GST Analytics Hackathon

Team ID: GSTN_539

Team Member(s): Sarang Dave

Summary of the project

- This project focuses on a supervised binary classification task that includes advanced techniques for handling missing data, resampling to manage class imbalance, and extensive hyperparameter optimization to improve model performance.
- I have trained two powerful machine learning classifiers, **XGBoost** and **LightGBM**, alongside two ensemble learning techniques, **VotingClassifier** and **StackingClassifier**. Missing data was addressed using **iterative imputation**, ensuring a more informed approach to handling null values. This was followed by comprehensive hyperparameter tuning to further enhance the models' accuracy and robustness.
- The primary challenge was to balance effective data imputation without introducing overfitting, while ensuring that missing values were properly accounted for to maintain model robustness.

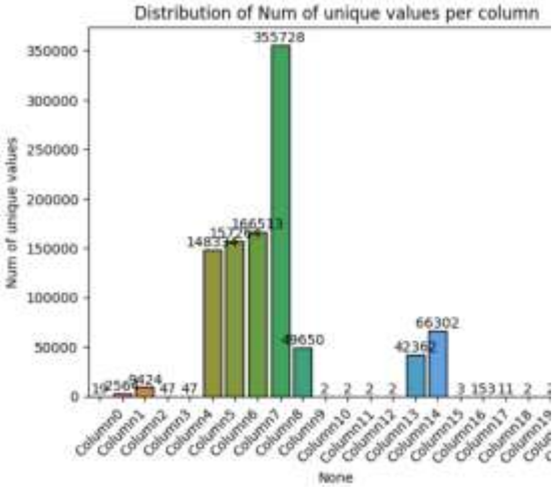
Disclaimer

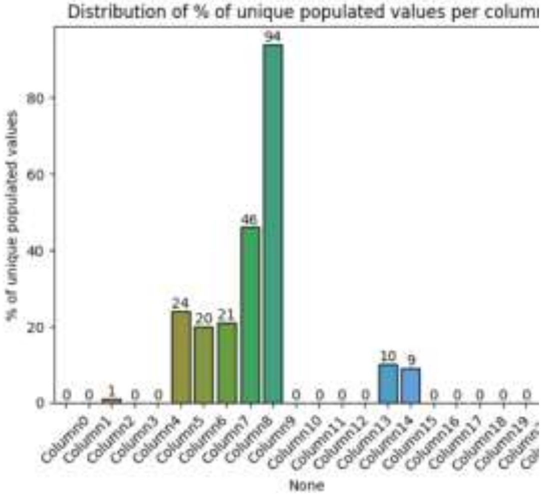
- The numbers quoted in this presentation may vary slightly due to the inherent non-deterministic nature of the predictive algorithms.
- However, the trends in the data and results remain identical.
- Only the supporting visualizations are shown in this presentation. To view all the visualizations, check out the main.ipynb file.

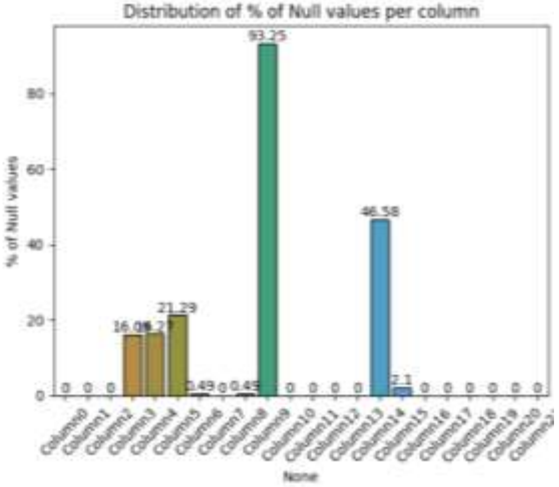
Approach

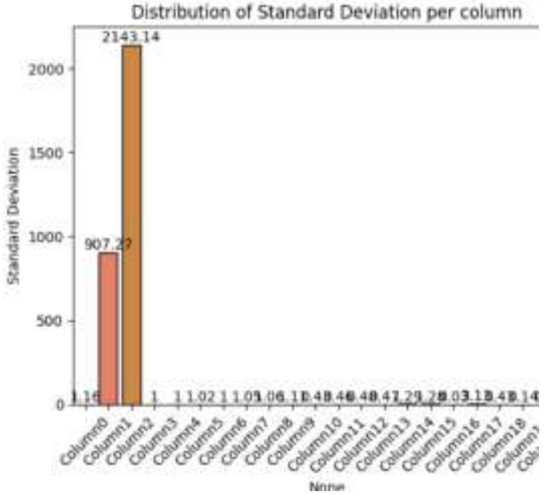
Exploratory Data Analysis

- Conducted statistical analysis and visualized trends such as **number of unique values**, **unique populated values**, **missing values**, **standard deviation**, and **variance** per column.
- Visualized the class distribution in the target variable.
- Visualized the data distribution in all the features.
- Identified numeric and categorical columns, preparing the data for preprocessing.





