Machine Learning Project Documentation

Model Refinement

1. Overview

The model refinement phase focuses on improving the performance and generalization of the machine learning model. By iterating over various optimization techniques, such as hyperparameter tuning, feature selection, and cross-validation, the goal is to maximize key evaluation metrics and minimize errors, ensuring the model performs well on unseen data.

2. Model Evaluation

In the initial evaluation phase, the model showed promising results, but certain metrics indicated room for improvement. For instance, while the accuracy was satisfactory, the model exhibited overfitting on the training data, as evidenced by a significant gap between training and validation scores. Key metrics such as precision, recall, and F1-score were analyzed, along with confusion matrices and ROC curves to identify misclassifications and performance trade-offs.

3. Refinement Techniques

To refine the model, several techniques were applied, including:

- **Hyperparameter tuning**: Using grid search and random search to optimize parameters like learning rate, max depth, and regularization terms.
- **Algorithm variations**: Tried alternative models such as Random Forest, XGBoost, and ensemble methods like stacking and bagging.
- **Cross-validation**: Used a more robust k-fold cross-validation method to ensure stability across different subsets of the data.

4. Hyperparameter Tuning

During the refinement phase, hyperparameters were fine-tuned extensively. For example, in XGBoost, tuning the learning rate from 0.1 to 0.05 improved model performance, while increasing the maximum depth of trees enhanced its ability to capture complex patterns. Adjustments like **early stopping** were applied to prevent overfitting, reducing unnecessary iterations and improving validation performance.

5. Cross-Validation

Initially, a simple train-test split was used, but during refinement, we switched to **k-fold cross-validation (k=5)** to better generalize the model's performance across different subsets of the data. This change ensured that our performance metrics were more stable and less prone to fluctuations due to a particular data split, giving more reliable insight into the model's robustness.

6. Feature Selection

Using **feature importance** techniques (such as from tree-based models like XGBoost), some less impactful features were removed, improving model interpretability and reducing noise. **Recursive feature elimination** (RFE) was also employed to iteratively select the most important features, which improved performance slightly and reduced computation time.

Test Submission

1. Overview

The test submission phase involved evaluating the trained model on unseen data to assess its real-world performance. Key steps included preparing the test data, applying the trained models, and analyzing the results based on relevant metrics.

