24CSAI03H
Machine Learning

Project 1 Report

Binary Classification of Bank Marketing

Sameh218767

Overview

This data set contains records relevant to a direct marketing campaign of a Portuguese banking institution. The marketing campaign was executed through phone calls. Often, more than one call needs to be made to a single client before they either decline or agree to a term deposit subscription. The classification goal is to predict if the client will subscribe (yes/no) to the term deposit (variable y). The data set is chronological, covering campaigns carried out from May 2008 to November 2010. The dataset provides a rich test bed for machine learning algorithms with all sorts of insights into customer behavior and effectiveness of campaigns. This would empower the institutions to run more effective campaigns by predicting the probability of a customer signing up for a term deposit, thereby reducing costs and improving
customer targeting and experience through ethical marketing practices.

Source: https://www.kaggle.com/datasets/ruthgn/bank-marketing-data-set/data

Dataset Attributes:

1. **age**: Age of the client (numeric).
2. **job**: Type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown').
3. **marital**: Marital status (categorical: 'divorced', 'married', 'single', 'unknown').
4. **education**: Level of education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown').
5. **default**: Has credit in default? (categorical: 'no', 'yes', 'unknown').
6. **housing**: Has housing loan? (categorical: 'no', 'yes', 'unknown').
7. **loan**: Has personal loan? (categorical: 'no', 'yes', 'unknown').
8. **contact**: Contact communication type (categorical: 'cellular', 'telephone').
9. **month**: Last contact month of the year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec').
10. **day_of_week**: Last contact day of the week (categorical: 'mon', 'tue', 'wed', 'thu', 'fri').
11. **campaign**: Number of contacts performed during this campaign for this client (numeric, includes last contact).
12. **pdays**: Number of days that passed since the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted).

13. **previous**: Number of contacts performed before this campaign for this client (numeric).
14. **poutcome**: Outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success').
15. **emp.var.rate**: Employment variation rate - quarterly indicator (numeric).
16. **cons.price.idx**: Consumer price index - monthly indicator (numeric).
17. **cons.conf.idx**: Consumer confidence index - monthly indicator (numeric).
18. **euribor3m**: Euribor 3-month rate - daily indicator (numeric).
19. **nr.employed**: Number of employees - quarterly indicator (numeric).

Output Variable (Desired Target):

y: Has the client subscribed to a term deposit? (binary: 'yes', 'no').

Data Exploration

The dataset is first explored by checking the basic structure and statistics such as the number of rows and columns, datatypes, number of unique values per class, number of nulls and class distribution of the target. Initial analysis shows that the data includes some missing values, especially in the minority classes (class that represents 'yes'), and most importantly there is a major class imbalance in the distribution of the target variable, with class 0 ('no') being the majority and class 1 ('yes') as shown below:

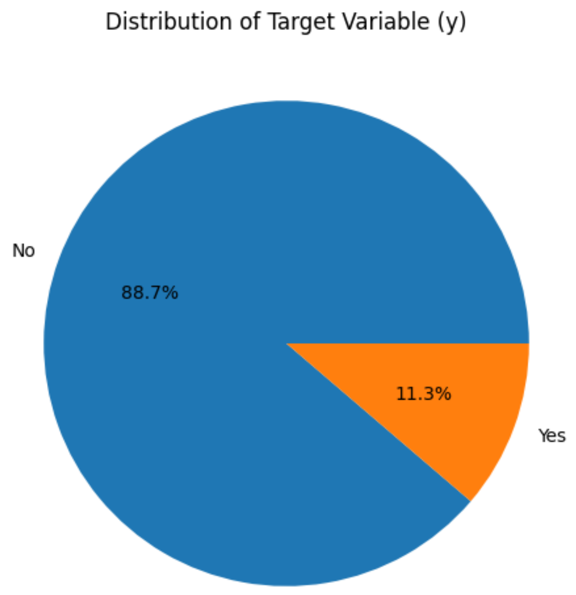
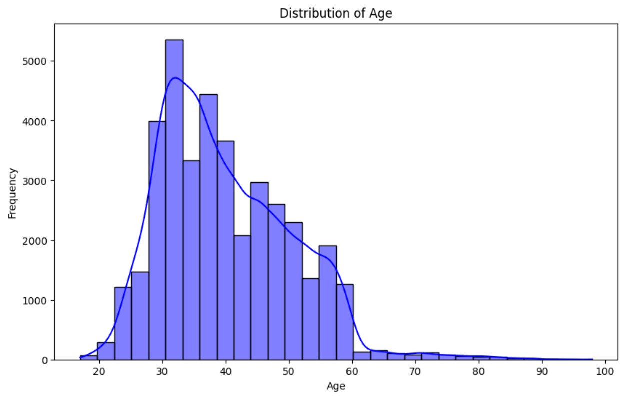


