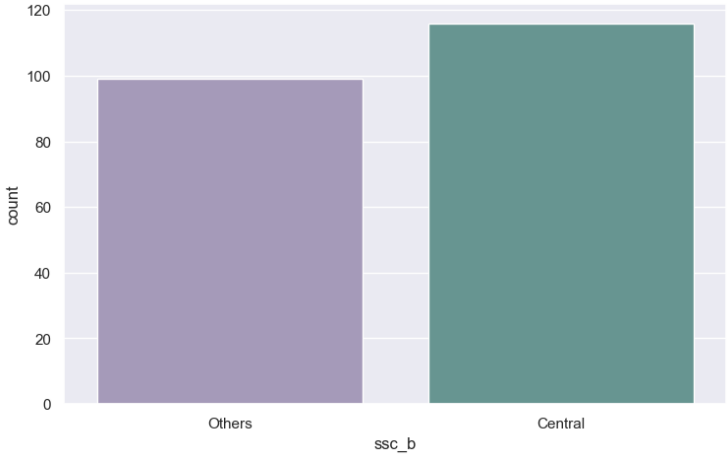

Figure 3: EDA

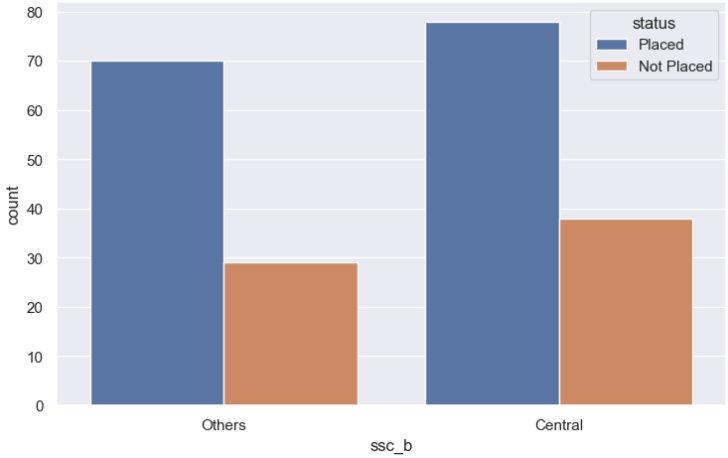

Figure 4: EDA

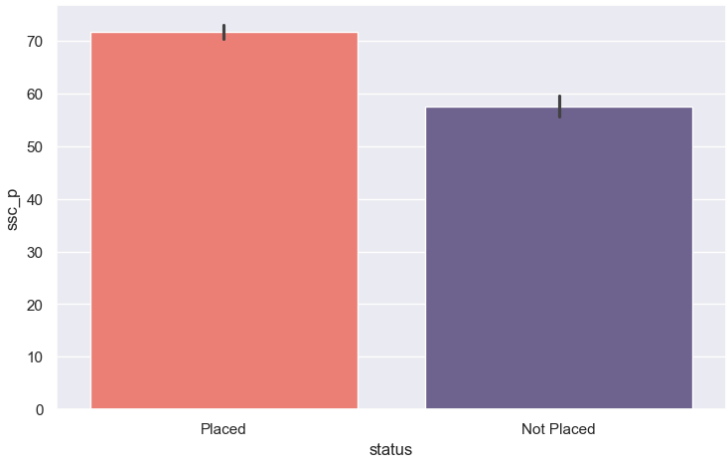

Figure 5: EDA

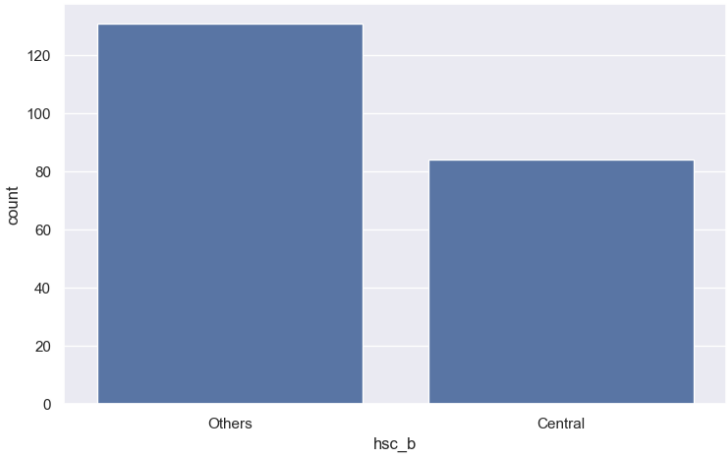

Figure 6: EDA

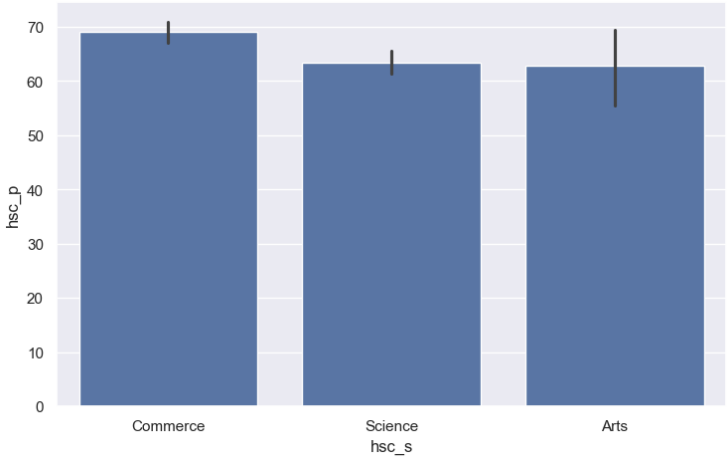

Figure 7: EDA

4 Data Cleaning

Data cleaning is a crucial preprocessing step that ensures the quality and reliability of the dataset before applying machine learning algorithms. In this project, several cleaning steps were performed to prepare the data for modeling.

First, we examined the dataset for missing values. Most columns were complete and had no missing entries. The only exception was the `salary` column, which contained missing values. However, upon inspection, we found that all individuals with missing `salary` values were labeled as `unemployed` in the `status` column. Thus, the missing values in `salary` were meaningful and did not require imputation or removal.

Next, categorical variables were converted into numerical format using appropriate encoding techniques to make the data compatible with machine learning models. This step included transforming categories like `gender`, `workex`, and `specialisation` into numerical representations.

After careful analysis, we decided to drop two columns: `degree_t` (type of undergraduate degree) and `hsc_s` (high school subject specialization). These features were found to have either low correlation with the target variable or redundant information relative to other columns.

4.1 Outlier Handling

Outliers can significantly skew the results of machine learning models, particularly those sensitive to scale and variance. To address this, we performed an outlier detection process focused on the numerical features of the dataset.

We began by visualizing the distribution of each numerical variable using boxplots to identify any obvious anomalies. Then, we applied the Interquartile Range (IQR) method to formally detect outliers. This involved calculating the IQR (the range between the first and third quartiles) and flagging any values falling below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ as outliers.

The number of outliers in each relevant column was printed for verification. Once identified, we used a custom function to remove these outliers from the dataset. This ensured the data used for training our models was more robust and not unduly influenced by extreme values.

5 Modeling

In the modeling phase, our primary goal was to prepare the dataset for the application of machine learning algorithms. We began by separating the target variable `status` from the rest of the dataset. The remaining columns were stored in a feature matrix denoted as X, while the target labels were stored in y.

To evaluate the models fairly and prevent overfitting, we split the dataset into training and testing subsets using an 80-20 ratio. This means that 80% of the data was used for training the models, and 20% was held out for testing their performance on unseen data.

Before training the models, feature scaling was performed using the `StandardScaler` from `scikit-learn`. Standardization is a critical preprocessing step, especially for algorithms that are sensitive to the scale of input data. It transforms each feature to have a mean of 0 and a standard deviation of 1, ensuring that features with larger numeric ranges do not dominate the learning process.

