Report: Bank Marketing Classification

This report details a machine learning project using the **Bank Marketing UCI dataset** to predict customer subscription to a term deposit. It covers the methodology, implementation, and results of training two classification models: **Logistic Regression** and **K-Nearest Neighbors (KNN)**. The project addresses the dataset's class imbalance using the **SMOTE** technique and employs **GridSearchCV** for hyperparameter tuning to optimize model performance.

---

1. Dataset Description

The analysis uses the **Bank Marketing UCI dataset**, which contains data from direct marketing campaigns conducted by a Portuguese banking institution. The objective is to predict whether a client will subscribe to a term deposit, with the target variable being 'y' (yes or no). The dataset has **45,211 rows** and **17 features**, including demographic information, campaign-related details, and economic indicators.

Feature Name	Description	Data Type
age	Client's age	Numeric
job	Type of job	Categorical
marital	Marital status	Categorical
education	Education level	Categorical
default	Has credit in default?	Categorical
balance	Average yearly balance	Numeric
housing	Has housing loan?	Categorical
loan	Has personal loan?	Categorical
contact	Contact communication type	Categorical
day	Last contact day of the month	Numeric
month	Last contact month of the year	Categorical
duration	Last contact duration (in seconds)	Numeric
campaign	Number of contacts during this campaign	Numeric
pdays	Number of days passed after previous contact	Numeric
previous	Number of contacts before this campaign	Numeric
poutcome	Outcome of the previous marketing campaign	Categorical
y	**Target variable**: has the client subscribed to a term deposit? (yes/no)	Categorical

The target variable 'y' is **highly imbalanced**, with a vast majority of the samples belonging to the 'no' class. This imbalance can lead to models that perform well on accuracy but fail to correctly identify the minority class.

2. Steps and Methodology

The project followed a structured machine learning pipeline to ensure robust and reproducible results:

1. **Data Loading and Initial Exploration**: The dataset was loaded from a CSV file. The shape and initial rows were inspected, and the class distribution of the target variable was visualized.
2. **Preprocessing**: The data was split into **training (80%)** and **testing (20%)** sets. A `ColumnTransformer` was created to handle different feature types:
    - **Numeric features** were scaled using `StandardScaler` to normalize their ranges.
    - **Categorical features** were encoded using `OneHotEncoder` to convert them into a numerical format suitable for machine learning algorithms.
3. **Handling Class Imbalance**: The **Synthetic Minority Over-sampling Technique (SMOTE)** was applied to the training data. SMOTE generates synthetic samples of the minority class ('yes'), effectively balancing the dataset and improving the model's ability to learn the characteristics of both classes. This was integrated into an `imblearn` pipeline to prevent data leakage.
4. **Model Training and Tuning**: Two distinct `ImbPipelines` were created, each combining the preprocessing, SMOTE, and a classifier.
    - **Logistic Regression**: A `GridSearchCV` was used to find the best hyperparameters for `C` (regularization strength) and `solver`.
    - **K-Nearest Neighbors (KNN)**: A `GridSearchCV` was used to tune the number of neighbors (`n_neighbors`) and the `weights` parameter.
5. **Evaluation and Comparison**: After training, both models were evaluated on the held-out test set using several metrics:
    - **Accuracy**: The overall percentage of correct predictions.
    - **Classification Report**: Detailed metrics including precision, recall, and F1-score for each class.
    - **Confusion Matrix**: A visualization of the model's predictions versus the actual values.
    - **ROC Curve and AUC**: The Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) were plotted to compare the models' ability to distinguish between classes.

---

3. Results and Summary

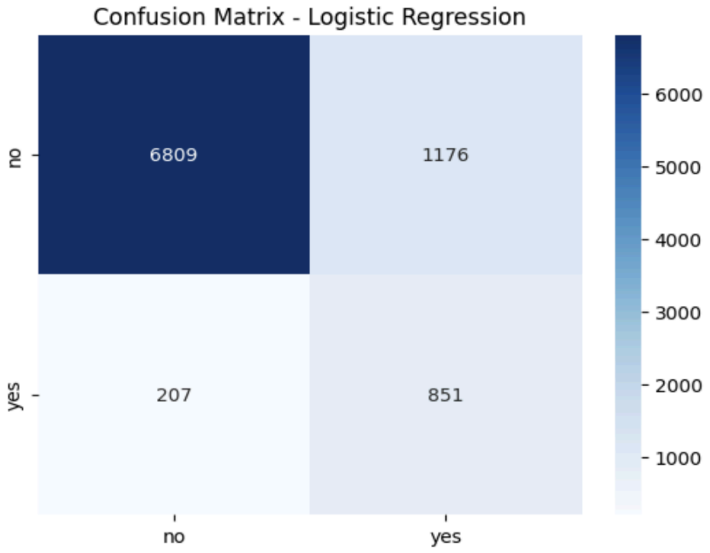
Logistic Regression Results

The `GridSearchCV` for Logistic Regression identified the best hyperparameters as `C=0.1` and `solver='liblinear'`. The model achieved an accuracy of approximately **84.7%** on the test set. The confusion matrix shows that while the model correctly predicts the majority class well, its performance on the minority class is also reasonable, thanks to SMOTE.

```
--- Logistic Regression Results ---
Best Params: {'clf__C': 10, 'clf__solver': 'lbfgs'}
Accuracy: 0.8470640274245272
              precision    recall  f1-score   support

          no       0.97      0.85      0.91      7985
         yes       0.42      0.80      0.55      1058

    accuracy                           0.85      9043
   macro avg       0.70      0.83      0.73      9043
weighted avg       0.91      0.85      0.87      9043
```



