GSTN Analytics Hackathon 2024

Himanshu Mittal

1. Introduction

The aim of this project is to predict a binary target variable using the training and test datasets provided for the GSTN Analytics Hackathon 2024. The challenge required building a model that generalizes well on unseen data, without relying on specific domain knowledge. A stacked tree-based machine learning algorithm was trained on newly engineered features, yielding an **average precision score of 0.9407** and surpassing the baseline approach.

Table 1.1 Missing column values in the training data

Column	Missing Values	% of Total Values
Column9	732137	93.3
Column14	365703	46.6
Column5	167180	21.3
Column4	127710	16.3
Column3	126303	16.1
Column15	16456	2.1
Column6	3850	0.5
Column8	3850	0.5
Column0	9	0.0

The training dataset contains approximately 785,000 rows, while the validation dataset contains around 261,000 rows, with 23 features provided. Both datasets exhibit class imbalance, with 9.43% of the data belonging to the positive target class. Among the 23 features, 9 had missing values, as shown in Table 1.1. All columns were preprocessed and numerical, except the 'ID' column, which was safely discarded as it served only as a unique identifier.

2. Methodology

2.1 Key Observations

Several insights emerged from the Exploratory Data Analysis (EDA) that were crucial in guiding the feature engineering process:

Observation 1: Missing Values as Predictive Features

Missing values in certain columns were found to be informative. For example, Column 14's missingness correlates with a higher likelihood of a positive target class, as shown in Table 2.1.1.

Table 2.1.1 Row counts for column 14 binned against the target

	Counts of rows in training dataset	
Column 14 Bins	**Target 0**	**Target 1**
X < -0.01	99	2
-0.01 <= X < 0	88	3
0 <= X < 0.01	406172	12975
X >= 0.01	91	0
NA	304650	61053

Observation 2: Relationship Between Columns 3 & 4 and Column 6 & 8

Columns 3 and 4 seem semantically related, as missing values in one often correlate with missing values in the other (fig. 1). Both are categorical with 47 unique values each. They are complementary but not correlated, as seen in the scatter plots and histograms (fig. 2 and 3 respectively). Similar properties were observed for column 6 and column 8.

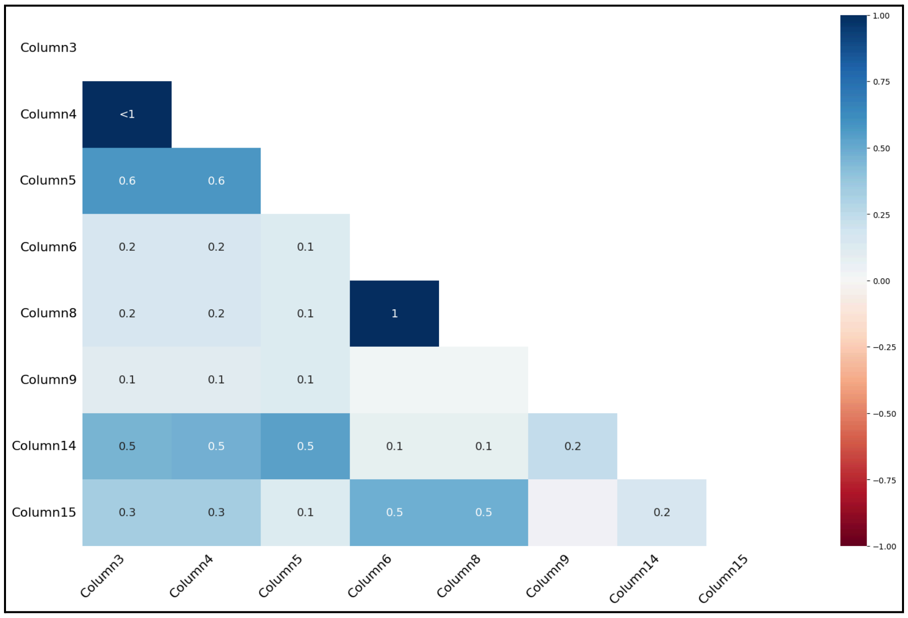


Fig. 1. Correlation heatmap between the columns having null values for missing analysis