Importantly, the scaling transformation was fitted only on the training set to prevent data leakage. The same transformation was then applied to both the training and test sets.

After this process, the shapes of X_{train}, X_{test}, y_{train}, and y_{test} were printed to verify that the data was split correctly and ready for model training and evaluation.

5.1 Logistic Regression

Logistic Regression was implemented as a baseline linear classifier for the binary classification task. After training the model, performance was evaluated on both training and test sets using various metrics.

Train Set: The model achieved an accuracy of 91%, with a precision of 0.92 and recall of 0.95 for class 1 (placed), indicating strong performance on the training data.

Test Set: However, on the test data, accuracy dropped to 79%. The precision for class 0 (not placed) was only 0.67, and its recall was even lower at 0.36. On the other hand, class 1 maintained good metrics with a precision of 0.81 and recall of 0.94. This performance difference hints at potential overfitting on the training set.

Figure 8 shows the confusion matrix, which visualizes how well the model distinguishes between placed and not placed students. Figures 9 and 10 present the ROC and Precision-Recall curves, respectively, demonstrating that the model is better at predicting class 1 than class 0.

5.2 Naive Bayes

Naive Bayes was applied as a probabilistic model assuming feature independence. Despite its simplicity, the model produced strong results on both training and test datasets.

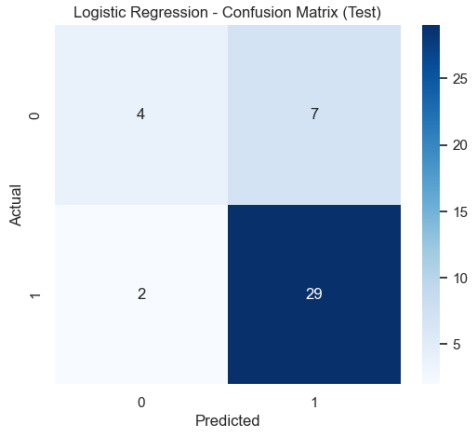

Figure 8: Confusion Matrix for Logistic Regression

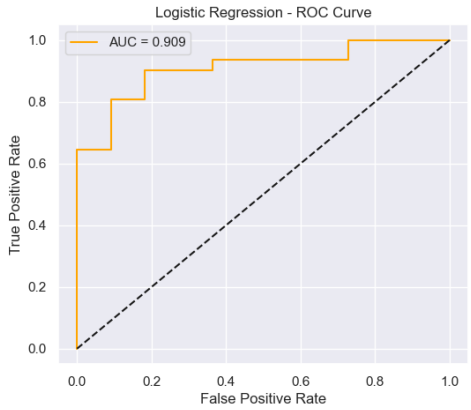

Figure 9: ROC Curve for Logistic Regression

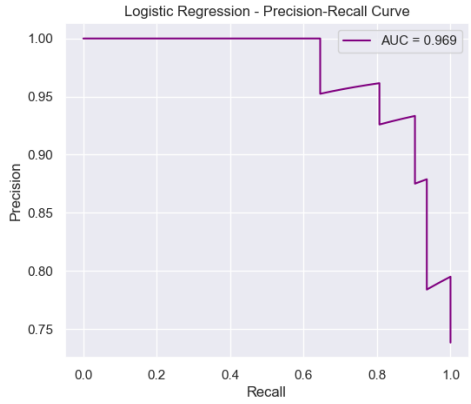

Figure 10: Precision-Recall Curve for Logistic Regression

Train Set: The model achieved an overall accuracy of 85%, with a recall of 0.90 and precision of 0.89 for the placed class (class 1). Performance for the not placed class (class 0) was slightly lower, with a recall of 0.75.

Test Set: On unseen data, Naive Bayes demonstrated even better generalization. The accuracy improved to 88%, with the placed class reaching a precision of 0.93 and recall of

0.90. Notably, the not placed class also had a high recall of 0.82, indicating that the model handled both classes well.

Figure 11 shows the confusion matrix, which illustrates the balance in prediction between the two classes. Figures 12 and 13 present the ROC and Precision-Recall curves, respectively, showing consistent and stable behavior across different thresholds.