Figure 1: Target class distribution

Data Visualization

Several general visualizations were done in this section to further explore and create initial hypotheses how some features might correlate with others and the target classes. Among the most notable is a heatmap of the missing values in the dataset, a plot distribution of age to find a pattern of the people contacted, a heatmap to show correlation between features, and finally, a pie chart of the class distribution of the target variable with emphasis on the class imbalance shown earlier.



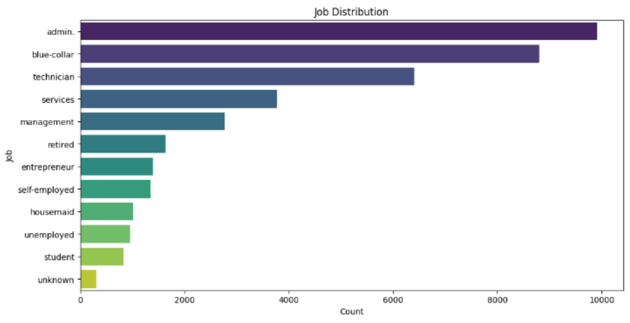


Figure 2: Age Distribution (Positive Skew)

Figure 3: Job Distribution Across Target Classes

Data Preprocessing

The preprocessing stages were done and modified in accordance with earlier findings as well as future operations that required redoing of certain processes or changing their order. A rigorous methodology was thus followed starting with renaming the columns into something more descriptive of what they represent for easier readability later on. This is closely followed by variable encoding for categorical features where 10 features with low cardinality were encoded into numeric representation. Then came null handling where the dataset was divided into two based on target classes, and each class was divided based on whether or not the features are numerical or categorical. As a result, 4 lists were conducted of the appropriate methodologies such as KNN, K-means, median, mode and K-mode imputations according to the datatype. It was then looped over all possible combinations until one yielded the highest results. We dropped nulls and converted features into their appropriate datatypes as to not mess with future stages. Lastly, Age, has its positive skewness fixed and then discretized into meaningful categories for better representation as shown below:

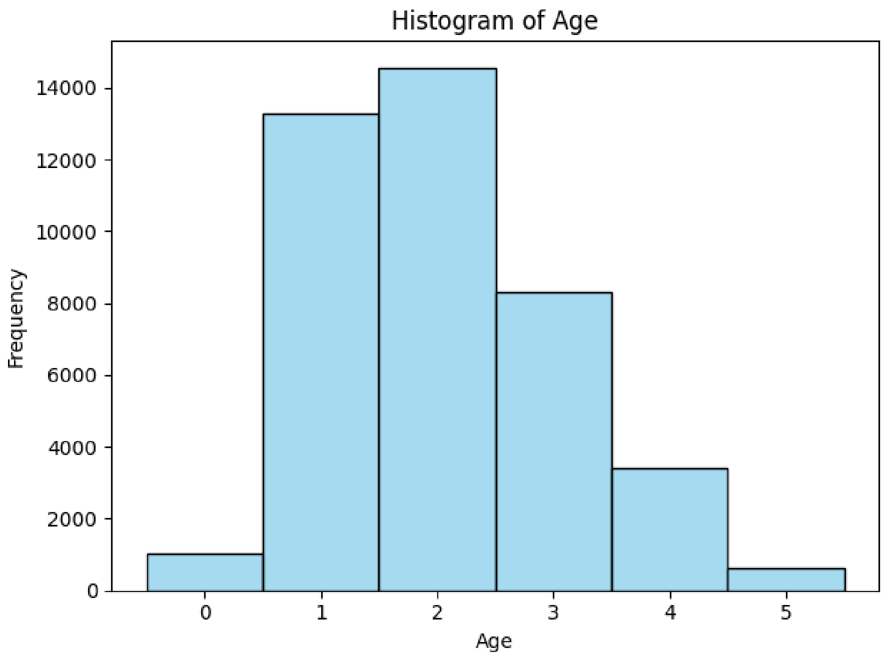


Figure 4: Age Distribution After Discretization

Feature Selection

In this section we prepare the final dataset that will be used in out models by carefully selecting the appropriate features that best predict the outcome of a model without overfitting it.
To select our features, we can try one of the following strategies and their combined results and compare:

1. **Filter method:** Correlation-based Feature Selection (for numerical data)
2. **Filter method:** Chi-Squared Test (for categorical data)
3. **Embedded method:** Random Forest

For the filter-based methods, we split the feature selection to specifically deal with numerical data for Correlation-Based Feature Selection, while mainly utilizing Chi-Squared Test for categorical data. First, we specifically focused on columns with high correlations between certain features might indicate redundancy. To visualize this, we computed and plotted the correlation matrix. Columns with the highest correlations with several other columns and manually dropping them accordingly.