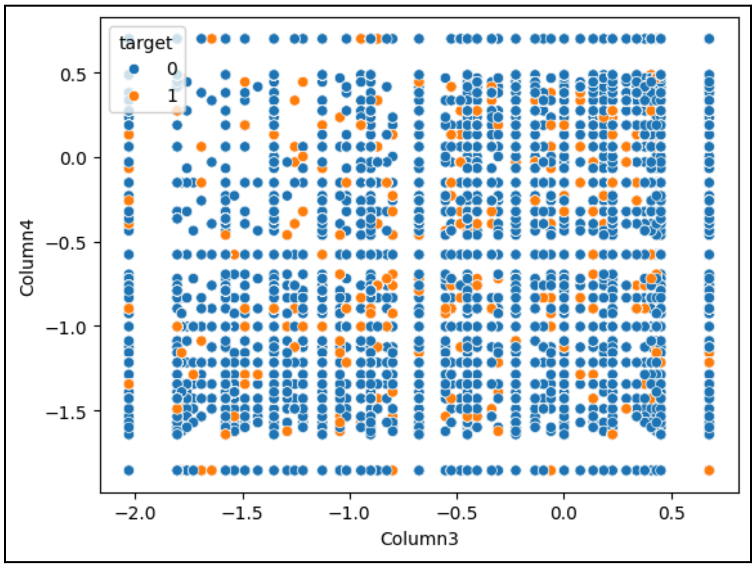


Fig. 2. Scatterplot between column 3 and column 4

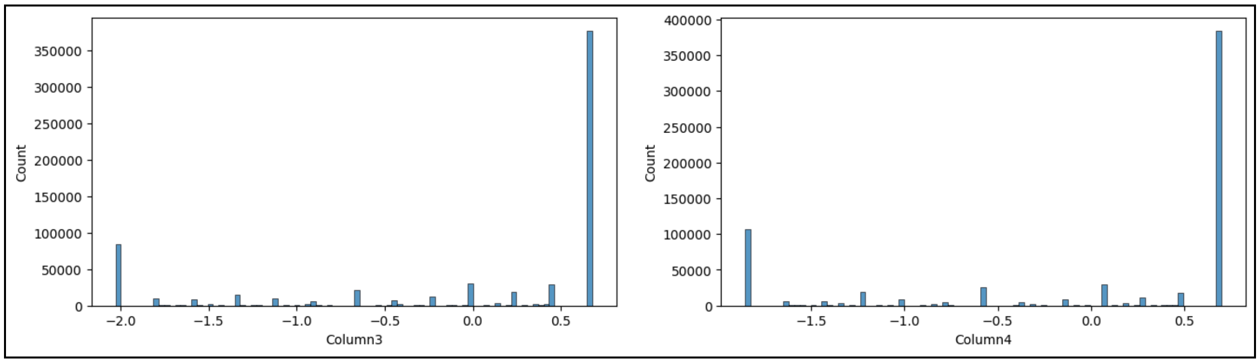


Fig. 3. Histogram of column 3 (left) and column 4 (right)

Observation 3: Outliers in Numerical Columns

Certain numerical columns exhibit unusual distributions, such as Column 6 (fig. 4), where the data is concentrated at specific values and flat at other regions. They don't offer much variation as it is to recognize the patterns. Therefore, outliers were flagged using isolation forest and further studied.

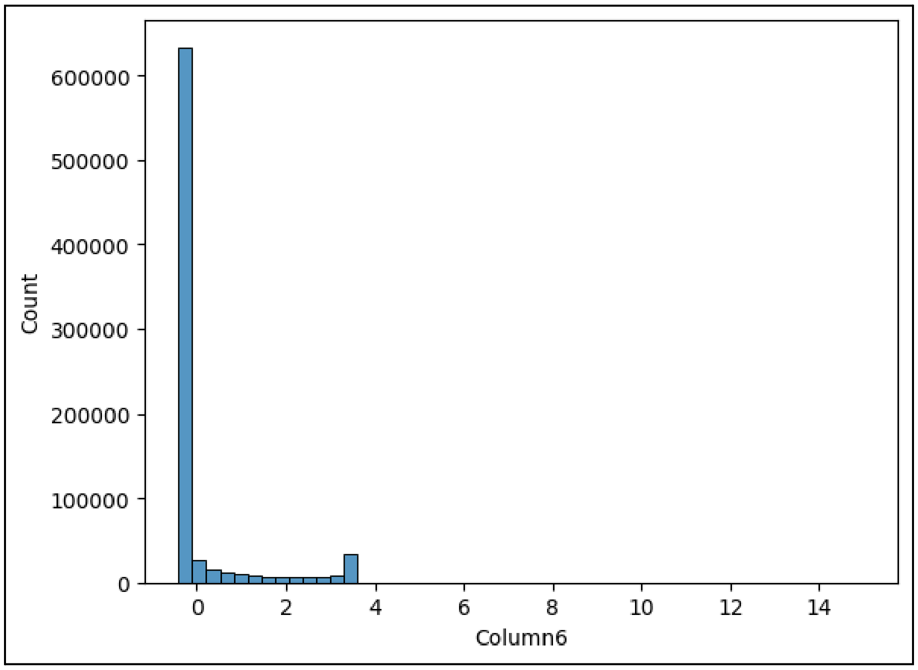


Fig. 4. Histogram of Column 6

Observation 4: Strong Deterministic Features

Specific conditions in certain columns, such as Column 18 being zero or Column 1 equaling 2495, consistently predict the target as zero. This is a statistically significant pattern. Approximately 87.2% of both the training and testing data exhibit this property, meaning no complex model is needed for these rows—they are automatically predicted as zero (the negative class).

This observation is crucial for the modeling approach, as two columns can predict 87.2% of the data with 100% accuracy. This insight led to the decision to adopt a stacked approach, varying models based on row selection. We can refer to rows with this feature as 'masked data,' while the rest are 'unmasked data.' As a result, one model can be trained on the full dataset, and another on just the unmasked data.

