GST Analytics Hackathon Report: AI and ML Solutions for GST Data

Pushkar Raj, Pratyush Tiwari, Tanish Maheshwari, Ritu Raj, and Shivam Singh

Problem Statement

The challenge in this hackathon was to develop a machine learning model capable of accurately predicting the target variable, using the provided GST data. The task is framed as a *binary classification* problem, where we aim to correctly identify the target based on input features, improving efficiency in GST analysis.

Approach

Our team addressed a binary classification problem, aiming to categorize data into two classes: positive and negative. We constructed a machine learning model implementing XGBoost, an efficient algorithm, to generate precise predictions. This report outlines our approach to the findings, and the practical applications of this model in real-life scenarios. Our approach involve following steps:

I. **Data Preprocessing**: Missing values were handled using median imputation. The dataset was normalized to ensure uniform feature scaling.

II. **Feature Selection**: We used the SelectKBest method to retain relevant features based on their ANOVA F-values.

III. **Model Selection**: XGBoost was selected due to its superior performance in binary classification problems.

IV. **Hyperparameter Tuning**: We employed Optuna for optimizing hyperparameters like learning rate, max depth, and the number of estimators.

V. **Evaluation**: The model was evaluated using metrics such as Accuracy, Precision, Recall, F1 Score, AUC-ROC, and a Confusion Matrix.

Objective

The task given to us was to build a model that predicts a target variable based on input data. To ensure that our model could perform accurately in real-world situations, we evaluated it on multiple metrics, such as accuracy, precision, recall, F1 score, and AUC-ROC. These metrics help us measure how well our model classifies the data.

Methodology

I. Data Preprocessing

We began by cleaning the data to handle any missing values. This step is crucial to ensure that our model learns from high-quality information. After cleaning, we standardized the data (bringing all values onto the same scale), which helps the model work more efficiently.

II. Feature Selection

We selected the most important features (characteristics) from the dataset. Feature selection helps the model focus on the variables that contribute most to the prediction.

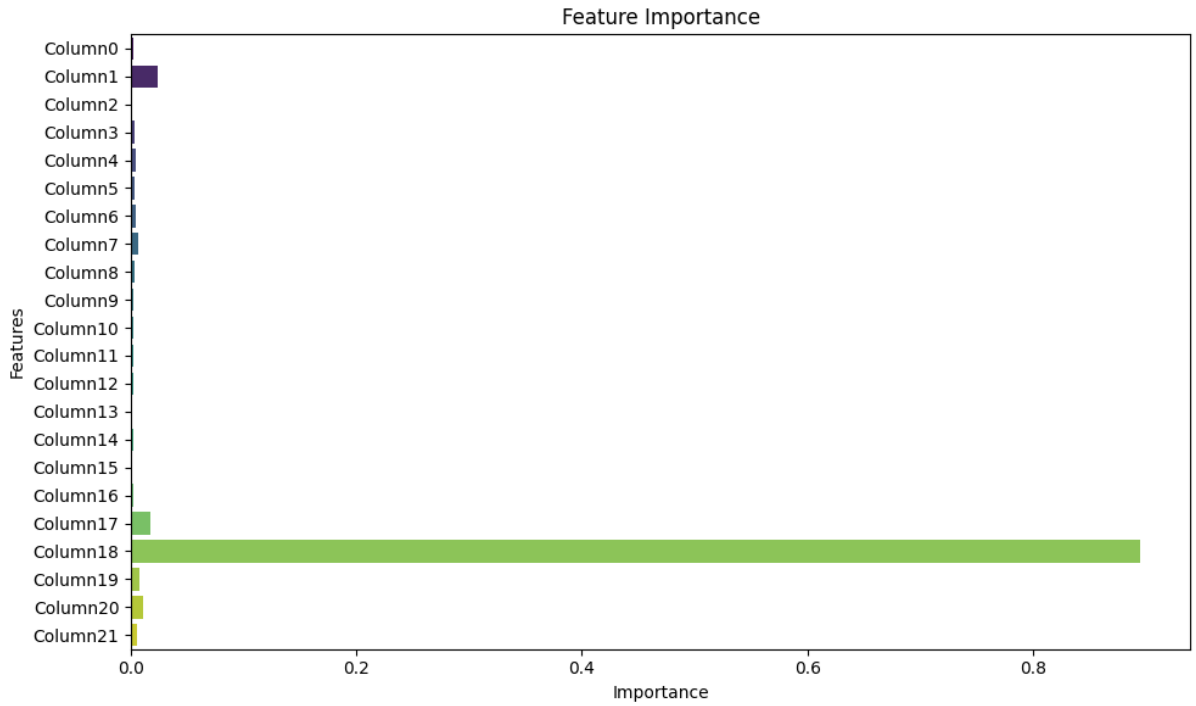

Figure 1: Bar diagram showing the importance of each feature.