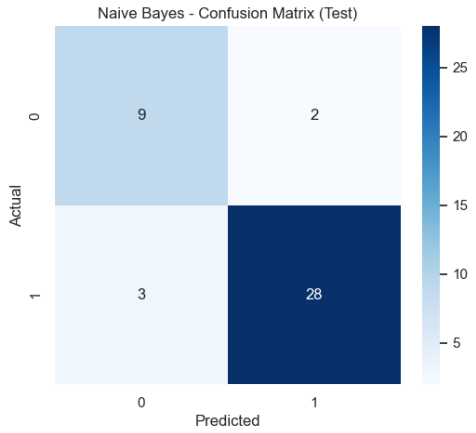

Figure 11: Confusion Matrix for Naive Bayes

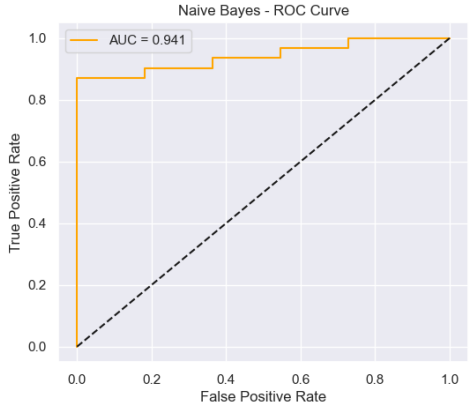

Figure 12: ROC Curve for Naive Bayes

5.3 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) was used as a linear classifier that projects data in a way that maximizes class separability. It assumes multivariate normality and equal covariance among classes.

Train Set: LDA performed well on the training set with an overall accuracy of 90%. The recall and precision for the placed class (class 1) were both high at 0.93, while the not placed class (class 0) had slightly lower but still strong metrics.

Test Set: The performance on the test set dropped to 79% accuracy. The placed class retained good recall (0.94), but the not placed class saw a significant drop in recall (0.36), indicating a bias toward predicting placement.

As seen in Figure 14, the confusion matrix shows that many not placed cases were misclassified. Figures 15 and 16 show the ROC and Precision-Recall curves, respectively, which help visualize the trade-offs in classification performance.

