Monsoon Credittech Assessment

Name: Vishal Vasoya
Mail: vasoyavishal33@gmail.com
Mobile: 7285052158

Solution Approach:

1. Load the Data and Get Basic Information
2. Data Preprocessing and Feature Engineering
3. Exploring Data Distribution and Outlier
4. Addressing Class Imbalance
5. Data Standardization and Model Selection
6. Model Selection, Evaluation and Comparison
7. Hyperparameter Tuning for the Best Model
8. Make Predictions and Prepare Final Submission

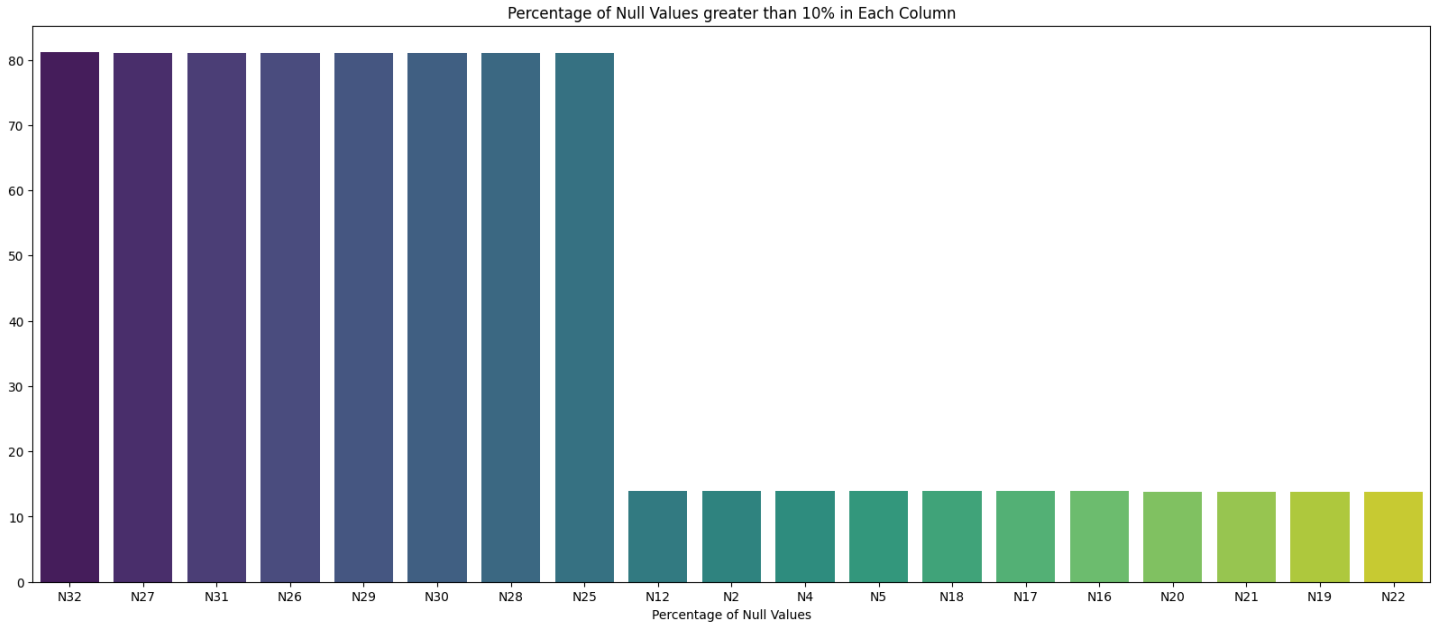
Step 1: Load the Data and Get Basic Information

The first step is to load the dataset into your environment. Once loaded, it's essential to get a basic understanding of the data. This includes:

- Checking the dimensions of the dataset (number of rows and columns).
- Inspecting the first few rows to understand the structure of the data.
- Using summary statistics to get an overview of numerical columns, such as mean, median, and standard deviation.
- Checking the data types of each column to identify categorical and numerical features.