2.2 Feature Engineering

To enhance the predictive power of the model, the following steps were applied during the feature engineering process:

1. **Imputation of Missing Values**:
    - **Column 0**: Imputed using the most frequent value since it had very few missing entries.
    - **Column 9**: Imputed using the median, which was more robust against the extreme values in this column.
    - **Remaining Columns**: Missing values were imputed using a custom formula that took the average of the median values from both the positive and negative target classes:
      Imputed Value = (Positive Median + Negative Median) / 2
      This method preserved any potential relationships between the feature and the target class.
2. **Column Transformations**: Several transformations were applied to create new features:
    - The difference between **Column 3** and **Column 4** was computed to capture any relationship between these two semantically related columns.

    - A new feature was created by multiplying **Column 6** and **Column 8**, which were found to have correlated patterns.
    - Values in **Column 1** were transformed to remove negative values, ensuring that the data followed a consistent range.
3. **Outlier Detection**:
    - An **Isolation Forest** algorithm was employed to detect outliers in the dataset. These outliers were then flagged, and new indicator features were introduced to highlight rows with potential anomalies. This helped the model learn from the typical data patterns while treating outliers separately.
4. **Feature Selection**:
    - After training the primary models, features that consistently ranked in the bottom five in more than 50% of the models were removed. This ensured that only the most relevant features were retained for the final model. As a result, **Column 9** and several other indicator features were dropped from the dataset.

Advanced encoding methods, such as CatBoost encoding and iterative imputation, were also tested, but they did not lead to any improvements and were ultimately discarded.

Categorical binning and one-hot encoding were explored but did not show improved performance, so these techniques were excluded from the final model.

2.3 Model Training

A stacked ensemble classifier was constructed using 10 tree-based models, each trained on different hyperparameters and row selections (see fig. 5). The ensemble comprised:

- 2 Random Forest models
- 3 XGBoost models
- 3 CatBoost models
- 2 LightGBM models

Each model contributed to the final predictions in a weighted ensemble.

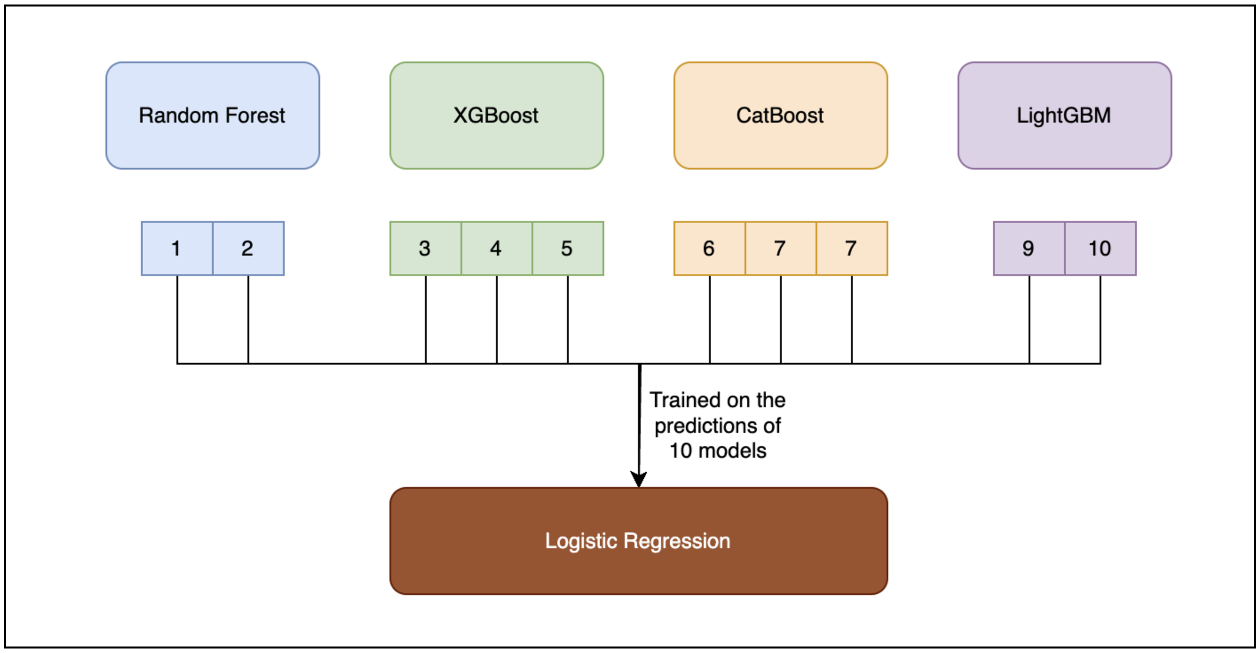


Fig. 5. Model Architecture

Here's a brief description of the five models used in the project:

1. **Random Forest**
   Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. It works by averaging the predictions of individual trees, each trained on a random subset of data and features, making it robust to noisy data and feature correlations.
2. **XGBoost**
   XGBoost (Extreme Gradient Boosting) is a powerful gradient-boosting algorithm known for its efficiency and performance in structured data tasks. It builds models in stages, where each new model attempts to correct the errors of the previous one. XGBoost includes regularization to prevent overfitting, making it highly effective in competition settings.
3. **CatBoost**
   CatBoost is another gradient-boosting algorithm, optimized specifically for handling categorical features. It uses a combination of ordered boosting and efficient encoding of

categorical variables to provide fast training and high accuracy, especially in datasets with mixed data types.