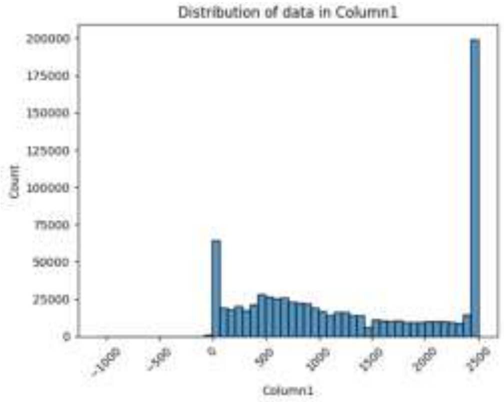


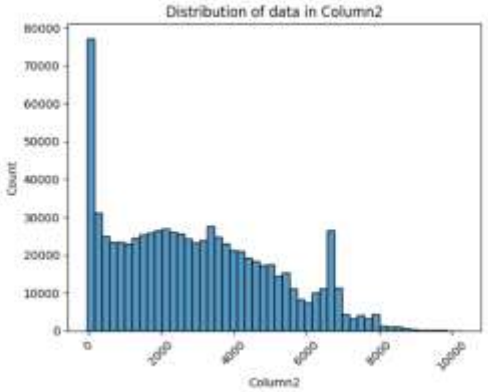


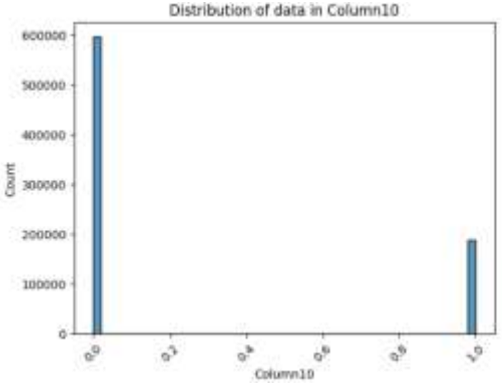
Approach

Exploratory Data Analysis

- Here are additional visualizations showing general trends in the training dataset.
- Histograms were created to visualize data distribution for each column, helping to identify which columns are numeric and which are categorical.
- I also analyzed the presence of missing values and unique entries in each column, providing insights into potential data quality issues.
- These visualizations were crucial in guiding the preprocessing steps that followed.