Step 2: Data Preprocessing

This step involves dealing with missing data. The dataset appears to contain columns with a significant number of missing values, notably columns N32, N27, N31, and others, which have more than 80% null values**.**



Here's how you plan to handle this:

- Identify Columns to Remove: Given that columns with over 80% null values may not contribute meaningfully to the analysis, your approach is to remove these columns to reduce bias. Columns like N32, N27, N31, etc., fall into this category**.**

- Fill Missing Values: For the remaining columns with less than 80% null values, you plan to perform imputation:

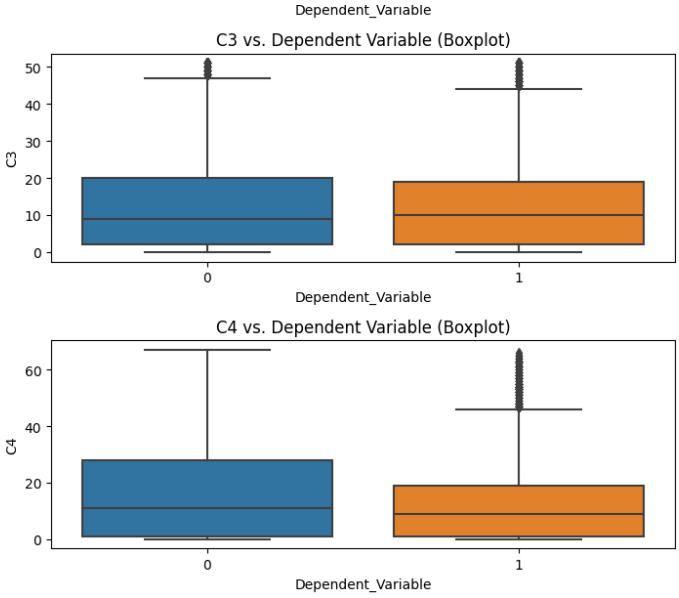
- Categorical Columns: For categorical columns, you will fill missing values with the mode (most frequent value) of that column. This ensures that categorical data remains categorical, and you don't introduce bias by imputing with a numerical value.

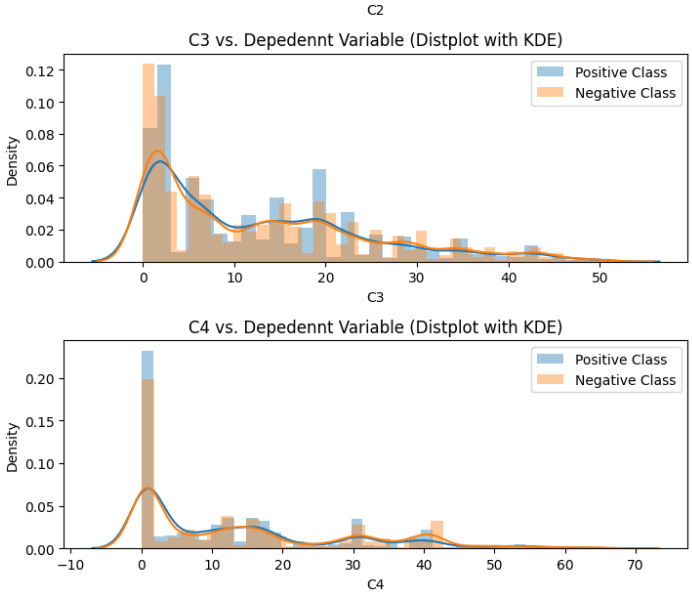
- Numerical Columns: For numerical columns, you will fill missing values with the median of that column. Using the median is a robust strategy as it's less sensitive to outliers compared to the mean.

Filling missing values with the median, rather than the mean, is preferred when dealing with outliers. The median is less sensitive to extreme values, making it a robust choice. It preserves the data distribution, maintains data integrity, and has a smaller impact on model performance when outliers are present. However, the choice depends on your dataset and goals.