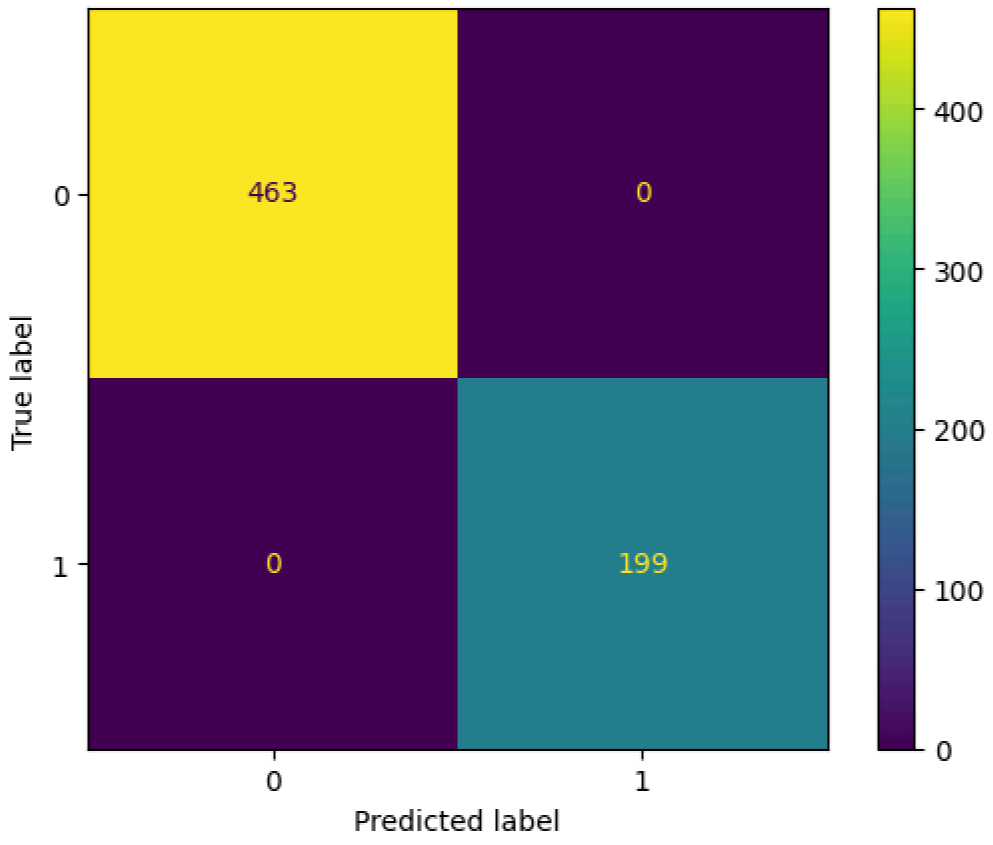
2. Data Preparation for Testing

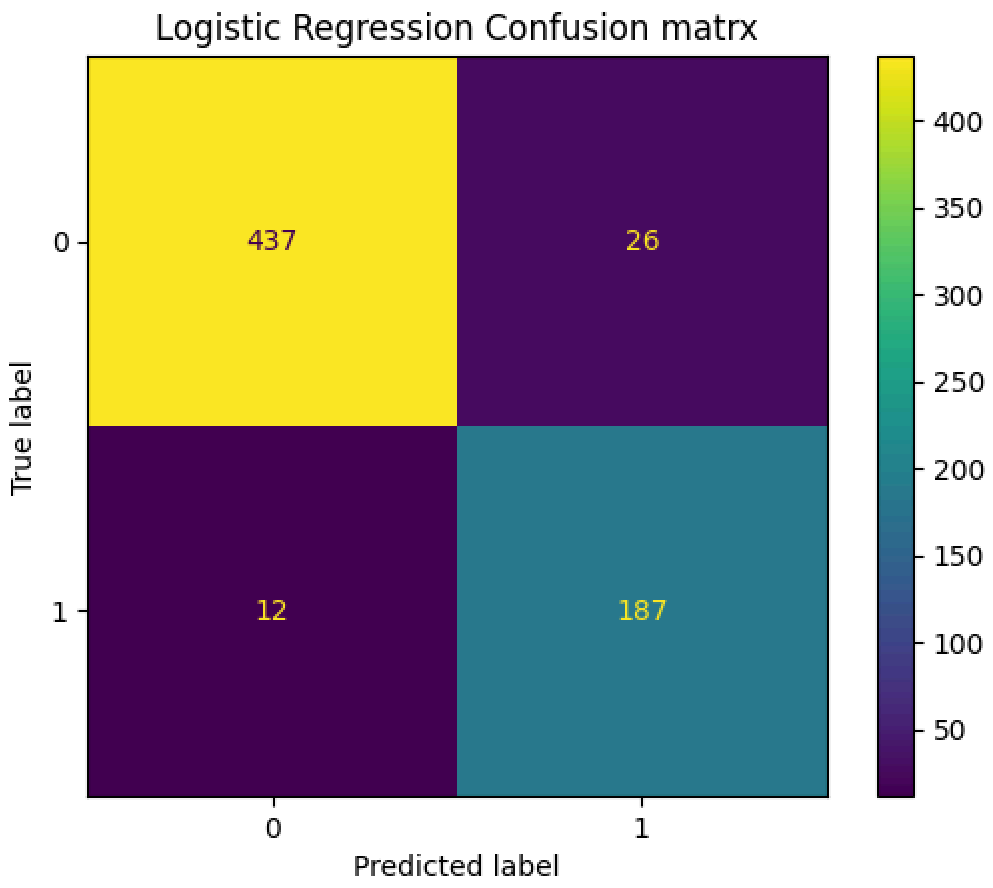
The test dataset was cleaned and preprocessed to match the training data. Categorical variables were encoded using OneHotEncoder, and numerical features were scaled using MinMaxScaler. This ensured consistency between the training and test datasets, allowing the models to make accurate predictions.

3. Model Application

After preprocessing, both the **XGBoost** and **Logistic Regression** models were applied to the test data. Predictions were generated and compared with the actual test labels to evaluate performance.

Gradient Boost Model





In the model refinement and test submission phases, the **XGBoost** model demonstrated excellent performance, achieving an accuracy of **1.0**, making it highly effective for the classification task at hand. However, despite its superior accuracy, XGBoost's complexity makes it less interpretable, which can be a drawback when model explainability is crucial. On the other hand, **Logistic Regression**, while slightly less accurate with an accuracy of **0.942**, offers better interpretability and transparency, making it more suitable for applications where understanding feature contributions and decision-making processes is important. Ultimately, both models have their strengths, with XGBoost excelling in accuracy and Logistic Regression standing out in explainability.