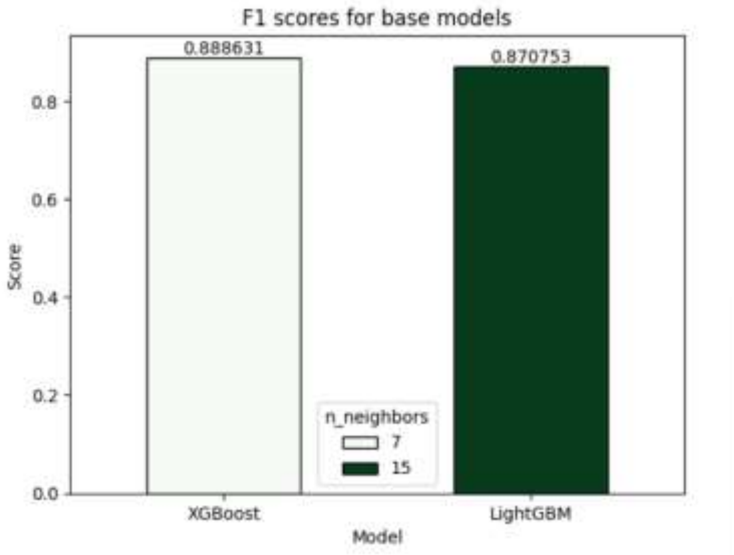


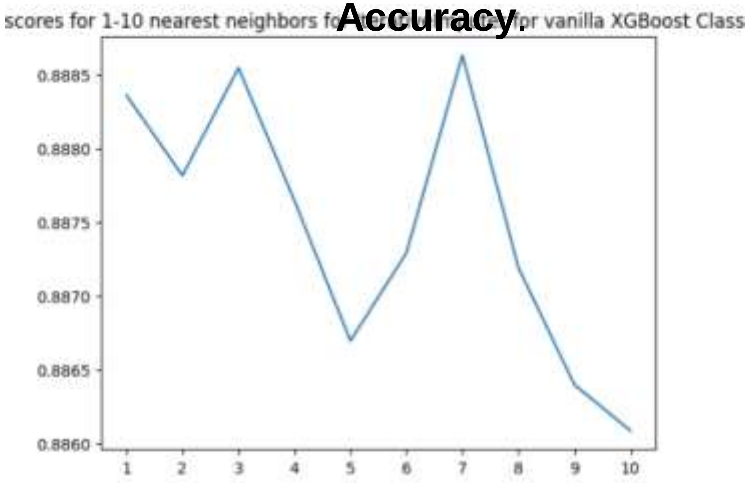


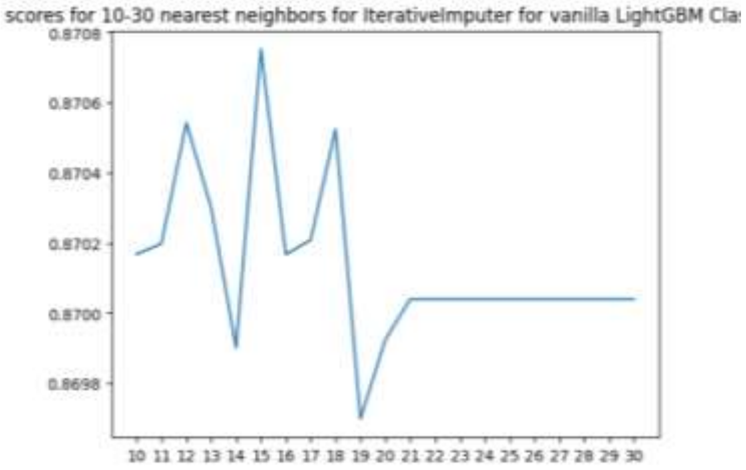
Approach

Data Preprocessing

- Split the data into training and validation sets using a stratified approach to preserve class balance.
- Identified the optimal parameters for IterativeImputer for both base models (**XGBoost** and **LightGBM**) to achieve the **highest F1 score**.
- Created copies of the training and validation datasets, imputing them with the optimal values found in earlier steps for XGBoost and LightGBM.
- Evaluated the performance of the vanilla base models on these imputed datasets using metrics such as **Accuracy**, **Precision**, **Recall, F1 Score**, **AUC-ROC**, **Log Loss**, and **Balanced Accuracy**.



