**Optimizing Ensemble Models with ROC-Based Thresholds**

Jean P. Melendez Villanueva

University of Maryland

Data 660 Advanced Topics in Data Science

Professor Ed Herranz

**Optimizing Ensemble Models with ROC-Based Thresholds**

1. **Introduction**

In this assignment, I applied ensemble machine learning methods to predict customer subscription to a term deposit based on the Bank Marketing dataset. This dataset, which was also used in Unit 3, involves a binary classification task with a known class imbalance, where the majority of customers do not subscribe. The purpose of the project was to build accurate, interpretable, and well-balanced models that could distinguish between likely subscribers and non-subscribers using structured customer data.

To address this classification task, I developed and evaluated three ensemble models: Random Forest, Gradient Boosting, and a Stacking Ensemble. Rather than using SMOTE or other oversampling techniques to handle class imbalance, I implemented a cutoff-based approach using the ROC curve. This method allowed me to adjust the prediction threshold to optimize model sensitivity and specificity while maintaining the original data distribution, a common practice in real-world deployments where oversampling may not be feasible.

The overall goal was to create models that performed well not only in terms of accuracy but also in identifying as many true positive “yes” predictions as possible. Adjusting the classification threshold provided a way to fine-tune the models’ ability to detect rare but important outcomes, which is particularly valuable in marketing, healthcare, fraud detection, and other fields where false negatives can be costly. This paper outlines the steps taken to prepare the data, train and evaluate ensemble models, and apply threshold optimization to improve performance.

1. **Data Quality Assessment**

The Bank Marketing dataset was first loaded into Python for a preliminary assessment of its structure and content. The dataset contained a mix of categorical and numerical features describing customer demographics, banking information, and contact history. Initial inspection revealed no missing values or duplicate entries, which meant no rows needed to be removed for data integrity purposes. Data types for each column were verified and found to be consistent with their expected formats, allowing for straightforward encoding in later steps.

A key focus of the quality assessment was identifying the presence of class imbalance in the target variable, which represents whether or not a customer subscribed to a term deposit. The majority of entries were labeled “no,” indicating a customer did not subscribe, while a significantly smaller portion were labeled “yes.” This imbalance raised concerns about the model’s ability to accurately detect the minority class and motivated the decision to use a cutoff-based approach instead of resampling methods like SMOTE. Understanding the imbalance at this stage was critical to shaping the modeling strategy.

In addition to class balance, I examined the distribution and range of feature values to check for outliers or anomalies. Visualizations and descriptive statistics helped confirm that the data was generally clean and did not require transformation or removal of extreme values. The dataset was therefore considered ready for encoding and modeling without the need for major cleansing steps. This streamlined the workflow and ensured that the modeling process would reflect the original characteristics of the data.

1. **Exploratory Data Analysis (EDA)**

Exploratory data analysis was conducted to gain a deeper understanding of the dataset’s structure and to identify patterns that could influence model performance. Visualizations such as histograms and box plots were created for key numerical features, including age, duration of last contact, and number of contacts. These plots revealed slight skewness in variables like duration, which is expected given that longer calls tend to occur in successful interactions. Most features appeared to have reasonable ranges and distributions, and no significant outliers required removal.

To explore potential relationships between features, I generated a correlation matrix for the numerical variables. Although the dataset contained many categorical variables, the correlation matrix helped highlight mild associations, such as a positive relationship between contact duration and subscription likelihood. Additionally, I examined cross-tabulations between categorical features and the target variable. For instance, job type, education level, and month of last contact showed varying levels of influence on customer decisions, suggesting they would be important predictors for the classification task.

Finally, I analyzed the distribution of the target variable, which confirmed the imbalance noted during the quality assessment. The majority of customers (approximately 88%) did not subscribe to a term deposit, while only a small fraction (around 12%) did. This imbalance emphasized the importance of using an evaluation strategy that considered both sensitivity and specificity. The EDA process provided insight into the data’s structure and confirmed that feature selection and transformation would proceed without major modifications.

1. **Data Splitting and Class Imbalance Strategy**

After completing the exploratory analysis, I partitioned the dataset into training and testing sets using an 80/20 split. The training set was used to develop and tune the ensemble models, while the test set was reserved for final evaluation. Stratification was applied during the split to preserve the original class distribution across both subsets. This ensured that the class imbalance—where most customers did not subscribe—remained present in both sets, reflecting a realistic deployment environment.

