HACKATHON
Foundations of Machine Learning

DOKKU HEMANADH
CS22BTECH11018

November 11, 2024

Introduction

This report outlines the data preprocessing steps, model selection, and evaluation process for a classification task. The goal was to enhance the model's performance through various preprocessing techniques, including handling missing values, feature selection, and class balancing. We experimented with multiple classifiers, including `RandomForestClassifier`, `GradientBoostingClassifier`, and `XGBClassifier`, and applied resampling techniques to address class imbalance.

Observations

1. Data Preprocessing

- **Scaling**: Two scaling techniques, **Min-Max Scaling** and **Standard Scaling**, were applied to the data. While these methods improved the **F1 score** during local testing, they did not have a significant impact on the final test results.
- **Pipeline**: We used a data processing pipeline to ensure consistent transformations across all steps. However, employing the pipeline did not lead to any noticeable improvement in model performance.
- **Missing Values**: We tested two imputation strategies for missing values: **mean imputation** and **median imputation**. **Mean imputation** outperformed median imputation for all missing values, providing better results across the dataset.
- **Column Removal**: Columns with more than **80%** missing values were removed to improve the quality of the dataset. Additionally, constant columns were dropped. Various thresholds for missing values were experimented with, but no further improvements in performance were observed.
- **Feature Selection**: We did not drop columns with low correlation to the target variable, as most of them had very low correlation. Columns exhibiting more than **90%** correlation with other features were removed to avoid redundancy and enhance model performance.

2. Data Observations

- **Class Imbalance**: The dataset exhibited a class imbalance, which was addressed using resampling techniques. **Oversampling** the minority class improved model performance, while **undersampling** the majority class resulted in a performance drop.

3. Model Selection

We evaluated three classifiers during the modeling phase:

- **RandomForestClassifier**: Initially, the `RandomForestClassifier` produced an **F1 score of 0.80** during local testing. However, the model's performance dropped significantly to **0.411** in the final test, likely due to overfitting or high model complexity.
- **GradientBoostingClassifier**: The `GradientBoostingClassifier` achieved an **F1 score of 0.50** locally, with a slight drop to **0.436** in the final test. Although the training time was longer, no column removal (aside from those with more than 80% missing values) was conducted.
- **XGBoost**: With hyperparameter adjustments, the `XGBoost` model reached an **F1 score of 0.439** and **0.440** after removing irrelevant columns like `UID`. However, it failed to perform well on private test cases, and we chose not to continue with it.

Conclusion

- **Data Preprocessing** techniques, including scaling and feature selection, led to some improvement in local performance but did not significantly impact the final test outcomes.
- **Resampling** techniques, particularly **oversampling** the minority class, effectively addressed class imbalance and improved model performance.
- The `XGBoost` model demonstrated potential, achieving the highest F1 scores but ultimately failed in private test scenarios. Despite this, it was the best-performing model compared to `RandomForest` and `GradientBoosting`.
- **Grid Search** was crucial in optimizing the hyperparameters of each model, ensuring that the best-performing configurations were selected.

Contributions

My Contribution:

- Focused on data preprocessing, including handling missing values, scaling (Min-Max and Standard), and removing columns with more than 80% null values or constant values.
- Experimented with resampling techniques to address class imbalance.
- Worked on model experimentation with the `RandomForestClassifier`, testing different parameters and adjusting data splits.