4. **LightGBM**
   LightGBM (Light Gradient Boosting Machine) is designed for speed and scalability. It uses a leaf-wise growth strategy, which allows it to build deep trees and find better splits. LightGBM is especially efficient for large datasets and high-dimensional data, with minimal memory consumption.
5. **Logistic Regression**
   Logistic Regression is a simple, linear model for binary classification. It uses the sigmoid function to model probabilities, making it interpretable and fast to train. Cross-validation was used to optimize regularization strength.

These models, when combined, offered diverse perspectives, improving performance in the stacked classifier.

3. Results and Discussions

For this project, performance evaluation was conducted using several key metrics, with a focus on **PR-AUC** (Precision-Recall Area Under Curve) due to the highly imbalanced nature of the dataset.

The following metrics were calculated to assess the model's performance across both the positive and negative classes.

- **PR-AUC (0.9407)**:
  The PR-AUC score was chosen as the primary evaluation metric. It measures the trade-off between precision (how many of the predicted positives are correct) and recall (how many actual positives are identified. A score of **0.9407** indicates strong performance in identifying positive instances.
- **Accuracy (97.74%)**:
  Accuracy reflects the overall percentage of correct predictions. While the model achieved a high accuracy of **97.74%**, this metric can be misleading for imbalanced datasets, where

the majority class dominates. Consider a case where all zeros are predicted, accuracy would be 90.5% but the PR-AUC score would be 9.4% only.

- **ROC-AUC (0.9950)**:
  The high score of **0.9950** ROC-AUC measures how well the model separates the target class across different thresholds, though also misleading in imbalanced data.
- **Precision, Recall, and F1-Score**:
  These metrics were calculated for each class and then summarized as macro and weighted averages:
  - The **macro average** provides an equal-weighted view of the model's performance across both classes. This is useful for understanding how the model performs on the minority class (positive target).
  - The **weighted average** accounts for the class imbalance by giving more weight to the majority class (negative target). This provides a more balanced assessment of the model's overall effectiveness.

Table 3.1. Precision, Recall and F1 with macro and average variation

Metric	Macro Avg	Weighted Avg
Precision	**0.935594**	0.977301
Recall	0.931509	0.977403
F1-Score	0.933539	0.977349

The average metrics shown in Table 3.1 indicate that while the model performs well on both classes, it excels more on the majority class, as reflected in the weighted averages.

Table 3.2 Classification Report

Class	Precision	Recall	F1-Score
0	0.986995	0.988069	0.987532
1	0.884193	0.874949	0.879547

The model demonstrates strong performance in predicting both classes, although it performs slightly better for class 0 (negative class) due to the class imbalance. The relatively lower precision and recall for class 1 (positive class) reflect the challenge of predicting the minority class accurately, but overall, the model balances the trade-off well, as shown by the F1-Score of 0.8795 for class 1.

To validate the improvements made through feature engineering and model stacking, the proposed model was compared with simpler baselines:

Table 3.3. Comparison of Models

Model	**Average Precision Score**
All 0 predictions	0.0942
Random (unmasked data, 0 otherwise)	0.4210
Random Forest (baseline)	0.9281
Proposed stack	**0.9407**

The proposed stacked classifier outperformed simpler baselines, including a random forest model, showing a significant improvement in precision and recall for the minority class.

4. Conclusion

This project successfully developed a robust machine learning model to predict a binary target variable in an imbalanced dataset. Using advanced feature engineering and a stacked ensemble of tree-based models, the model achieved a PR-AUC score of 0.9407, significantly outperforming baseline models.

While the model performs well, especially for the minority class, further improvements could be made with domain-specific knowledge. Understanding the dataset's context would allow for more meaningful feature engineering and preprocessing strategies. Additionally, techniques such as synthetic data generation could improve balance and further enhance the model's performance.

In summary, stacking tree-based models with thoughtful feature engineering can yield powerful models even in the absence of deep domain knowledge. Future work could involve more sophisticated data augmentation and deeper feature transformations to push performance further.

5. Appendix

5.1 Hyperparameter Tuning

Bayesian search via the Optuna library was used for 100 iterations with a 600-second timeout for each primary model. Tables 5.1.1, 5.1.2, 5.1.3, and 5.1.4 describe the configuration and hyperparameter settings for each primary model trained.