Step 3: Exploring Data Distribution and Handling Outlier

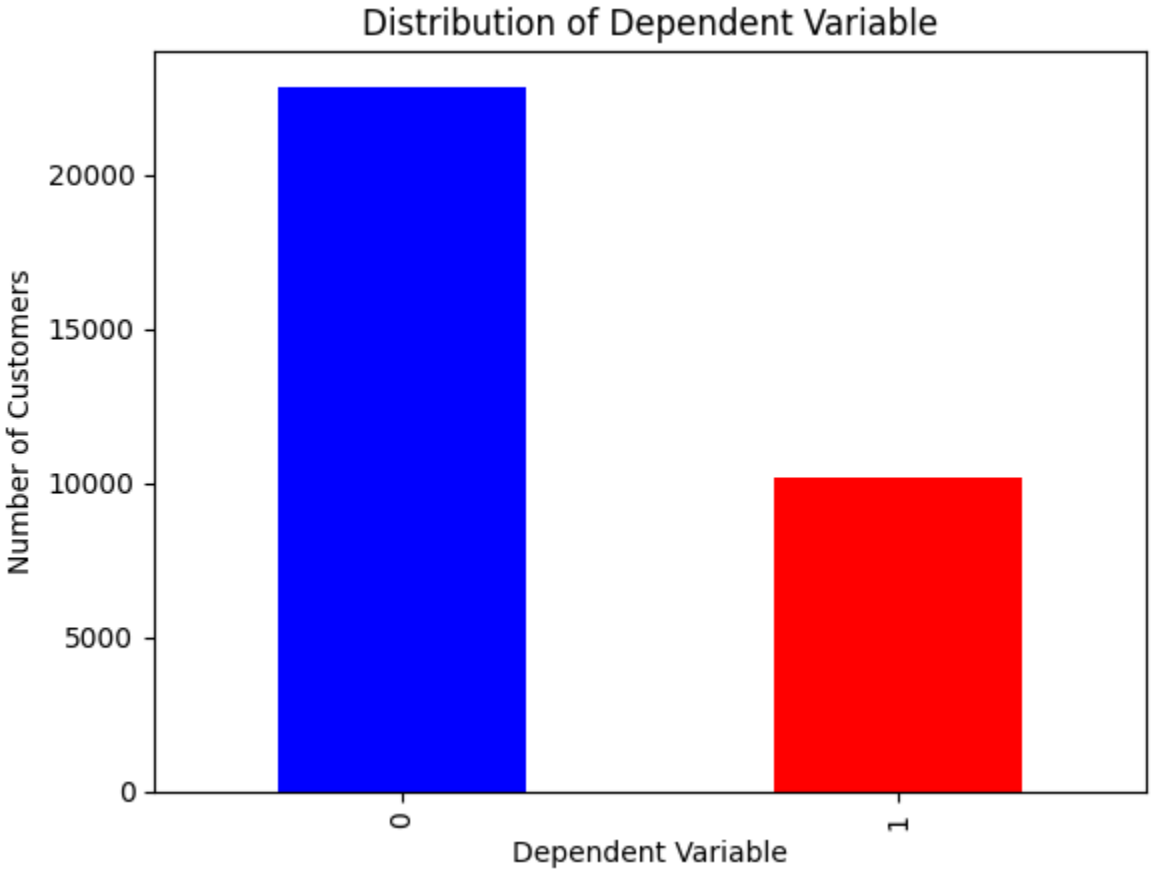
In this step, we analyze the data distribution to understand if there is significant skewness in the dataset. We utilize box plots to detect and visualize outliers. The presence of outliers influences our choice to use the median for missing value imputation rather than the mean.





Step 4: Addressing Class Imbalance

Our dataset exhibits a class imbalance, with 22,844 instances of class 0 and 10,206 instances of class 1. To mitigate this issue, we employ two techniques: **oversampling** and **undersampling**. Oversampling replicates the minority class instances to balance the class distribution, while undersampling randomly reduces the majority class instances. This rebalancing enhances the performance of our models.



Step 5: Data Standardization and Model Selection

After observing the data distribution and handling class imbalance, we decide to standardize the data to bring all features to the same scale. We also notice the presence of outliers, which is why we opt for tree-based algorithms, known for their robustness to outliers. We choose logistic regression and random forest for our models.