III. Model Training and Tuning

We used XGBoost, a robust classification algorithm known for its efficiency and accuracy. To fine-tune the model's performance, we employed *Optuna*, a hyperparameter optimization tool. This process helped us find the best possible combination of model settings.

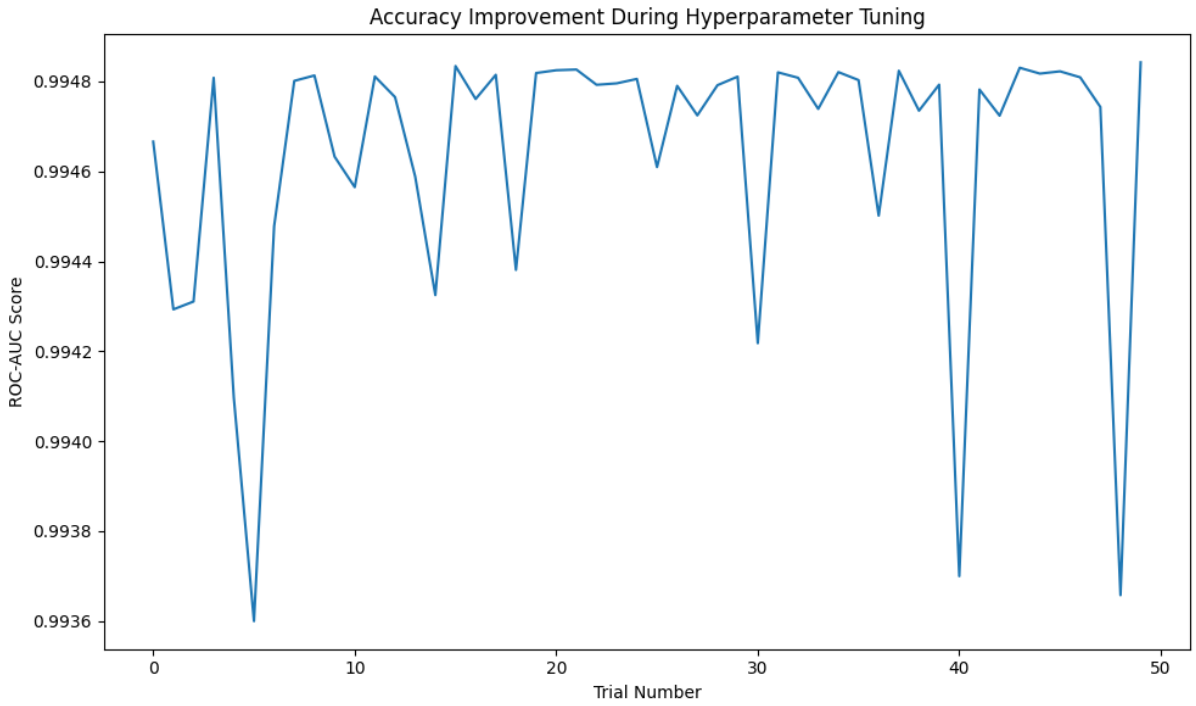

Figure 2: Graph showing accuracy improvement during hyperparameter tuning.

Evaluation Metrics

To evaluate how well our model works, we used the following key metrics:

I. Accuracy

This metric tells us how often the model makes the right prediction. An accuracy score of 80% means that the model is correct 80 out of 100 times.