Table 5.1.1 Random Forest (RF)

Configuration	**rf1**	**rf2**
Sampling	No	Yes
n_estimators	200	400
max_depth	25	25
min_samples_split	2	9
min_samples_leaf	3	2

Table 5.1.2. CatBoost (catb)

Configuration	**catb1**	**catb2**	**catb3**
Sampling	No	Yes	Yes
iterations	700	700	867
learning_rate	0.05	0.05	0.0259
depth	8	8	10
l2_leaf_reg	0.03	0.03	17.4682
random_strength	4e-5	4e-5	0.0208
bagging_temperature	0.45	0.45	0.0581

Table 5.1.3. LightGBM (LGBM)

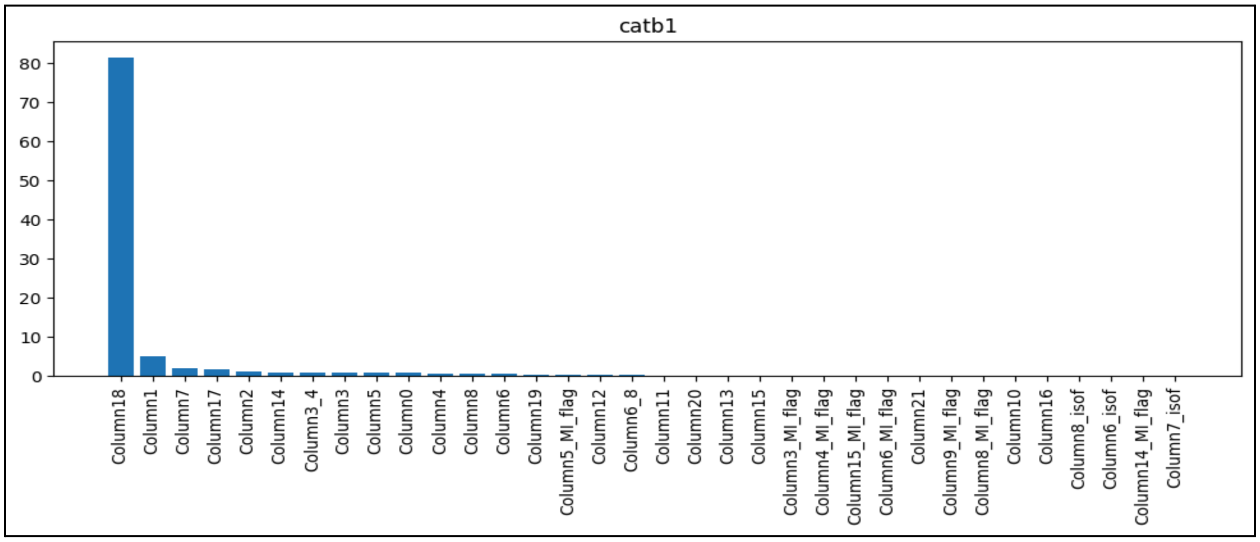
Configuration	lgbm1	lgbm2
Sampling	No	Yes
learning_rate	0.1587	0.1587
num_leaves	2920	2920
max_depth	11	11
min_data_in_leaf	700	700
lambda_l1	0	0
lambda_l2	25	25
min_gain_to_split	0.0932	0.0932
bagging_fraction	0.9	0.9
bagging_freq	1	1
feature_fraction	0.8	0.8

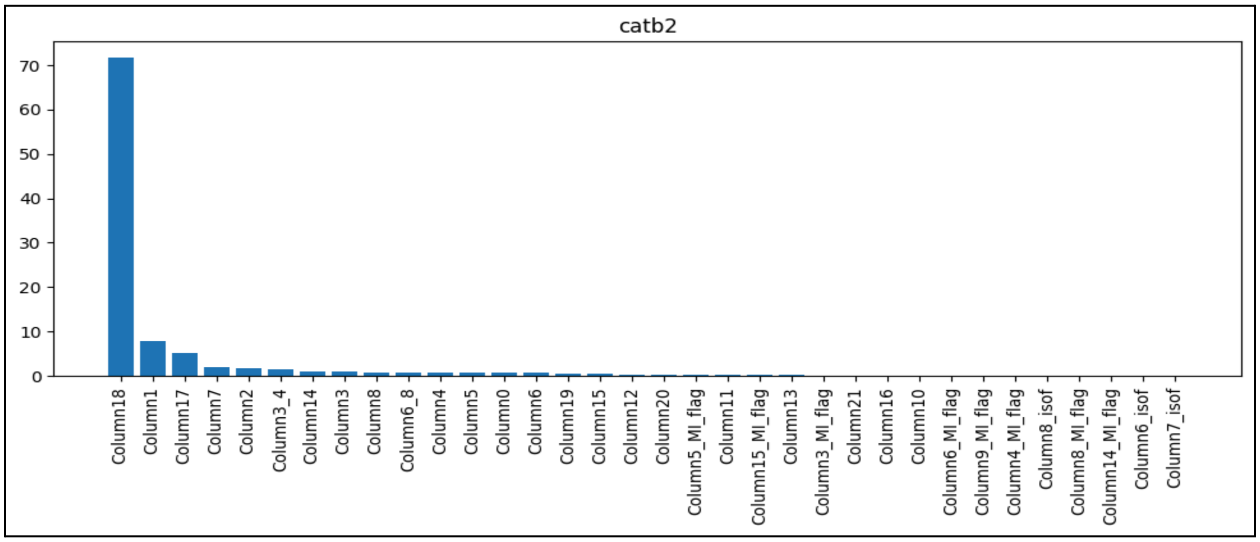
Table 5.1.4. XGBoost (XGB)