To address the imbalance, I implemented a cutoff-based approach rather than applying oversampling techniques like SMOTE. While SMOTE generates synthetic examples of the minority class to balance the training set, the cutoff method preserves the original data distribution and avoids introducing artificial data. This was done by using each model’s predicted probabilities to generate a Receiver Operating Characteristic (ROC) curve, from which an optimal decision threshold was selected. The threshold was chosen based on the point that maximized balanced accuracy, which is the average of sensitivity and specificity.

This approach allowed each model to achieve improved detection of the minority class (true “yes” cases) without compromising its ability to correctly identify the majority class. By adjusting the decision threshold downward (e.g., from 0.5 to 0.16 for Random Forest), the models were able to capture more true positives, improving sensitivity while maintaining strong specificity. This method is widely used in real-world applications where class imbalance is common, such as fraud detection, medical diagnosis, and marketing campaigns.

1. **Model Implementation and Evaluation**

Three ensemble models were implemented to perform the classification task: Random Forest, Gradient Boosting, and a Stacking Ensemble. Each model was trained using the original, imbalanced training data, and evaluated using cross-validation to generate probability scores. These probabilities were then used to calculate ROC curves and determine the optimal threshold for classification. Instead of relying on the default 0.5 threshold, each model’s threshold was adjusted to maximize balanced accuracy, thereby improving the detection of “yes” outcomes.

The Random Forest model was configured with 100 estimators and performed well after threshold tuning, increasing its sensitivity from 0.28 to 0.83. Gradient Boosting, also with 100 estimators, demonstrated similar performance, achieving high sensitivity and balanced accuracy after threshold adjustment. The Stacking model combined the predictions of Random Forest, Gradient Boosting, and a Decision Tree using Logistic Regression as the final estimator. While it produced slightly lower sensitivity at the default threshold, its performance matched the other models once the cutoff was optimized.

Each model’s performance was measured using four key metrics: sensitivity, specificity, accuracy, and balanced accuracy. These metrics provided a comprehensive view of how well the models distinguished between positive and negative outcomes, especially in the presence of class imbalance. The results confirmed that adjusting the threshold had a significant impact on sensitivity and balanced accuracy, highlighting the value of the cutoff-based strategy. All three models showed competitive performance after tuning, supporting their use in scenarios requiring high recall for minority class detection.

1. **Results and Comparison**

The results demonstrated that all three models improved substantially when evaluated using optimized thresholds instead of the default 0.5 cutoff. With the default threshold, each model exhibited high specificity but poor sensitivity, meaning they correctly identified most “no” cases but failed to capture many “yes” cases. After applying the ROC-based cutoff strategy, each model achieved a stronger balance between sensitivity and specificity, resulting in significantly higher balanced accuracy.

The Random Forest model achieved the highest balanced accuracy of 0.836 when its threshold was adjusted to 0.16. Gradient Boosting and Stacking models both reached balanced accuracy scores of 0.833 after adjusting their thresholds to 0.11 and 0.08, respectively. While Random Forest slightly outperformed the others in balanced accuracy, all three models were highly effective once threshold tuning was applied. The performance improvement was most evident in sensitivity, which more than doubled in each case—indicating the models became much better at identifying true “yes” outcomes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Threshold | Sensitivity | Specificity | Accuracy | Balanced Accuracy |
| Figure 1. Random Forest | 0.50 | 0.276 | 0.982 | 0.900 | 0.629 |
| Figure 1. Random Forest | 0.16(opt) | 0.827 | 0.845 | 0.843 | 0.836 |
| Figure 2. Gradient Boosting | 0.50 | 0.401 | 0.969 | 0.904 | 0.685 |
| Figure 2. Gradient Boosting | 0.11(opt) | 0.835 | 0.832 | 0.832 | 0.833 |
| Figure 3. Stacking | 0.50 | 0.381 | 0.973 | 0.905 | 0.677 |
| Figure 3. Stacking | 0.08(opt) | 0.827 | 0.838 | 0.837 | 0.833 |

The table above summarizes the performance of all three models at both default and optimized thresholds. Across all models, accuracy remained consistently high, but balanced accuracy and sensitivity were significantly improved through threshold tuning. These findings support the use of the cutoff-based approach in imbalanced classification problems where identifying positive cases is a priority, such as in marketing outreach and customer targeting strategies.

1. **Conclusion**

This assignment demonstrated the effectiveness of ensemble models and threshold tuning in addressing class imbalance within a real-world classification problem. By applying a cutoff-based approach instead of data resampling, I was able to preserve the dataset’s integrity while significantly improving model sensitivity and balanced accuracy. Each model—Random Forest, Gradient Boosting, and Stacking—benefited from threshold optimization, allowing for more accurate identification of positive cases without compromising overall model performance.