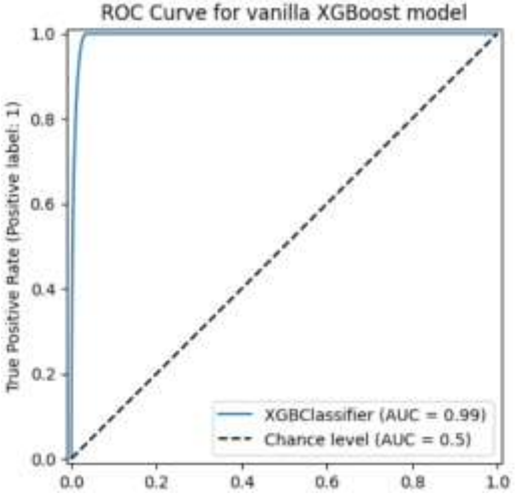


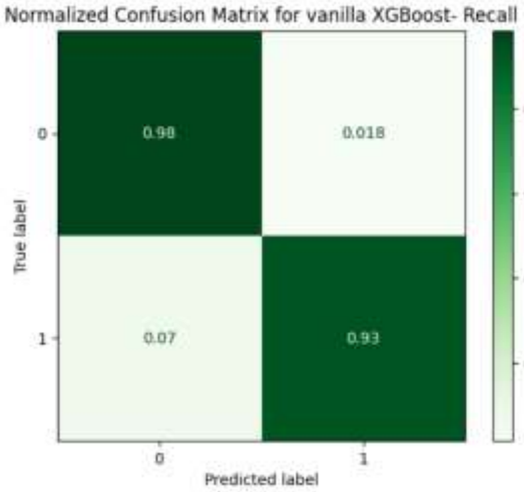


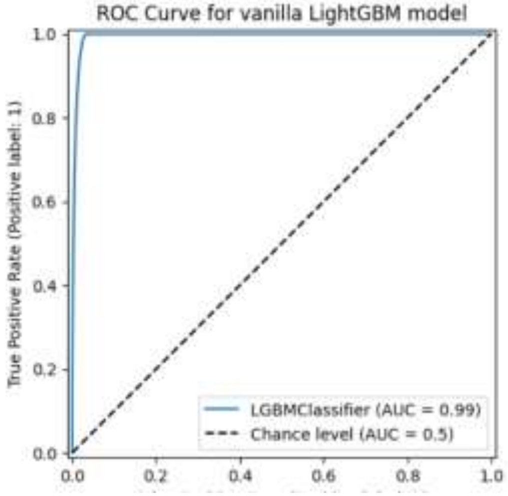
Approach

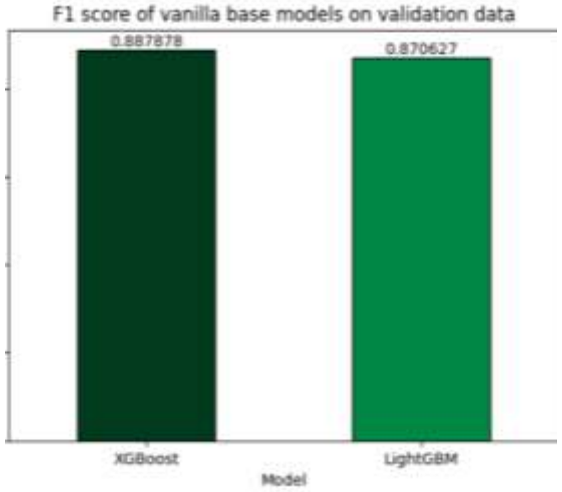
Data Preprocessing

- Applied scaling to the input data to assess its impact on model performance.
- Generated normalized confusion matrices for both models to visualize classification results.
- Here are additional visualizations for the data preprocessing stage.





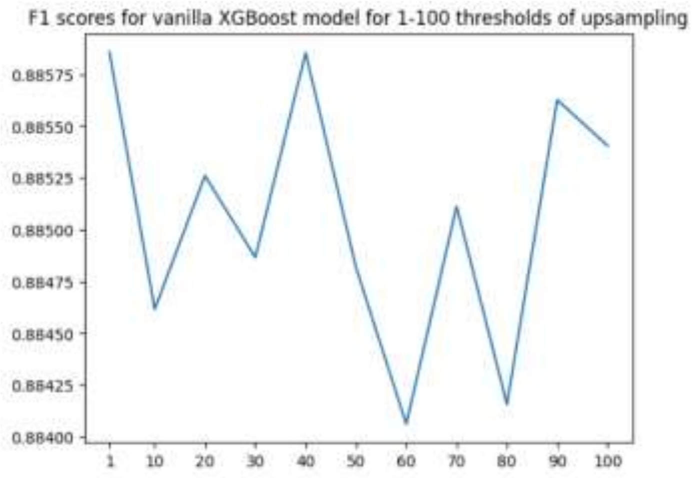


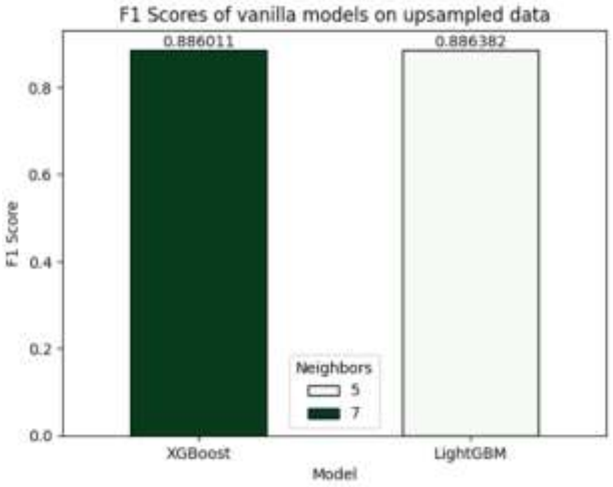


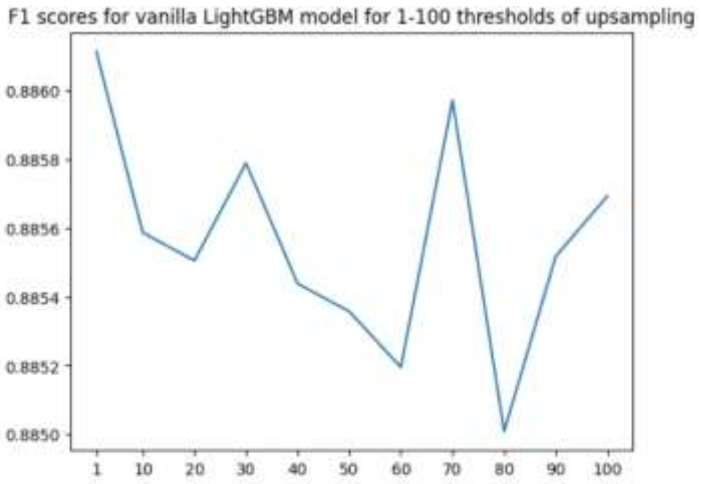
Approach

Handling Class Imbalance

- Utilized **SMOTE (Synthetic Minority Over-sampling Technique)** to address class imbalance and enhance the model's ability to predict the minority class.
- Iterated through various parameter values to identify the optimal SMOTE settings for both XGBoost and LightGBM, maximizing the F1 score.
- Compared the best F1 scores for both vanilla base models on datasets that were imputed and upsampled using the tuned SMOTE parameters.