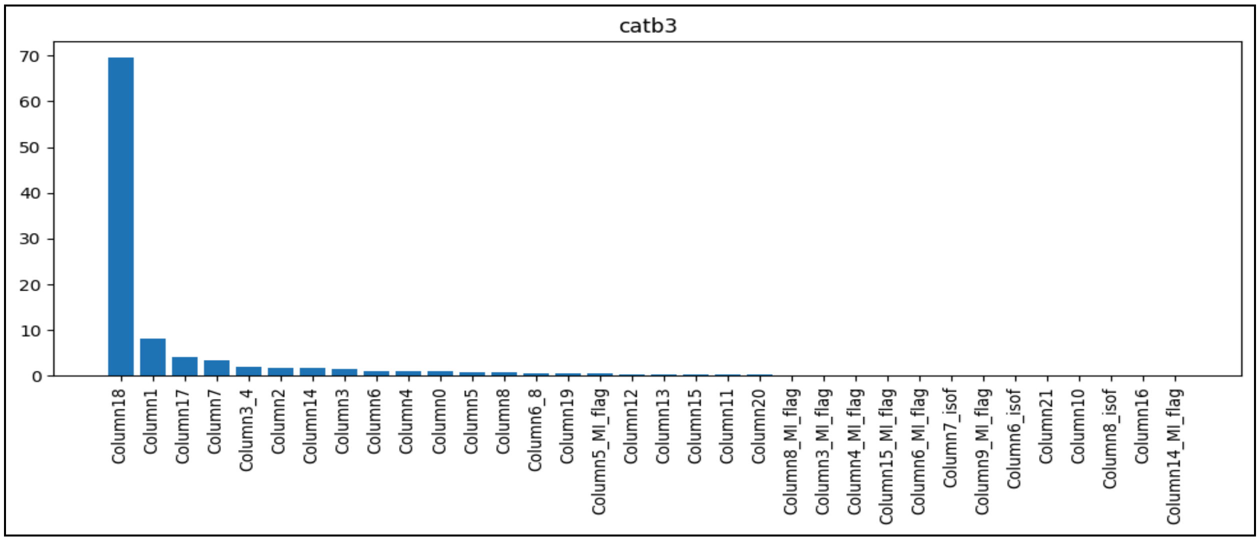
Configuration	xgb1	xgb2	xgb3
Sampling	No	Yes	Yes
lambda	0.000848	0.0484	3.5253e-07
alpha	1.6986e-05	1.6183e-08	3.0917e-06
subsample	0.6267	0.9503	0.7283
colsample_bytree	0.8716	0.9849	0.8566
max_depth	9	9	9
min_child_weight	3	3	3
eta	1.4252e-05	0.2374	0.2944
gamma	0.000573	7.5314e-06	0.2062