- Logistic Regression: Suitable for binary classification, we train it on both standardized data (including oversampled and undersampled datasets) and the original data (with oversampling).
- Random Forest: We use random forest for classification on standardized data (including oversampled and undersampled datasets) and the original data (with oversampling).

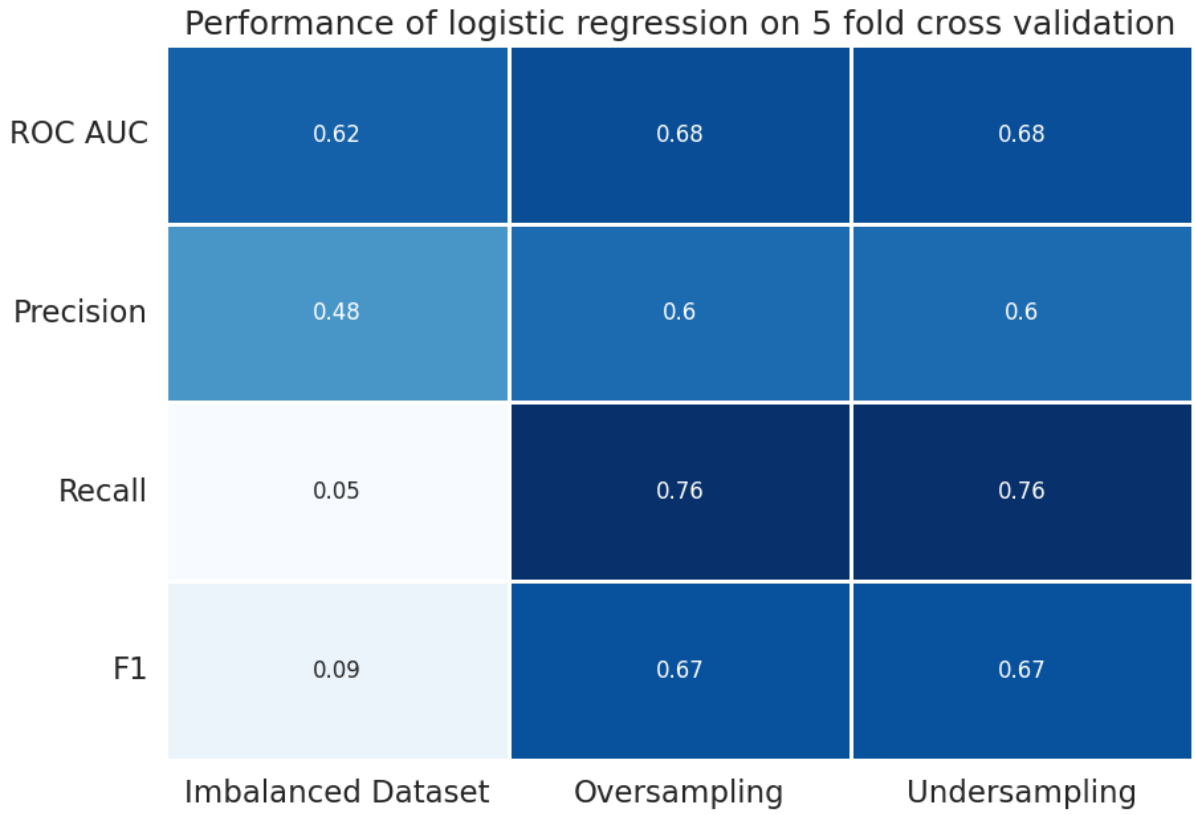
These steps aim to address data distribution, outliers, class imbalance, and model selection for a comprehensive approach to solving the problem.

Step 6: Model Selection, Evaluation and Comparison

In this step, we trained two different models: Logistic Regression and Random Forest, and evaluated their performance using 5-fold cross-validation.