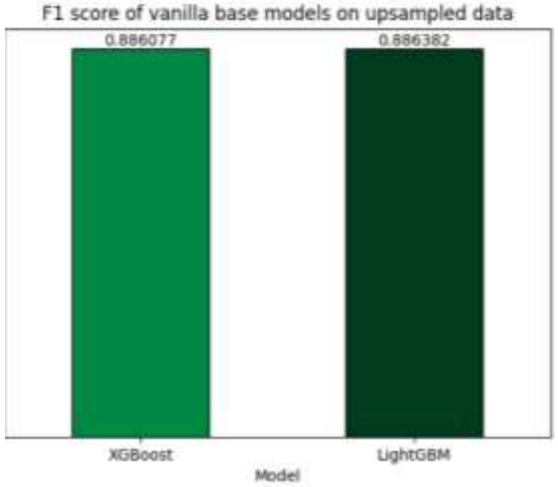


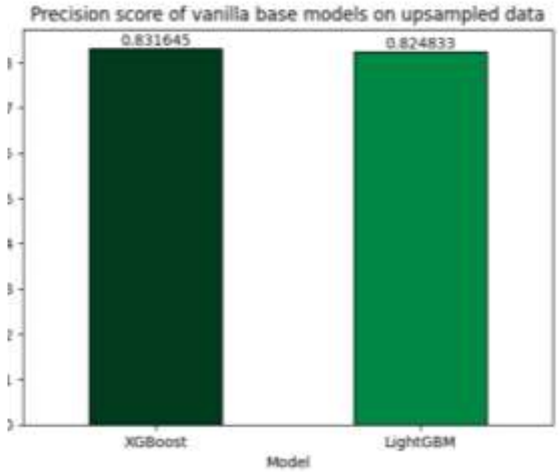


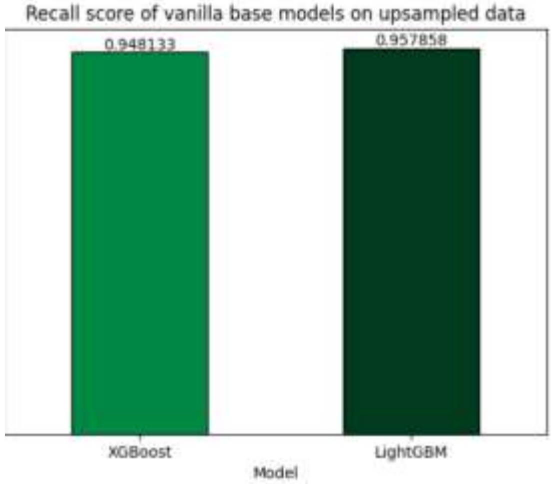
Approach

Handling Class Imbalance

- Upsampled each dataset using the optimal SMOTE parameters identified earlier for XGBoost and LightGBM.
- Verified the class distribution in the target variable after upsampling to **ensure balanced representation** of the **minority class**.
- Trained the base models on the newly imputed and upsampled datasets, and evaluated performance across multiple metrics, including Accuracy, Precision, Recall, F1 Score, AUC-ROC, Log Loss, and Balanced Accuracy.



