# GST Analytics Hackathon - Documentation

---

This documentation presents the development and evaluation of machine learning models for the GST Analytics Hackathon project. The primary objective of the project was to build a robust predictive model capable of accurately classifying data while effectively handling missing values. The dataset presented challenges such as missing data and class imbalance, which required advanced preprocessing techniques and extensive model tuning.

To address these challenges, a combination of machine learning algorithms, including RandomForest, XGBoost, and LightGBM, was employed. The workflow involved imputation of missing data using IterativeImputer, followed by model training, hyperparameter tuning, and evaluation across various performance metrics such as Accuracy, Precision, Recall, F1 Score AUC-ROC, Log Loss and Balanced Accuracy.

The report delves into the methodology, performance analysis, and insights derived from the predictions made by these models. It also includes recommendations for further model improvements and suggestions on how to enhance model performance. Various visual aids, such as performance plots and confusion matrices, support the findings, ensuring a comprehensive understanding of the model's effectiveness in solving the classification problem.

Clear and well documented code used to build train and test the submitted models can be found in the *main.ipynb* file.

Following is the key methodology and steps taken in the model development:

1. **Exploratory Data Analysis:** The initial phase involved performing a comprehensive statistical analysis of the dataset. This included identifying key trends such as the number of unique values in each feature, the proportion of missing values, and the distribution of populated values. Visualizations were created to better understand these trends and patterns. Based on the analysis, we inferred which columns were likely numeric and which were categorical, despite the dataset being anonymized. This step was crucial for determining the appropriate preprocessing steps for each feature type.

2. **Data Preprocessing**: The dataset was split into training and validation sets using a stratified approach to ensure an even distribution of classes. Missing values were handled using the IterativeImputer, which imputes missing data based on relationships between features. A missing indicator was added to the dataset to flag where imputations were made, allowing the model to account for missingness. The best parameters for the **IterativeImputer** were identified by evaluating the performance of XGBoost and LightGBM models on the validation set using a vanilla configuration.
3. **Model Evaluation on Validation Dataset**: After imputing the missing values, the performance of the base models (XGBoost and LightGBM) was evaluated on the validation set. Multiple performance metrics were considered, including **Accuracy, Precision, Recall, F1 Score, AUC-ROC, Log Loss,** and **Balanced Accuracy**. Normalized confusion matrices were also generated to assess true positives (recall) and predicted positives (precision). These visualizations provided clear insights into the strengths and weaknesses of each model's predictions.
4. **Scaling Input Data**: After determining the optimal imputation parameters, input data scaling was tested to see if it improved model performance. Standard scaling techniques were applied to ensure that features had the same range, which can benefit gradient-boosting algorithms like XGBoost and LightGBM. Additionally, the distribution of the target variable was visualized before applying any upsampling to better understand class imbalances.
5. **Handling Class Imbalance with SMOTE**: To address class imbalance, we applied **SMOTE** (Synthetic Minority Over-sampling Technique), which generates synthetic data points for the minority class. This technique helps reduce overfitting and enhances model robustness by balancing the dataset. Optimal parameters for SMOTE were identified for both base models, and performance metrics were re-evaluated on the upsampled datasets, with visualizations comparing the performance of models before and after SMOTE application.
6. **Hyperparameter Optimization: Optuna**, an open source hyperparameter optimization framework, was employed to automate the search for the best hyperparameters for both XGBoost and LightGBM. Optuna uses a dynamic approach to optimize hyperparameters by conducting efficient and flexible searches, improving the performance of the base models. The optimized models were saved for further evaluation.
7. **Building Ensemble Models:** Two ensemble models were developed: the **VotingClassifier** and the **StackingClassifier**. The VotingClassifier aggregates the predictions of multiple

base classifiers (XGBoost, LightGBM, etc.) and makes a final prediction based on the majority vote. The StackingClassifier takes a more sophisticated approach by combining multiple base classifiers and using a meta-classifier to make the final prediction. This approach leverages the strengths of different models, improving overall predictive power.

8. **Final Model Evaluation on Test Dataset:** Comprehensive data preprocessing pipelines were created to combine all the preprocessing steps (e.g., imputation, scaling, upsampling) for the base models. These pipelines were used to evaluate the performance of the optimized XGBoost, LightGBM, VotingClassifier, and StackingClassifier models on the test dataset. The same performance metrics (Accuracy, Precision, Recall, F1 Score, AUC-ROC, Log Loss, and Balanced Accuracy) were calculated and visualized. Normalized confusion matrices were generated for each model. Finally, a calibration plot was created to compare the performance and calibration of all classifiers, providing insights into how well the predicted probabilities aligned with actual outcomes.

This methodology ensured that every step was optimized for model performance, from preprocessing to hyperparameter tuning and ensemble learning, resulting in a robust predictive model suitable for deployment.