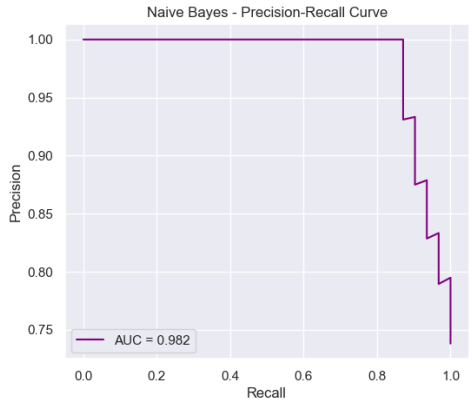

Figure 13: Precision-Recall Curve for Naive Bayes

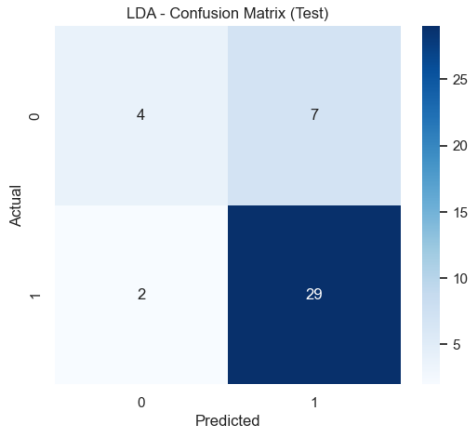

Figure 14: Confusion Matrix for LDA

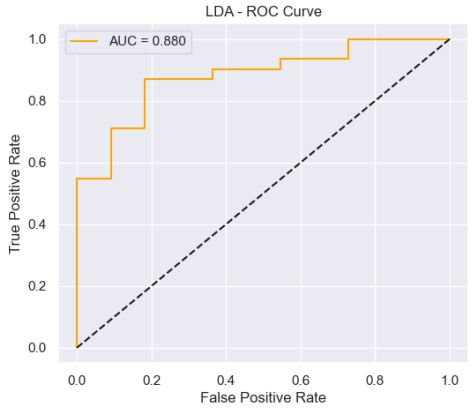

Figure 15: ROC Curve for LDA

5.4 Support Vector Machine (SVM) [Bonus Model]

Support Vector Machine (SVM) is included as a bonus model due to its powerful ability to find the optimal separating hyperplane, especially in high-dimensional spaces. SVM can be particularly effective for classification tasks where the classes are not linearly separable.

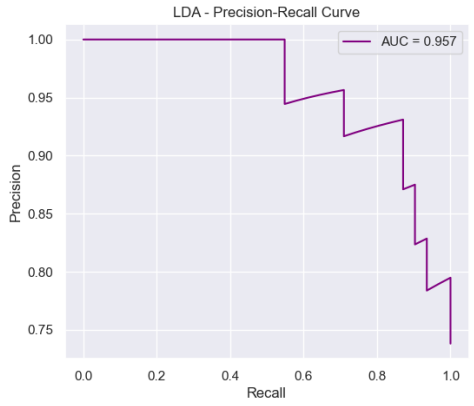

Figure 16: Precision-Recall Curve for LDA

Train Set: The SVM model achieved an impressive accuracy of 95% on the training data. It perfectly recalled all placed students (class 1) and showed a high precision (1.00) for not placed students (class 0), indicating a strong fit to the training set.

Test Set: On the test set, the performance dropped to 79% accuracy. While the placed class (1) still had high recall (0.94), the not placed class had poor recall (0.36), suggesting that the model may be overfitting to the training data.

The confusion matrix in Figure 17 highlights the misclassification of not placed students. Figures 18 and 19 display the ROC and Precision-Recall curves, which provide further insights into the model's threshold behavior and class imbalance sensitivity.

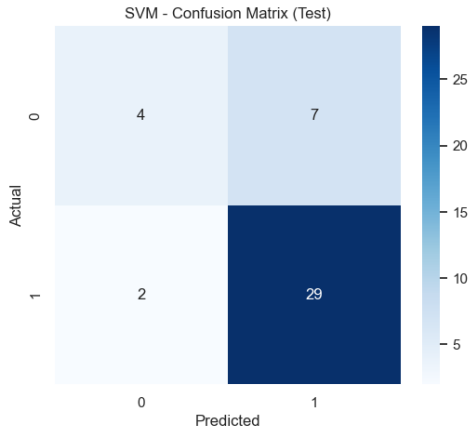

Figure 17: Confusion Matrix for SVM

5.5 K-Nearest Neighbors (KNN) [Bonus Model]

K-Nearest Neighbors (KNN) was explored as a bonus model due to its simplicity and non-parametric nature, which can be useful for classification when the decision boundary is non-linear. KNN classifies a data point based on the majority label among its k closest neighbors.

Train Set: The model showed strong results on the training data, achieving an overall accuracy of 91%. It achieved very high recall for placed students (0.99), but its recall for not placed students (0.73) was comparatively lower, reflecting an imbalance in sensitivity.