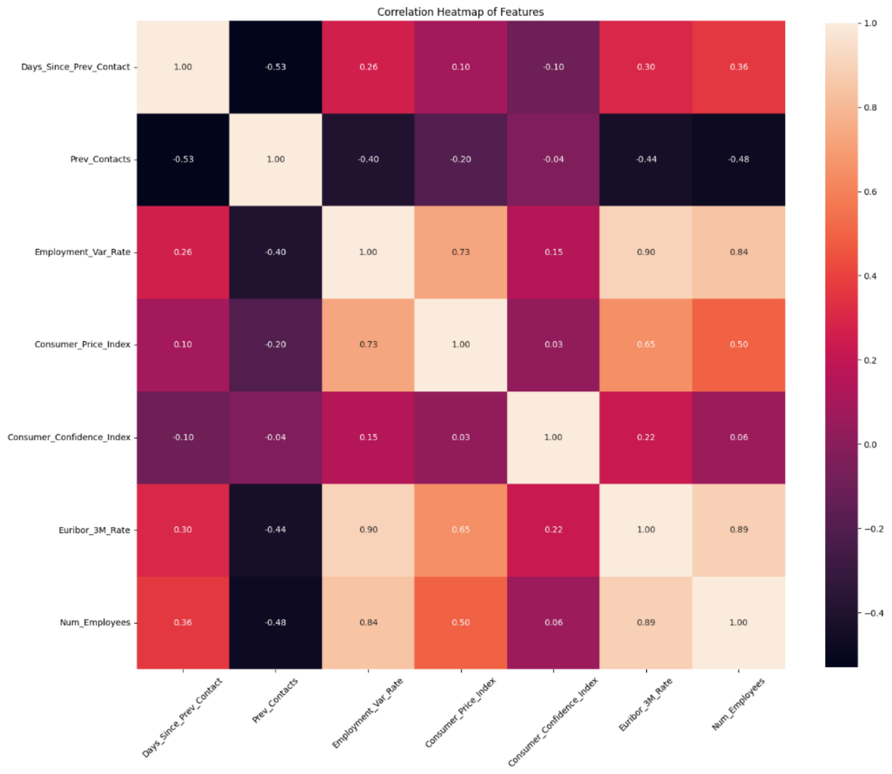


Figure 5: Correlation Matrix Across Numerical Features

A Chi-Squared test was performed in order to find out how each feature correlated with the target variable. It checked if the distribution of values for each feature differs significantly from what would be expected, assuming that the feature is independent of the target. The Chi-Squared scores and p-values for each feature were calculated, applying the Chi-Squared test. The features were then ranked according to the Chi-Squared score, whereby the features with the highest scores were considered most relevant for prediction. The top 20 features were taken for further analysis.

For the embedded method, Random Forest was used in order to estimate the feature importance while training the models. One of the strengths of using Random Forest is the ready-made way of ranking features based on their usefulness in prediction of the target variable. By training a Random Forest model and inspecting the feature importances, one could find out which features contributed most to the model's predictions. The top 17 features were selected based on their importance scores.

Finally, we combine the features selected by Correlation-based Feature Selection, Chi-Squared test and Random Forest to identify the most commonly selected features. The intersection of these sets gives us the final set of features for model training.

Resampling

Now for the most crucial step for this dataset - resampling. Three strategies were taken and compared against one another to find the best methodology.

1. Undersample majority class down to the same size as the minority. It's a straightforward approach to ensure a 1:1 ratio.
2. Undersample majority and Oversample minority. Since we only have 2 classes, we can under sample the majority class and oversample the minority to the halfway point of the dataset size, which was 20,000, to ensure equal representation during the training phases without going one extreme or the other. Afterwards, SMOTENC was employed for its ability to work with categorical data as well as numerical ones to oversample the minority class as well as the middle class to also reach the same 20,000 size.
3. Utilize class weights. Since some models such as tree-based like Random Forest have this parameter, it was decided to simply put the dataset as it into the model with 1:9 ratio for class 0:class 1.