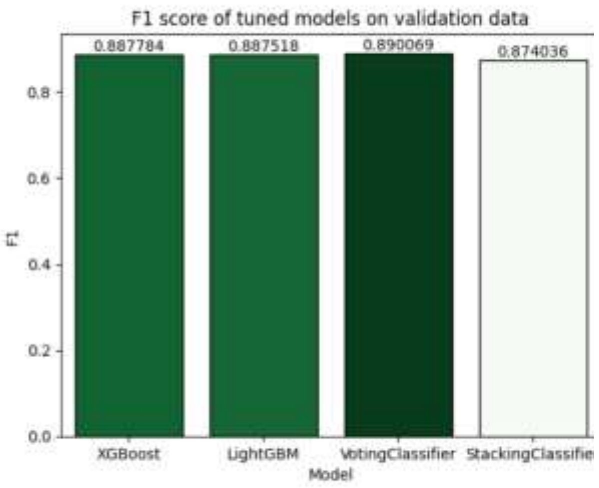


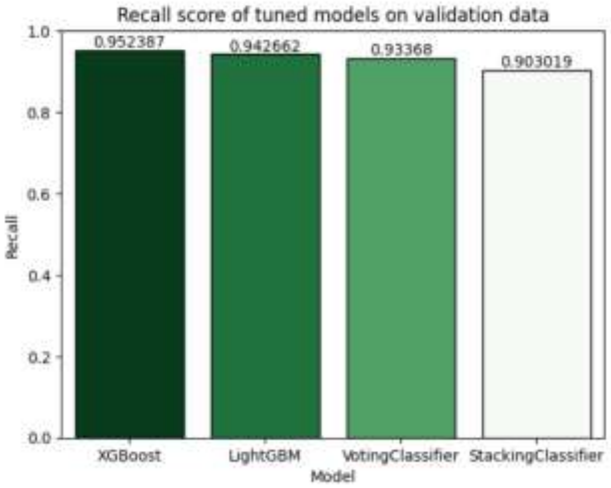


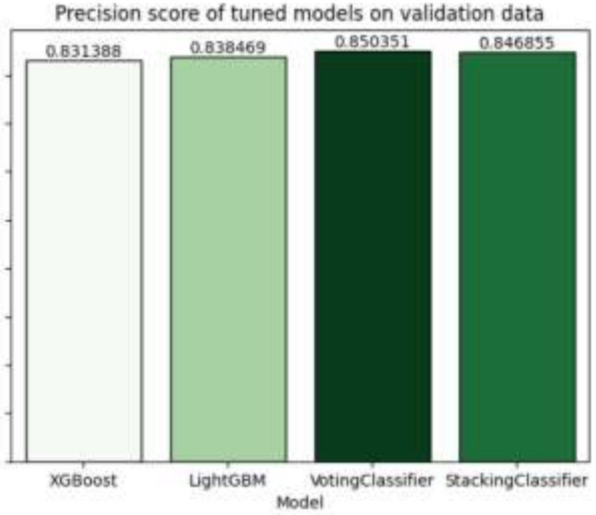
Approach

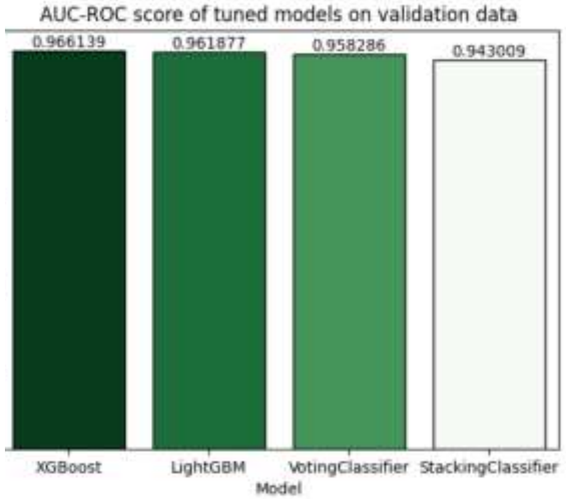
Model Building, Optimization, and Evaluation

- Built and evaluated base models: **XGBoost** and **LightGBM**.
- Applied **Optuna** for hyperparameter tuning to optimize model performance.
- Developed ensemble models, including **VotingClassifier** and **StackingClassifier**, to combine the strengths of individual models.
- Assessed all models using multiple performance metrics: **Accuracy**, **Precision**, **Recall**, **F1 Score**, **AUC-ROC**, **Log Loss**, and **Balanced Accuracy** on validation data and test data.