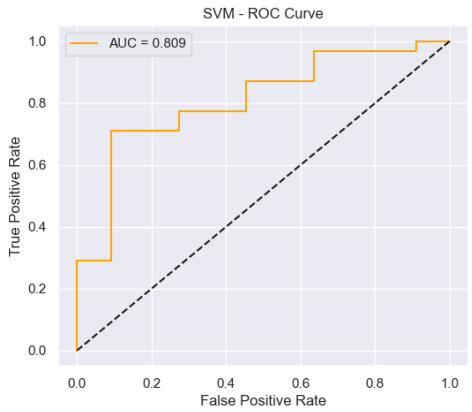

Figure 18: ROC Curve for SVM

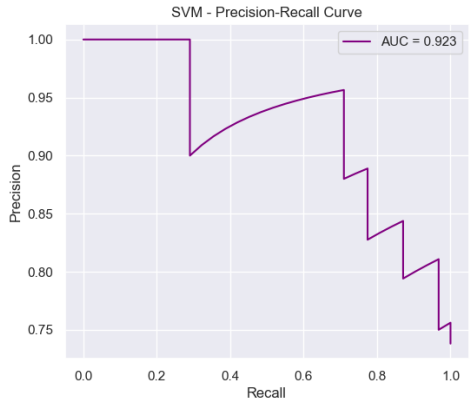

Figure 19: Precision-Recall Curve for SVM

Test Set: On the test data, KNN performance dropped notably to 74% accuracy. It had difficulty correctly identifying the not placed class, with a recall of just 0.18. This highlights a common limitation of KNN in generalizing to unseen data, especially when the class distribution is skewed or the feature space is not well separated.

Figure 20 presents the confusion matrix, emphasizing the misclassification of not placed students. Figures 21 and 22 show the ROC and Precision-Recall curves, which further illustrate the model's lower confidence and weaker discriminative ability on the test set.

5.6 Summary of All Models

The following table summarizes the performance of all five models in terms of training and test accuracy, along with ROC AUC and PR AUC scores:

Model	Train Accuracy	Test Accuracy	ROC AUC	PR AUC
Naive Bayes	0.8537	0.8810	0.9413	0.9817
Logistic Regression	0.9085	0.7857	0.9091	0.9688
LDA	0.9024	0.7857	0.8798	0.9566
SVM (Bonus)	0.9451	0.7857	0.8094	0.9227
KNN (Bonus)	0.9085	0.7381	0.8299	0.9403

Table 1: Performance Comparison of All Models

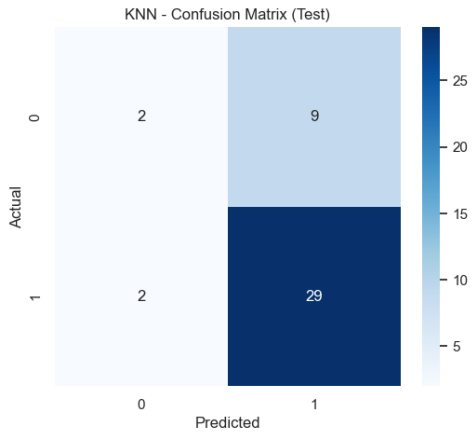

Figure 20: Confusion Matrix for KNN

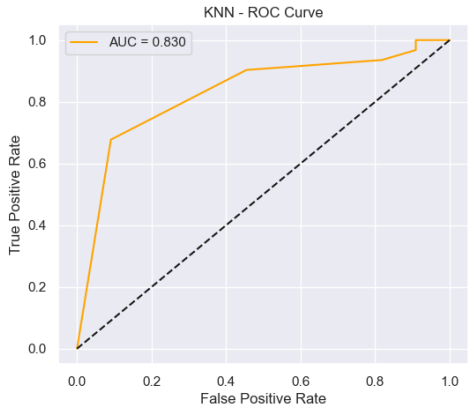

Figure 21: ROC Curve for KNN

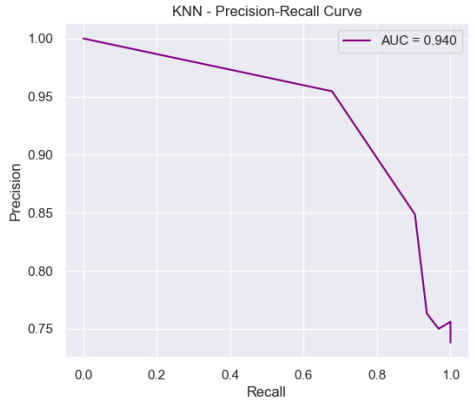

Figure 22: Precision-Recall Curve for KNN

Key Insights

- **Naive Bayes** achieved the highest **Test Accuracy** and also led in both **ROC AUC** and **PR AUC**, making it the most effective model for this classification problem.