The results can be summarized in the following table:

Metric	Undersampling Only	Undersampling + SMOTENC	Class Weights
Accuracy	0.7946	0.8214	0.9119
Precision	0.7964	0.8198	0.9024
Recall	0.7946	0.8214	0.9119
F1-score	0.7943	0.8204	0.9035
Support (Class 0)	1383	2766	10142
Support (Class 1)	1383	1383	1383
Precision (Class 0)	0.77	0.86	0.93
Recall (Class 0)	0.83	0.88	0.97
F1-score (Class 0)	0.80	0.87	0.95
Precision (Class 1)	0.82	0.74	0.71
Recall (Class 1)	0.76	0.71	0.46
F1-score (Class 1)	0.79	0.73	0.55
Accuracy (overall)	0.79	0.82	0.91
Macro Avg (Precision)	0.80	0.80	0.82
Macro Avg (Recall)	0.79	0.79	0.71
Macro Avg (F1-score)	0.79	0.80	0.75
Weighted Avg (Precision)	0.80	0.82	0.90
Weighted Avg (Recall)	0.79	0.82	0.91
Weighted Avg (F1-score)	0.79	0.82	0.90
Total Samples	2766	4149	11525

The metrics shown above offer mixed results. On the one hand there is undersampling only, which offers a decent balance between both classes, however, causes the majority class to lose a lot of information and thus result in a lower f1 score. Then there is class weights, which had really high f1 score for the majority class, but doesn't seem to have influenced the minority class as it has a measle 0.55 accuracy. Lastly, there is the mixed approach. On one hand it retains some of the high f1 score for the majority class, on the other hand it has a higher f1 score than the class weights approach, but not higher than the undersampling only approach. From this, one could conclude that, stability-wise, the mixed approach offers the best of both worlds, especially considering that this data will be used against a variety of other models.