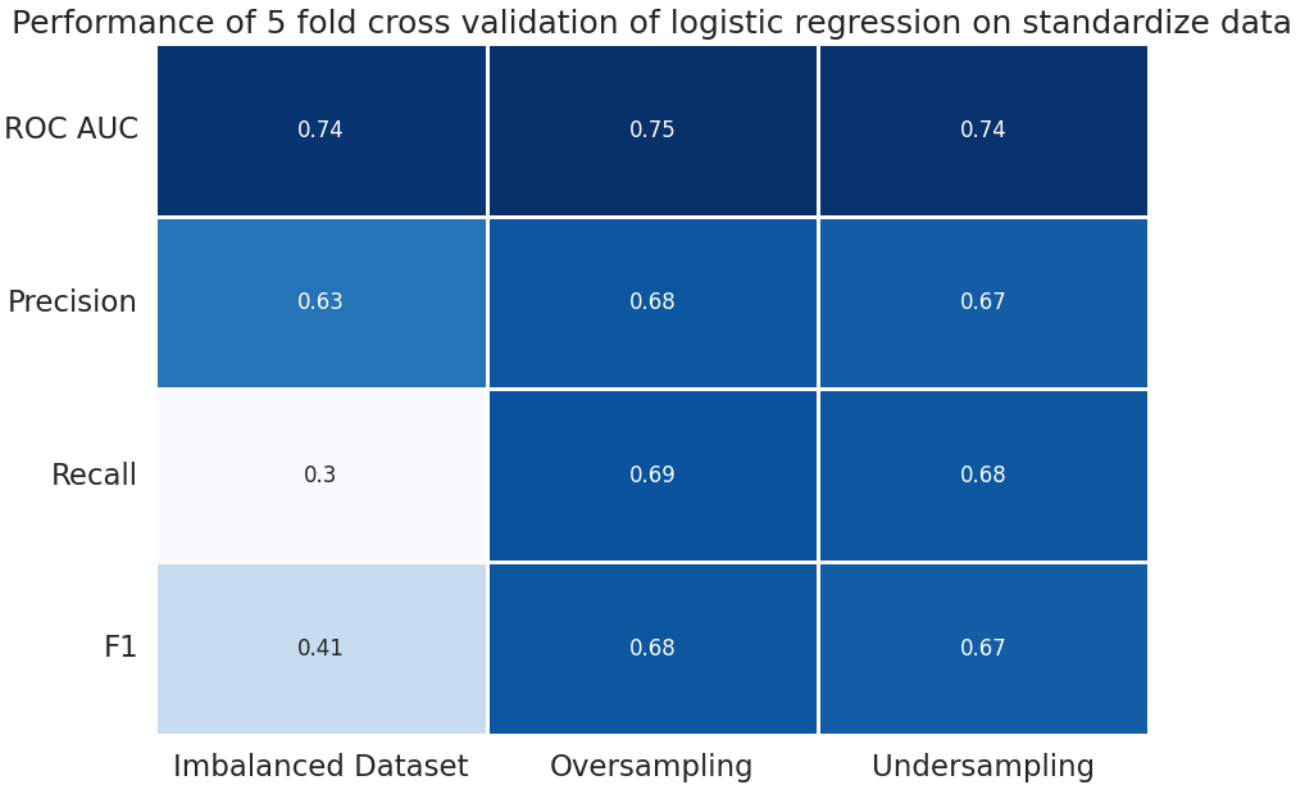
Logistic Regression:

- **Original data**
  - Original Data
  - Undersampling Data
  - Oversampling data



- **Standardized Data**
  - Original Standardized Data
  - Oversampling Standardized Data
  - Undersampling Standardized Data

Logistic Regression performs modestly on the original data but shows improvement when we introduce oversampling and undersampling. However, when using standardized data, we observe a notable increase in performance metrics. Notably, the ROC AUC score is higher, and precision, recall, and F1 scores are more balanced, indicating better overall model performance with standardized data. Standardization enhances the model's ability to generalize and make predictions.