- **Logistic Regression** and **LDA** offered similar performance but were outperformed by Naive Bayes in all test metrics.
- **KNN**, although performing decently on the training set, exhibited the lowest test accuracy, indicating issues in generalization.
- **SVM** showed a very high training accuracy, but the test performance did not reflect this, suggesting possible *overfitting*.

Based on these metrics and generalization performance, **Naive Bayes** is selected as the most reliable model for this task.

6 Model Enhancement

Although the **Naive Bayes** classifier demonstrated the best overall performance in terms of *Test Accuracy* and *ROC-AUC*, we have chosen to focus on enhancing the **Logistic Regression** model. This decision was made to better illustrate the impact of several performance-improving techniques that are particularly effective and interpretable in the context of logistic models.

Logistic Regression is a linear model that is highly interpretable and responds well to thoughtful adjustments in data representation and optimization. Therefore, it provides a suitable foundation for applying and demonstrating advanced machine learning strategies, including:

- **Feature Engineering**: We will introduce new features derived from existing ones—such as polynomial feature expansion—to capture non-linear relationships that the base model might miss.
- **Regularization**: Both L1 (Lasso) and L2 (Ridge) penalties will be explored to prevent overfitting and to possibly improve model generalizability. These techniques are useful in simplifying the model while retaining predictive performance.
- **Cross-Validation**: To ensure robust model evaluation and to prevent results that depend on a single train-test split, we will use k-fold cross-validation. This helps assess how well the model generalizes to unseen data.

Through the application of these enhancement techniques, we aim to improve the performance of Logistic Regression and make it more competitive with Naive Bayes, while also demonstrating practical and interpretable ways to refine a baseline model.

6.1 Feature Engineering: Polynomial Features

To enhance the model's ability to capture non-linear relationships among features, we applied a polynomial feature transformation. Specifically, we used `PolynomialFeatures` from `scikit-learn` with a degree of 2. This transformation expanded the feature set by adding all possible interaction terms and squared terms of the original features, excluding the bias term.

By doing this, the Logistic Regression model was given access to a richer feature space that can better model complex relationships within the data, potentially leading to improved classification performance.

6.2 Regularization: L1 and L2 Penalties

To improve the generalization ability of the Logistic Regression model and reduce the risk of overfitting, we employed regularization techniques—specifically L2 (Ridge) and L1 (Lasso) regularization.

L2 Regularization (Ridge): This technique penalizes the sum of the squared coefficients and helps in shrinking the model weights, which reduces overfitting and enhances stability. We implemented it using the `liblinear` solver with a regularization strength parameter $C = 1.0$

L1 Regularization (Lasso): In addition to L2, we explored L1 regularization, which encourages sparsity in the model by driving less important feature coefficients to zero. This is particularly useful for feature selection

By incorporating both types of regularization, we aimed to not only prevent overfitting but also to assess the relevance of features through the sparsity induced by the L1 penalty.

6.3 Cross-Validation and Hyperparameter Tuning

To further optimize the Logistic Regression model, we applied **Stratified K-Fold Cross-Validation** with 5 splits. This ensures that each fold maintains the same class distribution as the original dataset, resulting in more reliable performance estimation.

We used `GridSearchCV` to perform an exhaustive search over a predefined hyperparameter grid. The grid included different values for the regularization strength C and both `L1` and `L2` penalties, using the `liblinear` solver

The best parameters were selected based on the highest cross-validation accuracy:

- **Best Parameters:** {C: 0.1, penalty: 'l2', solver: 'liblinear'}
- **Best CV Score:** 0.878
- **Test Accuracy:** 0.857

Model Evaluation and Visualization

The performance of the tuned model was evaluated on the test set using multiple metrics and visualizations:

- **Confusion Matrix:** Provided insight into true positives, true negatives, false positives, and false negatives.
- **ROC Curve and AUC:** Assessed the model's ability to distinguish between classes. The AUC score was $\sim$0.92, indicating high discriminative power.
- **Precision-Recall Curve and PR AUC:** Useful for evaluating performance in imbalanced classification. The PR AUC was $\sim$0.96.