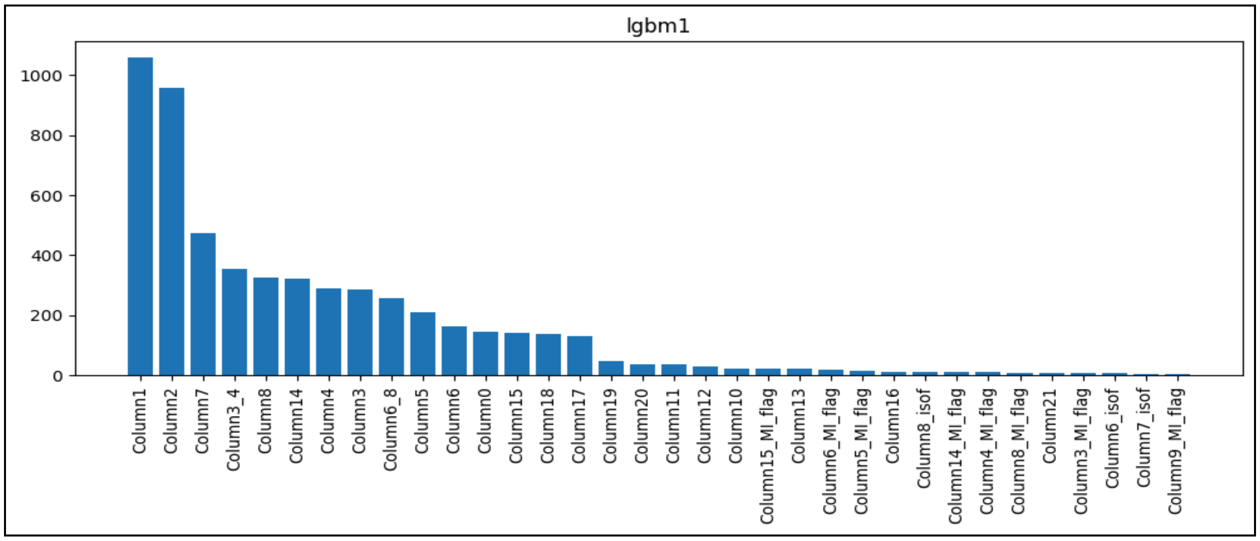
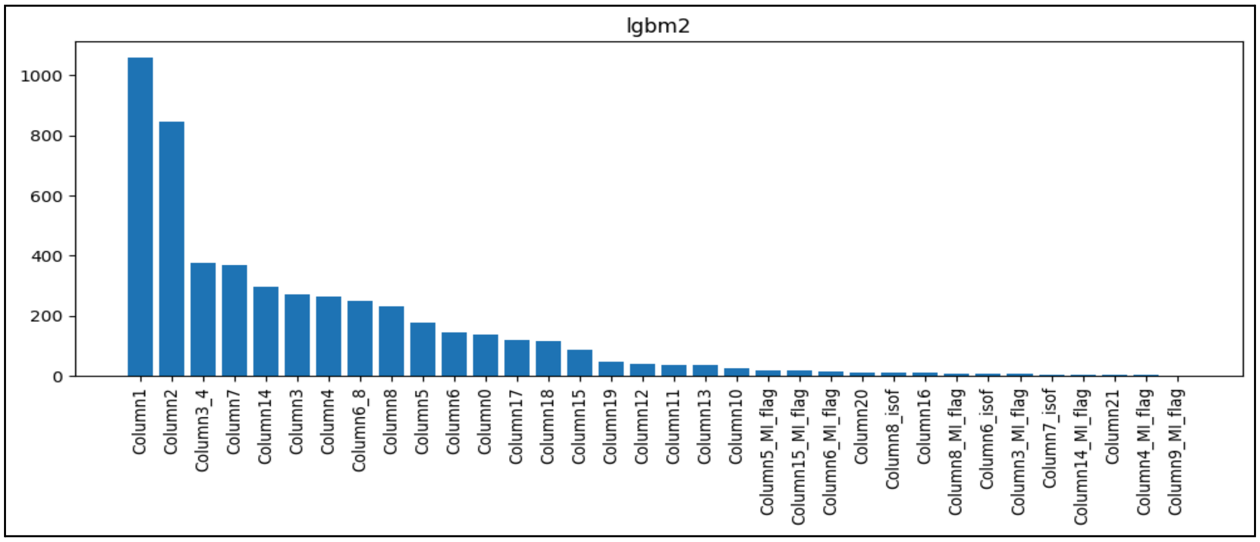
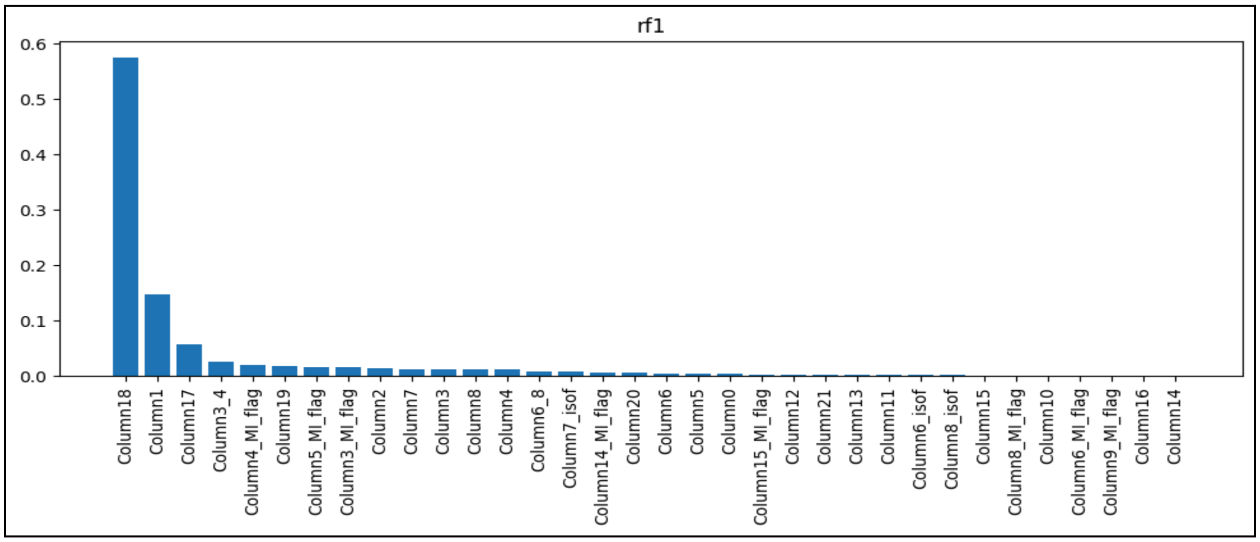
5.2 Feature Importance