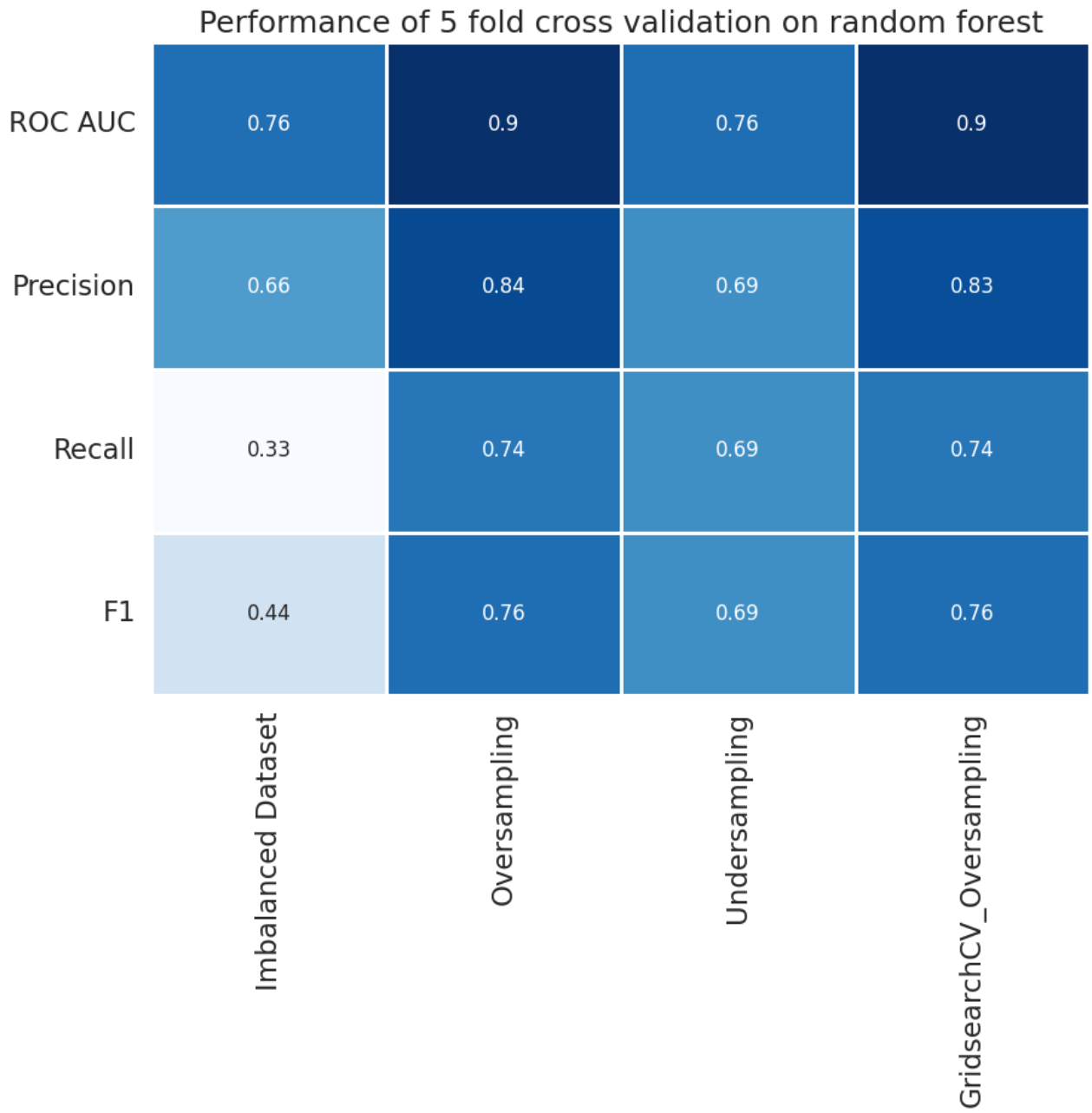


Comparing standardized and non-standardized data, the standardized data consistently outperforms in terms of ROC AUC and other metrics, making it the preferred choice for this problem.

Random Forest:

- Original data
- Undersampling Data
- Oversampling data

In our quest for better model performance, we turned to Random Forest, a more complex model known for its robustness against outliers and its ability to handle data with different scales. We trained three variations of the Random Forest model: using the original data, undersampled data, and oversampled data. The results were impressive, with the highest ROC AUC reaching 90%.