These evaluations confirm that the enhanced Logistic Regression model achieved a strong balance between precision and recall, with robust test set performance after hyperparameter tuning.

7 Conclusion

In this project, we evaluated several machine learning classifiers to predict the target variable based on the available features. The pipeline began with Exploratory Data Analysis (EDA) and essential preprocessing steps, including data cleaning, feature transformation, and handling class imbalance.

Multiple models were tested, such as Logistic Regression, Naive Bayes, Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).

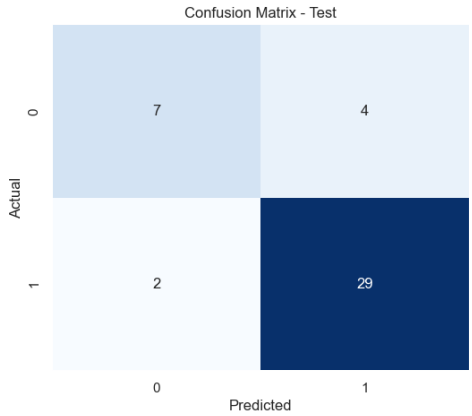

Figure 23: Confusion Matrix - Test Set

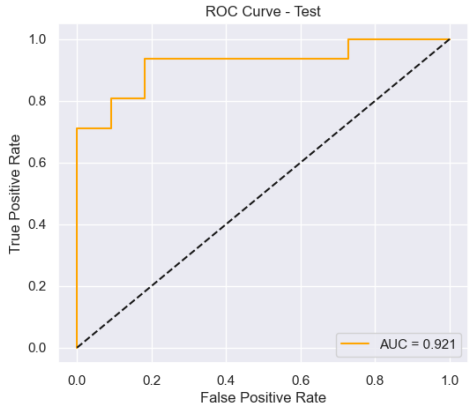

Figure 24: ROC Curve with AUC

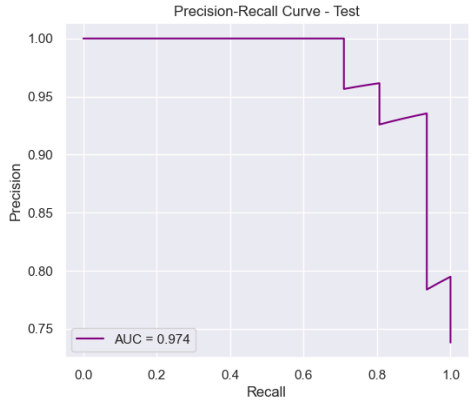

Figure 25: Precision-Recall Curve with AUC

Among these, the Naive Bayes classifier achieved the highest test accuracy and ROC-AUC score, making it a strong baseline model.

To illustrate the impact of systematic model enhancement, we focused on improving the Logistic Regression model. Through polynomial feature expansion, we introduced non-linear relationships; with regularization (L1 and L2), we controlled model complexity and promoted generalization. Finally, hyperparameter tuning via GridSearchCV with Stratified K-Fold cross-validation allowed us to select the best configuration based on validation accuracy.

These enhancements significantly improved the Logistic Regression model's performance and showcased the importance of proper feature engineering, regularization, and robust evaluation. While Naive Bayes remained the top-performing model, the improved Logistic Regression provided competitive results and greater interpretability.

Overall, this study emphasizes the value of iterative model improvement and highlights practical strategies that can be employed to boost performance in real-world machine learning tasks.

8 Future Work

Future improvements could include testing ensemble methods such as Random Forests and Gradient Boosting, which often yield strong predictive performance. Additionally, incorporating domain-specific features or external datasets related to student academic background, internships, or soft skills could further enhance model accuracy and generalization. Exploring deep learning approaches or automated feature selection techniques may also offer promising results for complex patterns in the data.

References

1. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

3. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

4. Kaggle. (n.d.). *Factors Affecting Campus Placement Dataset*. Retrieved from `https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement/data`