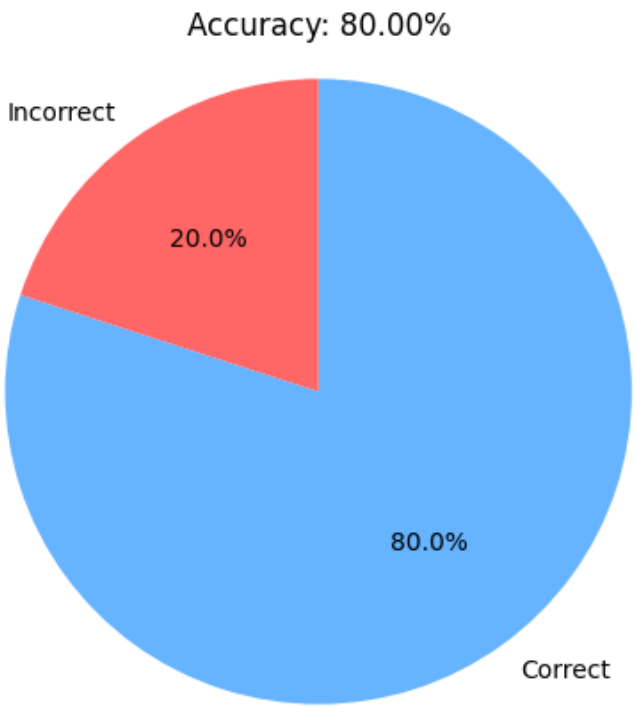

Figure 3: Pie chart showing distribution of correct vs incorrect predictions.

II. Precision

Precision measures the percentage of correctly identified positives out of all the predictions the model made.

III. Recall (Sensitivity)

Recall tells us how good the model is at identifying positive cases. This metric becomes essential when missing a positive case can have serious consequences.

IV. F1 Score

The F1 score is a balanced measure that considers both precision and recall. It provides a single number to summarize performance when both metrics are equally important.

V. AUC-ROC

AUC-ROC is a graphical measure of how well our model distinguishes between the two classes. The closer the value is to 1, the better the model performs.

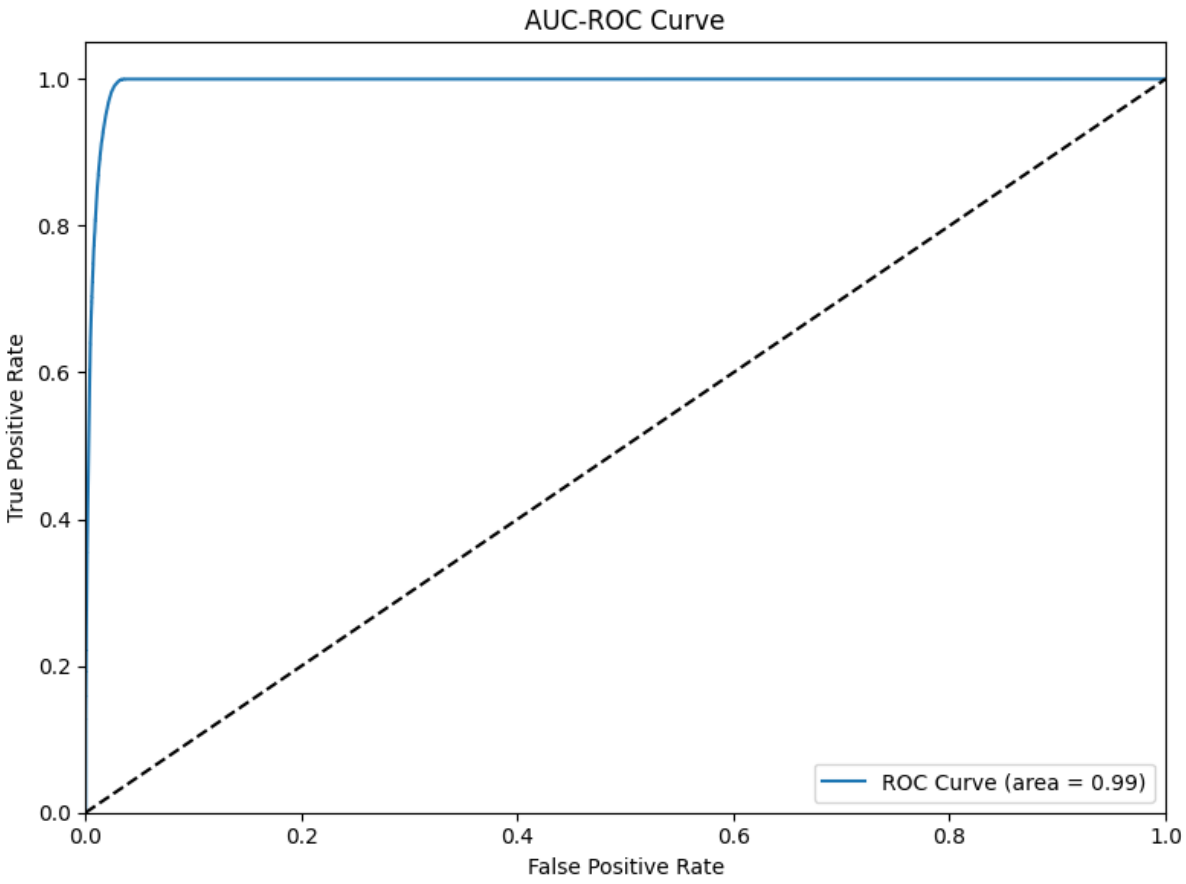

Figure 4: AUC-ROC curve showing trade-off between true positives and false positives.