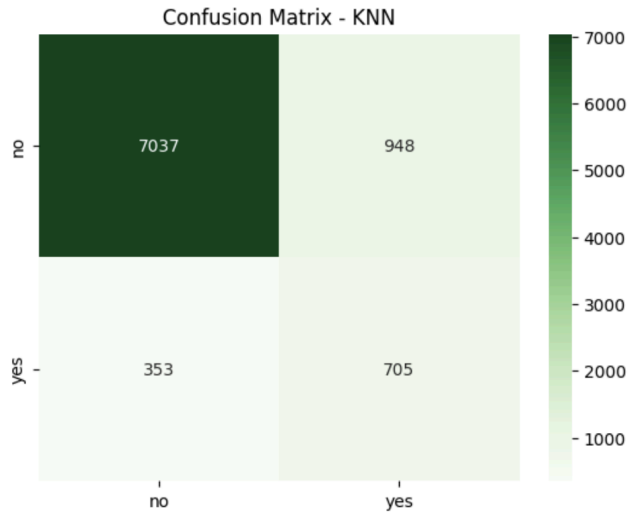
K-Nearest Neighbors (KNN) Results

For KNN, the best hyperparameters were found to be `n_neighbors=9` and `weights='distance'`. This model achieved an accuracy of approximately **85.6%**, slightly outperforming the Logistic Regression model. The confusion matrix for KNN also shows a strong performance, particularly in correctly classifying the majority class.

```
--- KNN Results ---
Best Params: {'clf__n_neighbors': 3, 'clf__weights': 'distance'}
Accuracy: 0.8561318146632755
              precision    recall  f1-score   support

          no       0.95      0.88      0.92      7985
         yes       0.43      0.67      0.52      1058

    accuracy                           0.86      9043
   macro avg       0.69      0.77      0.72      9043
weighted avg       0.89      0.86      0.87      9043
```

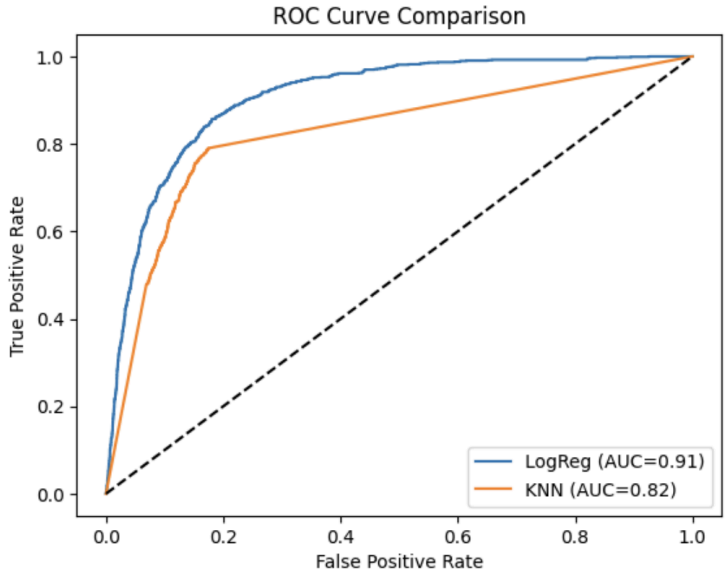


Comparison with Plots

The **ROC Curve** plot clearly illustrates the trade-off between the True Positive Rate and the False Positive Rate for both models. The **Area Under the Curve (AUC)** values provide a single metric to compare their overall performance.

- **Logistic Regression AUC**: 0.91
- **KNN AUC**: 0.82

The ROC curve shows that Logistic Regression has a slightly higher AUC, suggesting it is better at ranking positive instances than KNN, despite KNN having a higher overall accuracy.



Final Summary

Both Logistic Regression and K-Nearest Neighbors, when combined with SMOTE, proved to be effective models for classifying customer subscriptions. The KNN model achieved a higher overall accuracy of **85.6%** compared to the Logistic Regression model's **84.7%**. However, the Logistic Regression model had a slightly better AUC score, indicating its superior ability to rank predictions and distinguish between the classes. For this specific problem, the **KNN model appears to be the better choice** due to its higher accuracy, but the choice between the two may depend on which metric (accuracy vs. AUC) is prioritized for the business case.