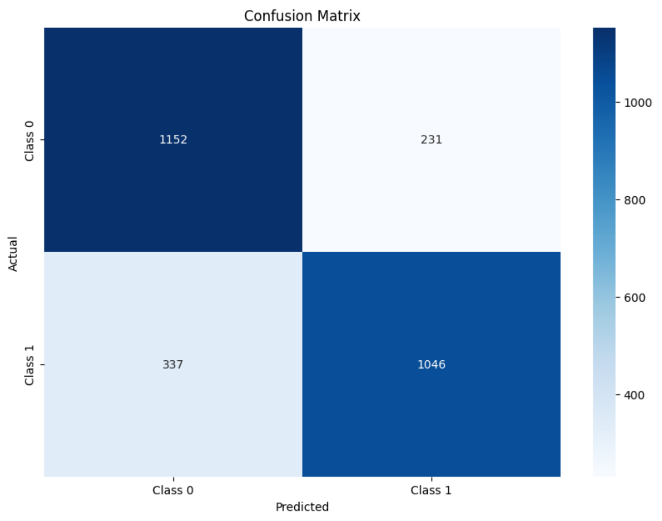


Figure 6: Confusion Matrix of Undersampling only

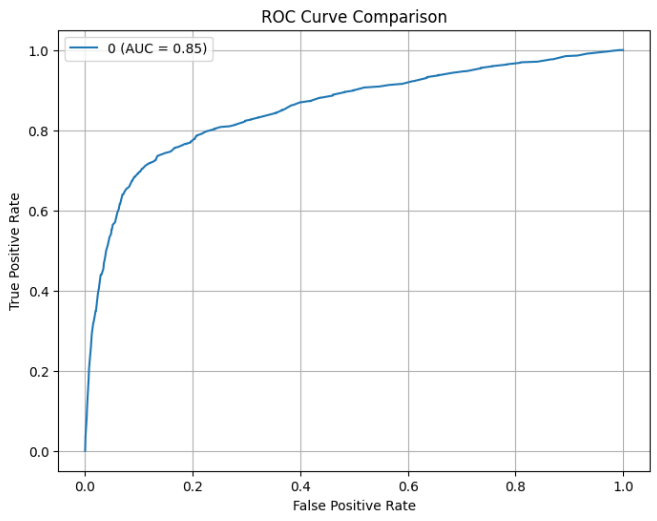


Figure 7: Learning curve of Undersampling only

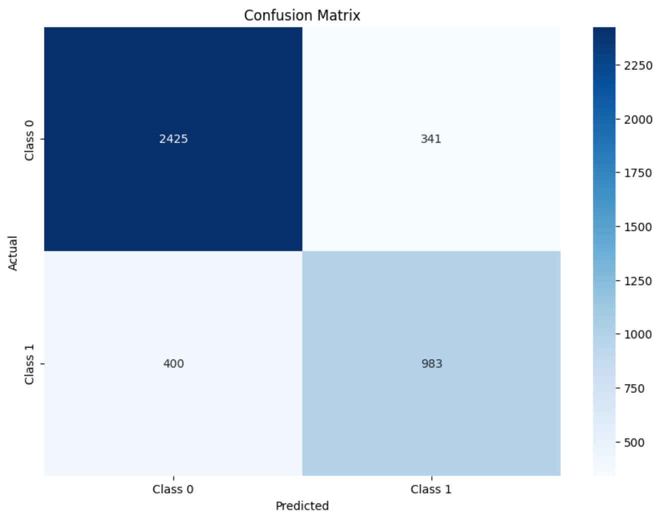


Figure 8: Confusion Matrix of Undersampling + SMOTENC

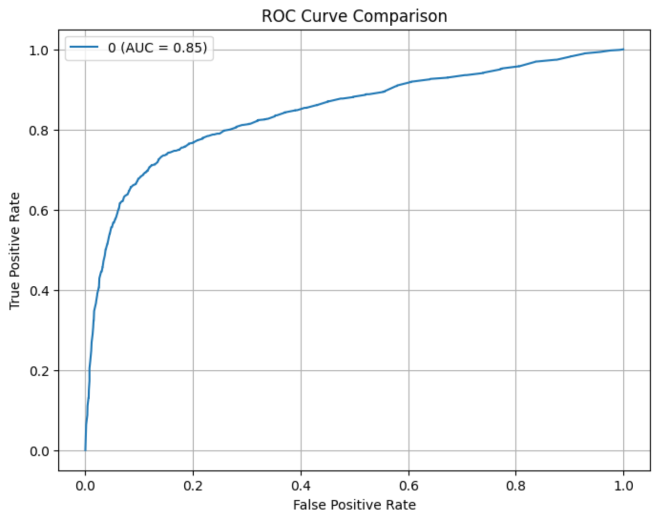


Figure 9: Learning curve of Undersampling + SMOTENC

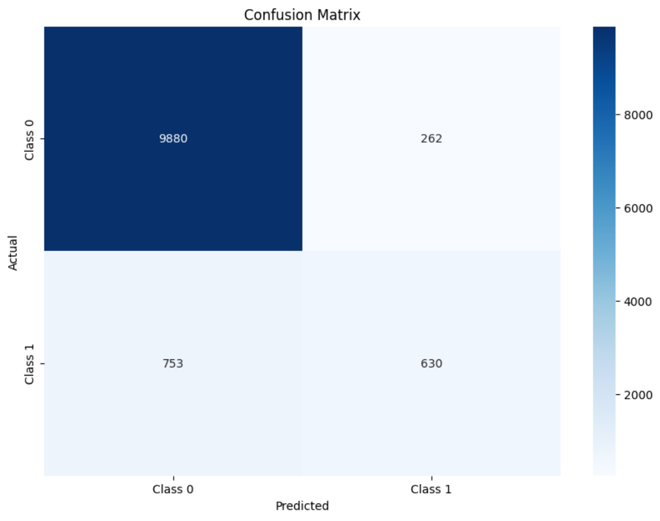


Figure 10: Confusion Matrix of Class Weights

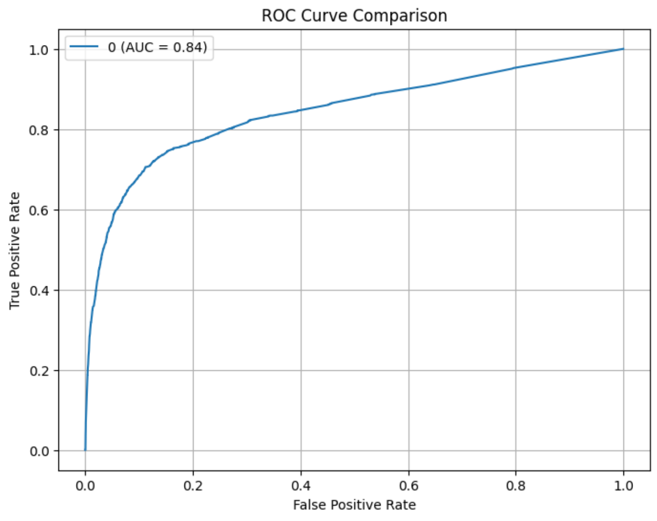


Figure 11: Learning curve of Class Weights

Model Training and Evaluation

As we finally arrive at the meat and bone of our work, we need to address a couple of things to justify what comes later. Since the domain of this dataset is a financial one, the goal is to make the most of this data and provide the highest possible accuracy. And because accuracy as a metric tends to be misleading, the goal of this training is to **increase the f1 score** for classes 0 and 1, and more specifically try to make both values reach comparable levels.