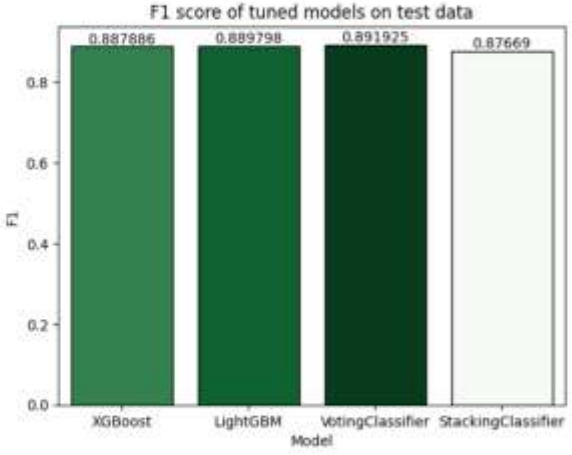


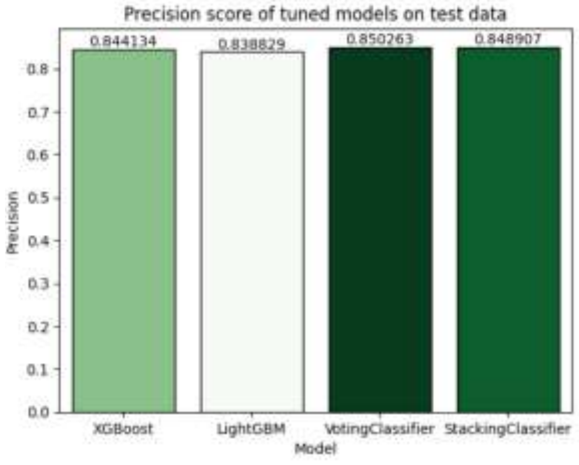


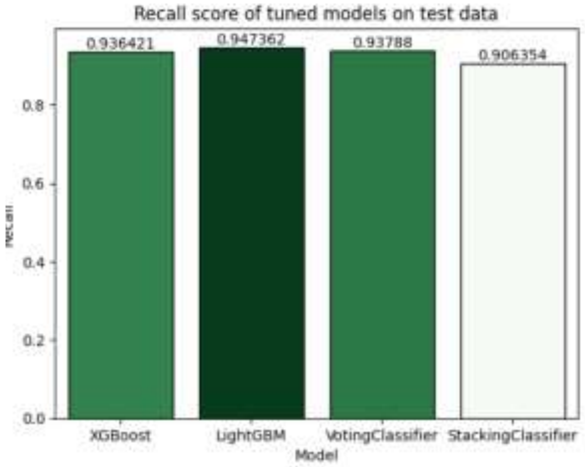
Approach

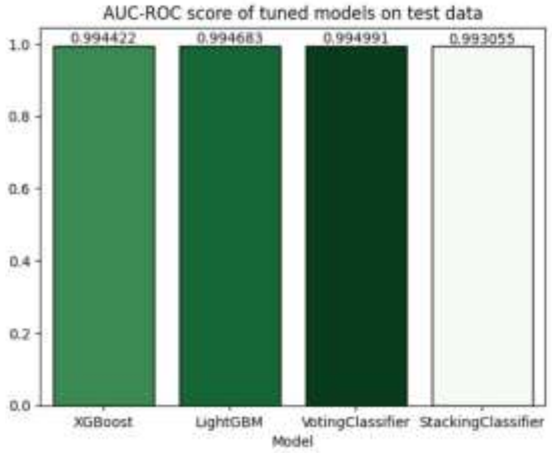
Model Building, Optimization, and Evaluation