The process reinforced the importance of evaluating models beyond accuracy, especially in cases where the minority class holds greater value. ROC curve analysis and threshold adjustment provided a practical and interpretable method to enhance model effectiveness for decision-making in business and analytics contexts. These techniques are directly applicable in domains such as marketing, finance, and healthcare, where predicting rare but meaningful outcomes can drive critical actions and improve outcomes.

**References**

Brownlee, J. (2020). *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, and Improve Model Performance*. Machine Learning Mastery.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Provost, F., & Fawcett, T. (2013). *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media.

**Appendix**

**Appendix A –** Refer to the accompanying Jupyter Notebook file for code implementation

**Appendix B –** Screenshots

A screen shot of a graph

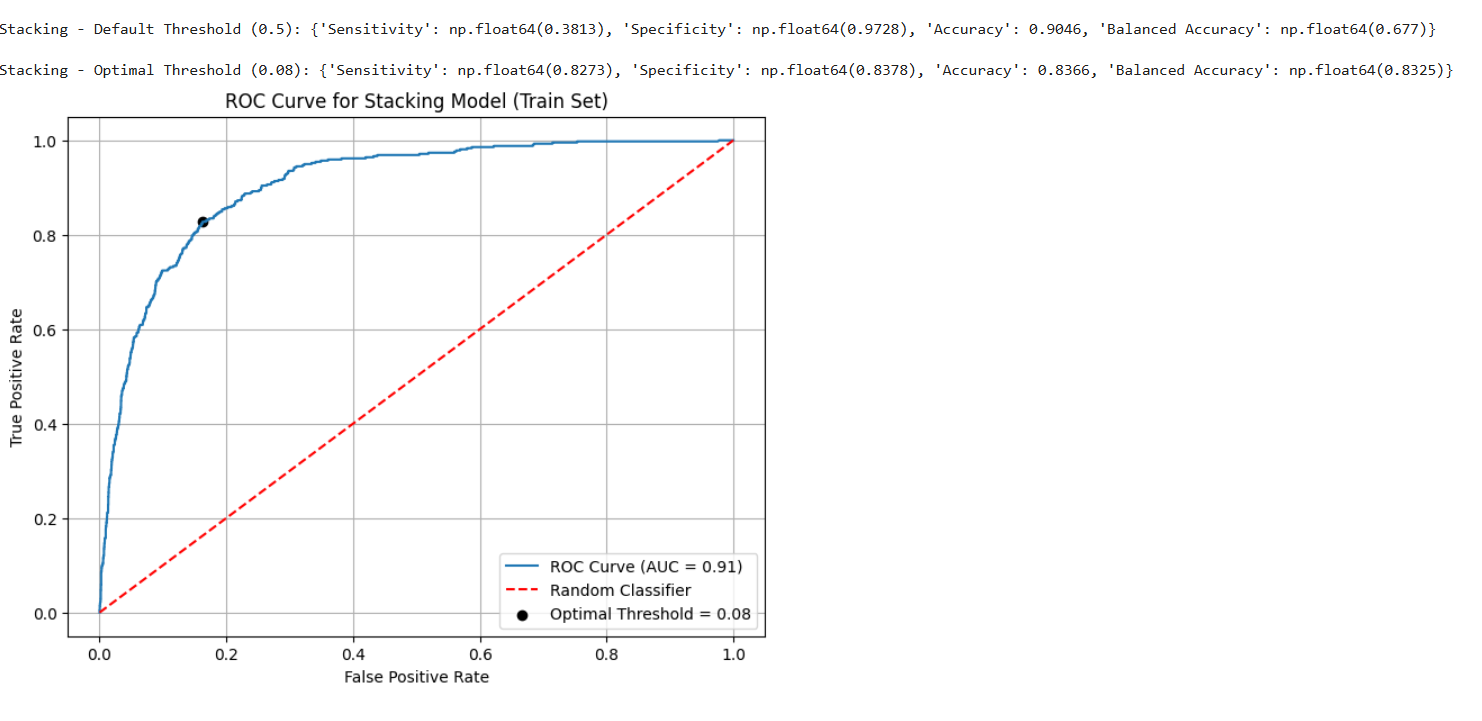
AI-generated content may be incorrect.

(Figure 1) ROC Curve for Random Forest

A screen shot of a graph

AI-generated content may be incorrect.

(Figure 2) ROC Curve for Gradient Boosting



(Figure 3) ROC Curve for Stacking Model