VI. Confusion Matrix

The confusion matrix provides a detailed breakdown of correct and incorrect predictions. It includes the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

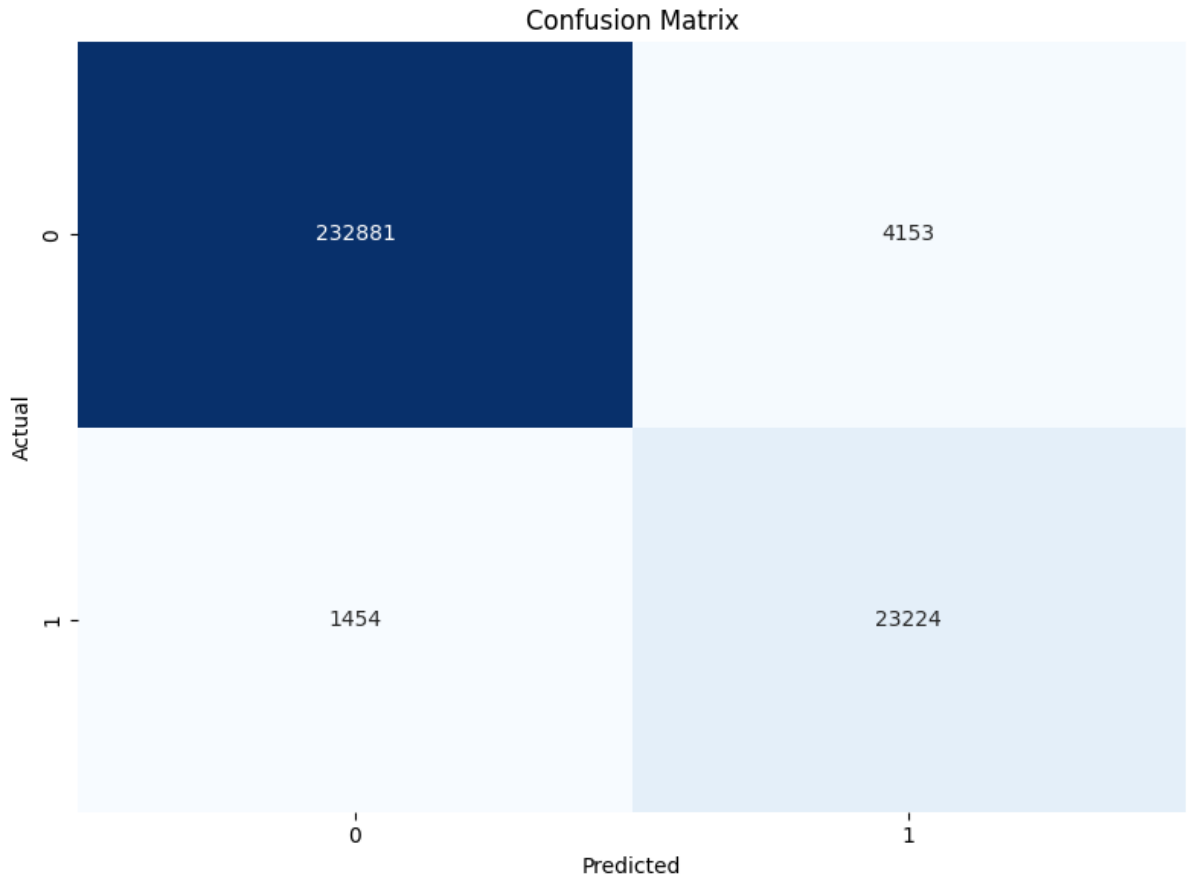

Figure 5: Confusion matrix heatmap showing classification results.

Results and Insights

- **Accuracy**: Our model achieved an accuracy of 97.84% on test data, meaning it correctly predicted the target variable in 97.84 out of 100 cases.
- **Precision**: We achieved a precision score of 0.8496, meaning that 84.96% of the positive predictions were correct.
- **Recall**: With a recall score of 0.9394, our model successfully captured 93.94% of the actual positive cases.
- **F1 Score**: The F1 score of 0.8923 indicates a balanced performance between precision and recall.
- **AUC-ROC**: Our model obtained an AUC score of 0.9949, showcasing excellent performance in distinguishing between the two classes.

- **Confusion Matrix**: The confusion matrix showed that the model made only a few incorrect predictions, with the majority of cases being accurately classified.