The following sequence of methodologies was used to achieve this goal:

- Implement 3 individual models (KNN, Decision Tree, Logistic Regression)
- Implement 3 ensemble models (Random Forest, LightGBM, XGBoost)
- Hyperparameter Tuning (Perform random search for hyperparameter optimization)
- Display Learning Curve for each to illustrate bias/variance
- Generate 3 versions of each model to show performance change
    - Without feature selection and hyperparameters
    - Without feature selection, but with hyperparameters
    - With feature selection and hyperparameters
- Evaluate performance: accuracy, precision, recall, F1-score, and ROC-AUC
- Confusion Matrix

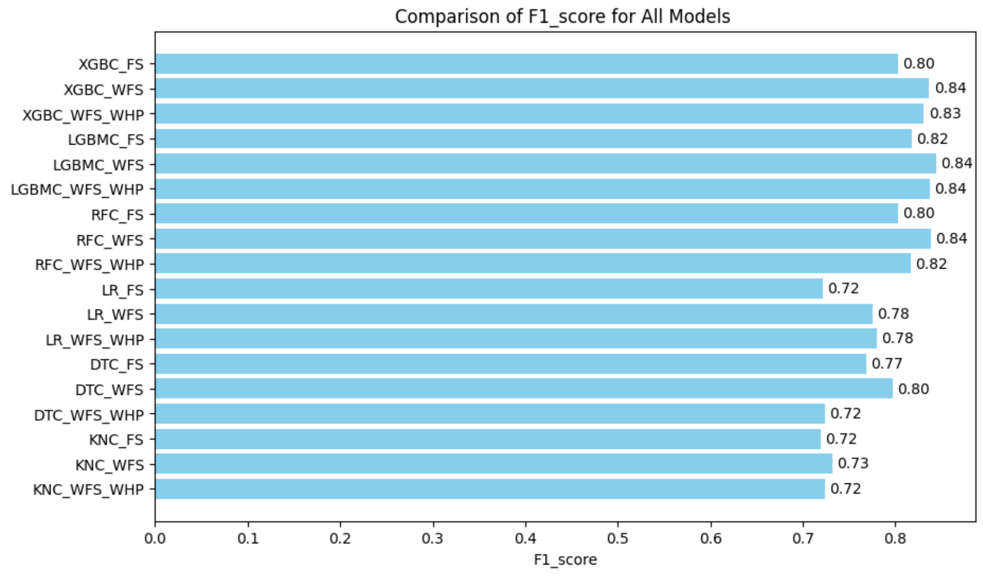
Each model was trained and evaluated in three distinct configurations:

1. **Baseline Model:** Trained without feature selection or hyperparameters to provide a performance benchmark.
2. **With Hyperparameter Tuning:** Trained without feature selection but optimized to improve model performance.
3. **With Feature Selection and Hyperparameters:** Uses selected features and tuned parameters to maximize predictive accuracy and efficiency.

Each setup shows how feature selection and tuning impact performance. This approach reveals the importance of identifying the best model setup that maximizes f1 score as much as possible for either class.

Comparison of Models

After cleaning, resampling, and training various models, we can now compare their performance across different metrics. Each model is evaluated based on accuracy, precision, recall, F1-score, and ROC-AUC, which provides a comprehensive view of their predictive capabilities. Starting with the most notable graph that compares accuracy for all models of different configurations:



Legend for Model Names:

RFC: Random Forest

DTC: Decision Tree

LGBMC: LightGBM

LR: Logistic Regression

XGBC: XGBoost

KNC: KNN

WHP: Without Feature Selection and Hyperparameters

WFS: Without Feature Selection, With Hyperparameters

_FS: With Feature Selection and

Figure 12: F1 Comparison Across All Models

This trend can also be clearly seen in the precision, recall and f1-scores accordingly. Indicating that if a model performs well in one metric, it well performs just as good in another. Perhaps the greatest save by grace was the Decision Tree model. As seen above, it seems as though it is the only model that benefitted the most from hyperparameter tuning and went on to achieve comparable, if not greater results, to the boosting algorithms. This, along with other trends in this graph, can also be further illustrated in the following ROC curves:

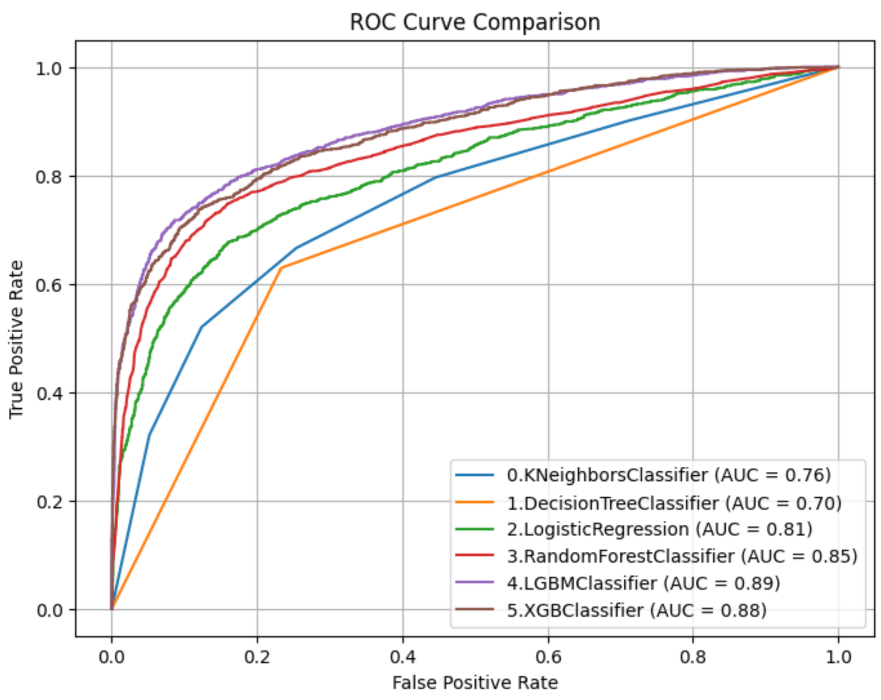


Figure 9: ROC Curve Comparison Between Baseline Models

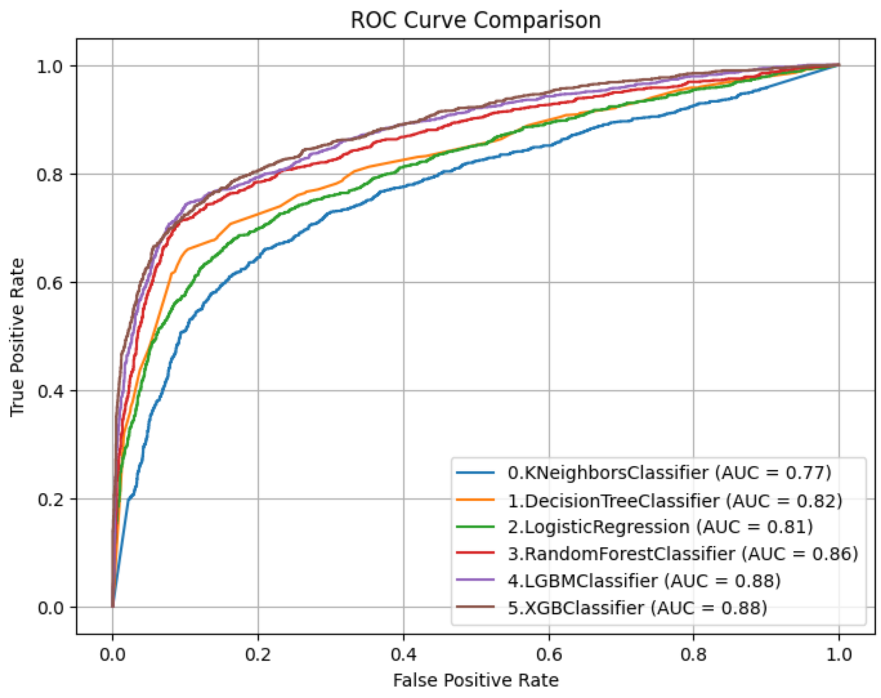
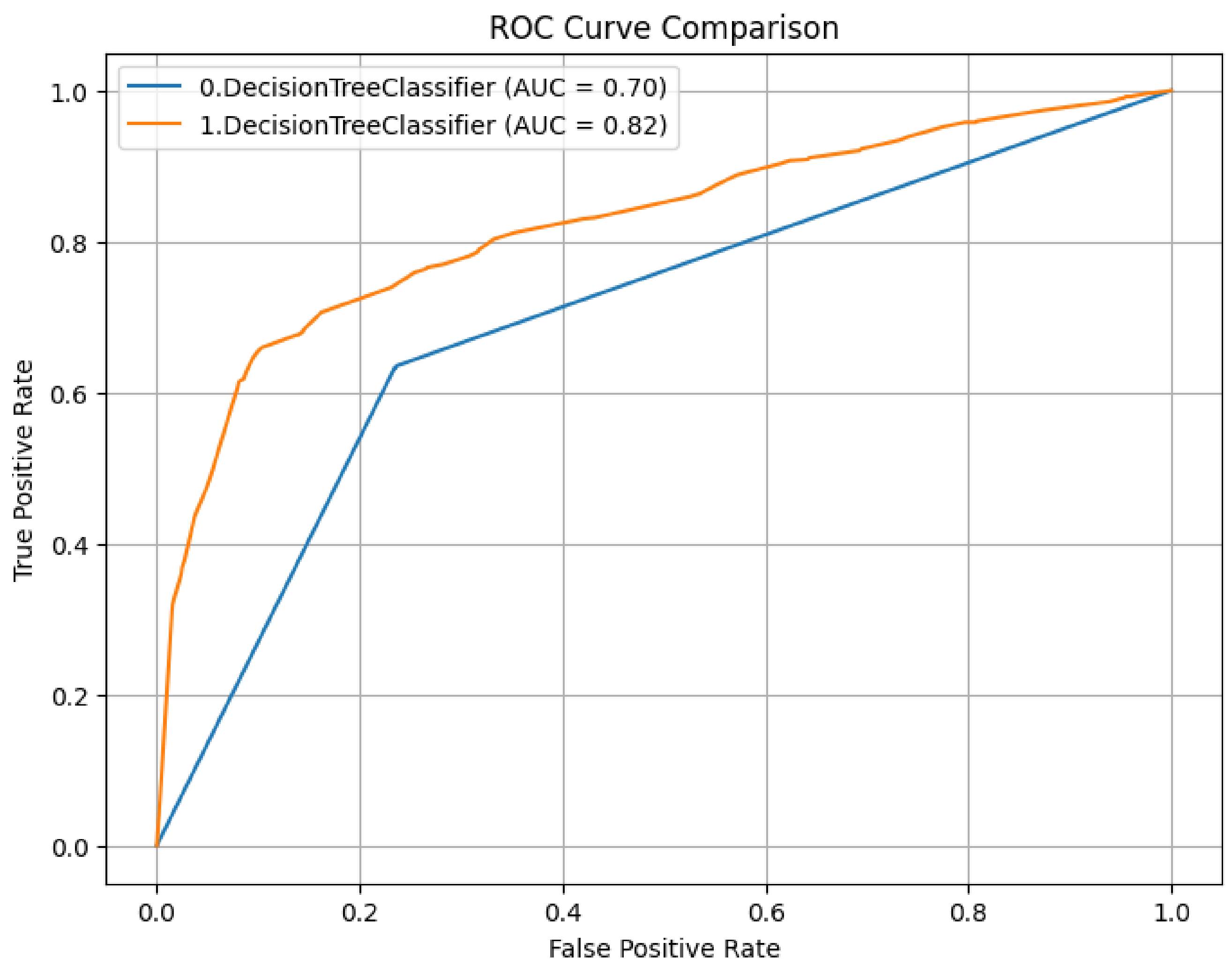


Figure 10: ROC Curve Comparison Between Models
(With hyperparameter tuning)

Conclusion

Finally, after several trials and assessments, the boosting algorithms **(XGBoost and LightGBM)** turned out to be the top-performing models in classifying whether or not the client will subscribe to the term deposit or no. Both models had a near identical performance in every metric while outperforming all the other models. The inclusion of feature selection does not enhance either of the models' individual performances, and even comes as a detriment across all metrics. Similarly, hyperparameter-tuning adds little value, and performing grid-search to find the best parameters only yields results comparable to the base models performance at best. Thus, we conclude that **(XGBoost and LightGBM)** without feature selection and hyperparameter-tuning are the most suitable model for predicting whether a client will subscribe or not.


ROC Curve Comparison
0.DecisionTreeClassifier (AUC = 0.70)
1.DecisionTreeClassifier (AUC = 0.82)
True Positive Rate
False Positive Rate