The Random Forest model on the original dataset showed promising results, especially in terms of ROC AUC. However, precision, recall, and F1 scores were somewhat unbalanced.

The oversampled dataset greatly improved model performance, with a remarkable ROC AUC of 0.896. This means the model is highly effective at distinguishing between the classes, resulting in excellent precision and recall scores. This is a significant improvement and a notable success.

The undersampled dataset performed consistently well, with balanced precision, recall, and F1 scores. While the ROC AUC is slightly lower than oversampling, it remains a strong choice.

In summary, Random Forest, particularly when trained on the oversampled dataset, outperforms Logistic Regression in all aspects. It exhibits exceptional ROC AUC and balanced precision and recall scores, making it the preferred choice for this problem.

Logistic Regression vs. Random Forest

- **Logistic Regression** had limitations with class imbalance, struggled with outliers, and achieved moderate ROC AUC scores (up to 0.74). It performed best with standardized data but had relatively lower recall and F1 scores.

- **Random Forest** excelled in handling class imbalance and outliers, consistently outperforming Logistic Regression. It achieved high ROC AUC scores (up to 0.896) and balanced precision and recall. The oversampled Random Forest model is the top choice for this task.

In a nutshell, Random Forest is the better choice, offering robustness and superior performance, especially in dealing with class imbalance and outliers.

Step 7: Hyperparameter Tuning for the Best Model

In Step 6, we identified that the Random Forest Classifier with oversampling performed the best among the models we experimented with. Now, let's further improve its performance by fine-tuning its hyperparameters using GridSearchCV.

Hyperparameter Tuning Process:

1. **Random Forest Classifier**: We will focus on optimizing the hyperparameters of the Random Forest Classifier, as it yielded the best results.

2. **Hyperparameters**: We will tune the following hyperparameters.
    - **n_estimators**: The number of trees in the forest.
    - **max_depth**: The maximum depth of the trees.
    - **min_samples_split**: The minimum number of samples required to split an internal node.
    - **min_samples_leaf**: The minimum number of samples required to be at a leaf node.

3. **GridSearchCV**: We will use GridSearchCV to perform an exhaustive search over a specified hyperparameter grid.

Results:

After running GridSearchCV, we can identify the best hyperparameters that maximize the ROC AUC score. These optimized hyperparameters will be used to train the final Random Forest Classifier.

Summary:

In this step, we focused on hyperparameter tuning for the Random Forest Classifier, which performed the best on the oversampled data. GridSearchCV was used to systematically explore a range of hyperparameters and identify the combination that produces the highest ROC AUC score. This fine-tuning process will help us achieve the best possible model performance for our classification task.

Step 8: Making Predictions and Preparing the Final Submission

Now that we have fine-tuned our model and everything is set, it's time to make predictions on the test dataset, perform the same preprocessing steps as we did for the training data, handle any missing values, and create the final submission file in CSV format.

Conclusion:

Throughout this machine learning project, we have covered various essential aspects of the machine learning workflow:

- **Data Understanding and Preprocessing:** We began by understanding the dataset, handling missing values, and performing exploratory data analysis.
- **Feature Engineering**: We created relevant features, especially focusing on text data, and transformed them into a suitable format for machine learning.
- **Model Selection**: We experimented with different machine learning algorithms and techniques, including oversampling for handling imbalanced data.
- **Model Evaluation and Hyperparameter Tuning:** We evaluated the models using appropriate evaluation metrics, and fine-tuned the best-performing model using GridSearchCV for optimal hyperparameters.
- **Prediction and Submission**: Finally, we applied our model to the test data, performed necessary preprocessing, and created the final submission file.

This comprehensive approach demonstrates a solid understanding of the end-to-end machine learning process, from data preprocessing to model deployment. It showcases your ability to work with real-world datasets, handle challenges like imbalanced data, and fine-tune models for the best performance, all of which are crucial skills in the field of machine learning.