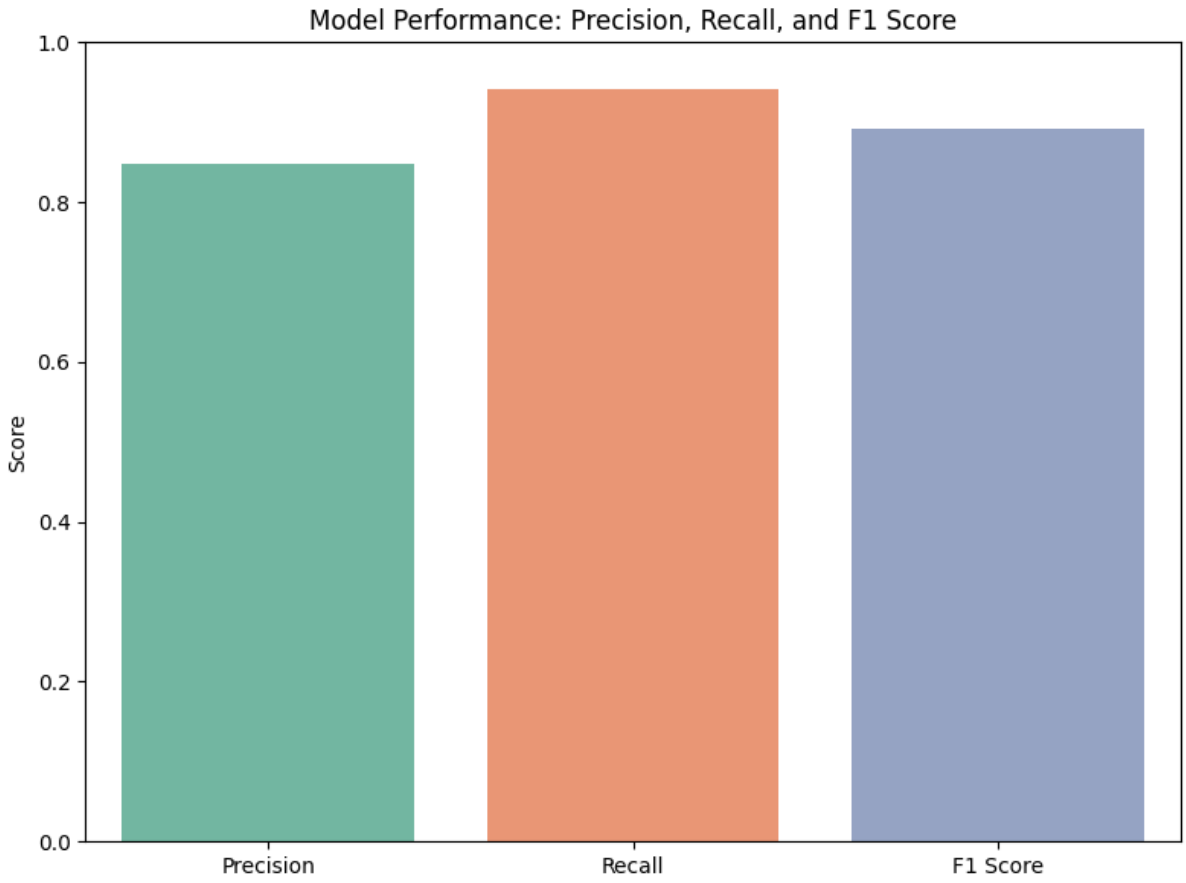

Figure 6: Bar Diagram showing precision, recall, and F1 score.

Impact and Conclusion

Our solution leverages state-of-the-art machine learning techniques to provide a reliable and robust classification model for GST data. By accurately predicting the target variable, the model can assist in improving the efficiency of the GST analytics framework, aiding government agencies in detecting anomalies or patterns that require further investigation.

The use of hyperparameter optimization has further enhanced the model's performance, making it suitable for real-world application where accuracy and reliability are crucial. Our approach demonstrates the potential of AI and ML in enhancing tax data analysis, benefiting the overall GST system.

Citation Report

- **XGBoost Documentation**: XGBoost Documentation (https://xgboost.readthedocs.io/en/latest/)
- **Optuna Documentation**: Optuna Documentation (https://optuna.readthedocs.io/en/stable/)
- **Scikit-learn Documentation**: Scikit-learn Documentation(https://scikit-learn.org/stable/)
- **Pandas Documentation**: Pandas Documentation(https://pandas.pydata.org/docs/)