Here are some more visualizations of the model performance on the test data





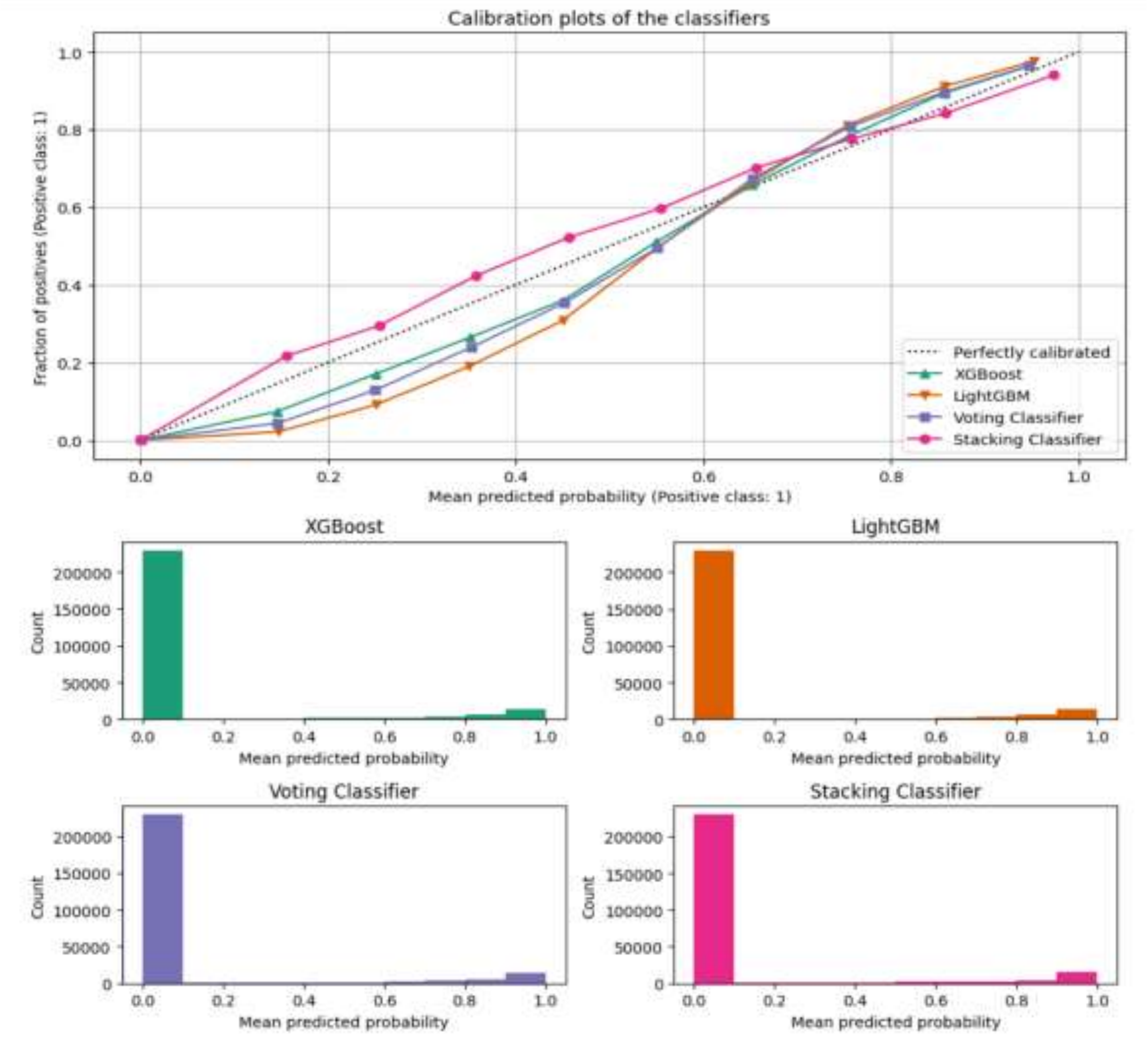




Approach

Model Building, Optimization, and Evaluation

Below are the calibration plots of the classifiers, showcasing each model's performance visually



Findings - Models

- **XGBoost** demonstrated strong performance, achieving a high **AUC-ROC score of ~0.994**, indicating excellent distinction between classes. Its **Precision and Recall were balanced**, resulting in a solid **F1 Score of ~88.79%**. Although **Log Loss was slightly higher**, it maintained competitive accuracy across both classes, showing it's a good fit for the dataset.

- **LightGBM** performed slightly better than XGBoost with an **F1 Score of ~88.98%** and an **AUC-ROC score of ~0.995**. LightGBM's strength lies in its efficiency with large datasets. **Precision was slightly lower than XGBoost**, meaning it identified fewer positive cases, **but its Recall was stronger**, indicating it often correctly predicted positives. **Log Loss was marginally lower**, reflecting greater confidence in its probability predictions.

Findings - Models

- The **VotingClassifier was the top performer**, achieving the **highest scores in nearly all key metrics** (except Recall). It outperformed XGBoost and LightGBM by combining their predictions. It had the **highest Accuracy (~97.9%)** and **F1 Score (~89.2%)**, showcasing the benefits of leveraging multiple classifiers. With an **AUC-ROC score of ~0.995** and the **lowest Log Loss**, the **VotingClassifier provided the most accurate probability estimates**.

- The **StackingClassifier** was a close runner-up, delivering high scores in all key metrics, including **Accuracy (~97.6%)**, **F1 Score (~87.7%)**, and **AUC-ROC (~0.993%)**. This approach, where a meta-learner makes the final prediction after combining several models, proved to be an effective solution for this problem. *(Note: The hyperparameters of the meta learner were not optimized due to high computational requirements)*.

Findings - Miscellaneous

- **SMOTE significantly improved the model's ability to predict the minority class**, reducing the class imbalance issue and enhancing Recall and Precision for the minority class.

- Imputation and scaling contributed to performance but had a moderate effect compared to class balancing techniques like SMOTE.

- **Ensemble models outperformed individual models** in terms of balancing Precision, Recall, and Log Loss.

Recommendations

- **Use VotingClassifier**: For future projects, leveraging the VotingClassifier is recommended for its superior performance in combining multiple models.
- **SMOTE for Imbalanced Data**: Continue using SMOTE when dealing with datasets with significant class imbalance, as it showed notable improvements in minority class predictions.
- **Iterative Imputation**: Use IterativeImputer for handling missing data but focus on tuning the number of neighbors to ensure optimal imputation accuracy.
- **Model Calibration**: Monitor the calibration of models for high-confidence predictions, especially for applications where the cost of incorrect predictions is high.
- **StackingClassifier**: Tune the hyperparameters of the meta-estimator of StackingClassifier to improve its performance.

The end.

Thank you for watching.

In case you want to contact me, my contact information is as follows:

Email-ID: davesarang08@gmail.com

Phone Number: +91 9082783721

LinkedIn: https://www.linkedin.com/in/sarang-dave/

GitHub: https://github.com/S84v