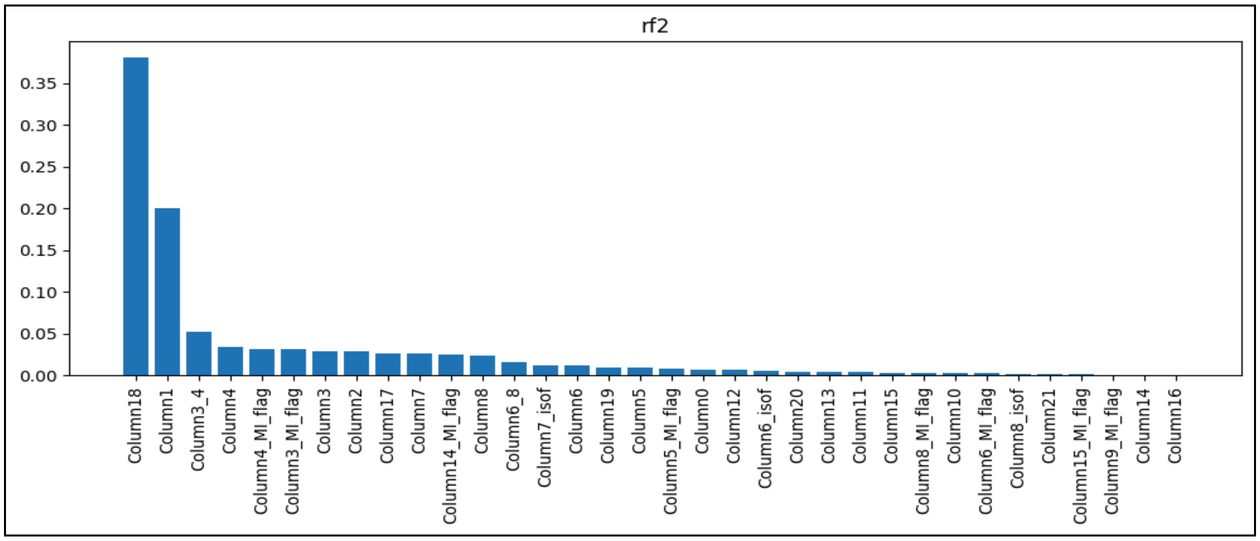
After training the ensemble, less important features are removed from the final model. The bar charts below, labeled by model name, display the feature importance results from each model.

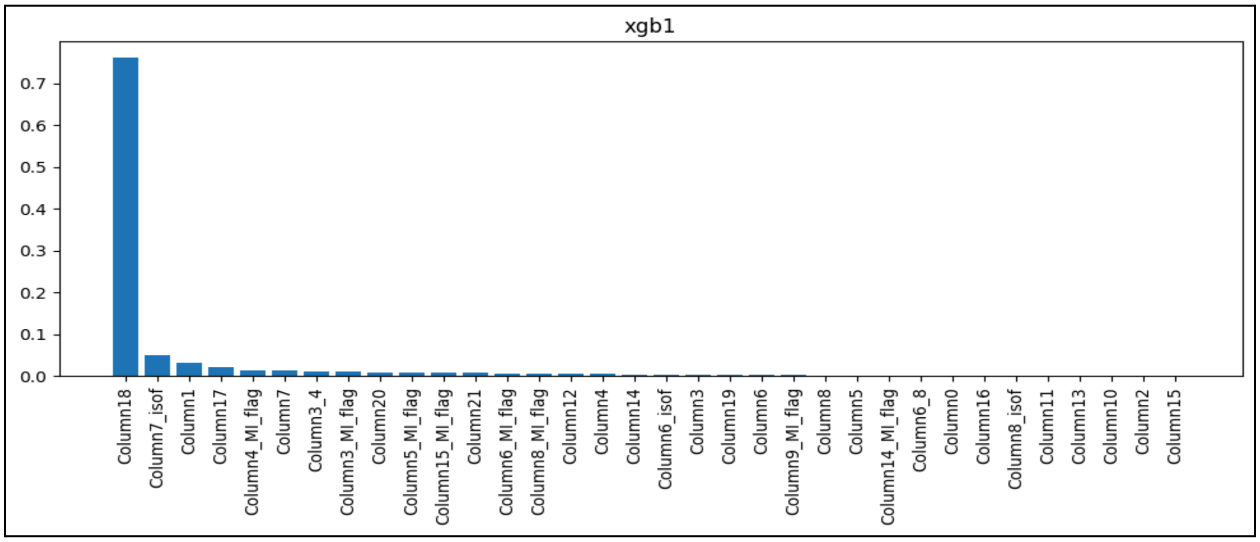

catb1
0
10
20
30
40
50
60
70
80
Column18
Column1
Column7
Column17
Column2
Column14
Column3_4
Column3
Column5
Column0
Column4
Column8
Column6
Column19
Column5_MI_flag
Column12
Column6_8
Column11
Column20
Column13
Column15
Column3_MI_flag
Column4_MI_flag
Column15_MI_flag
Column6_MI_flag
Column21
Column9_MI_flag
Column8_MI_flag
Column10
Column16
Column8_isof
Column6_isof
Column14_MI_flag
Column7_isof

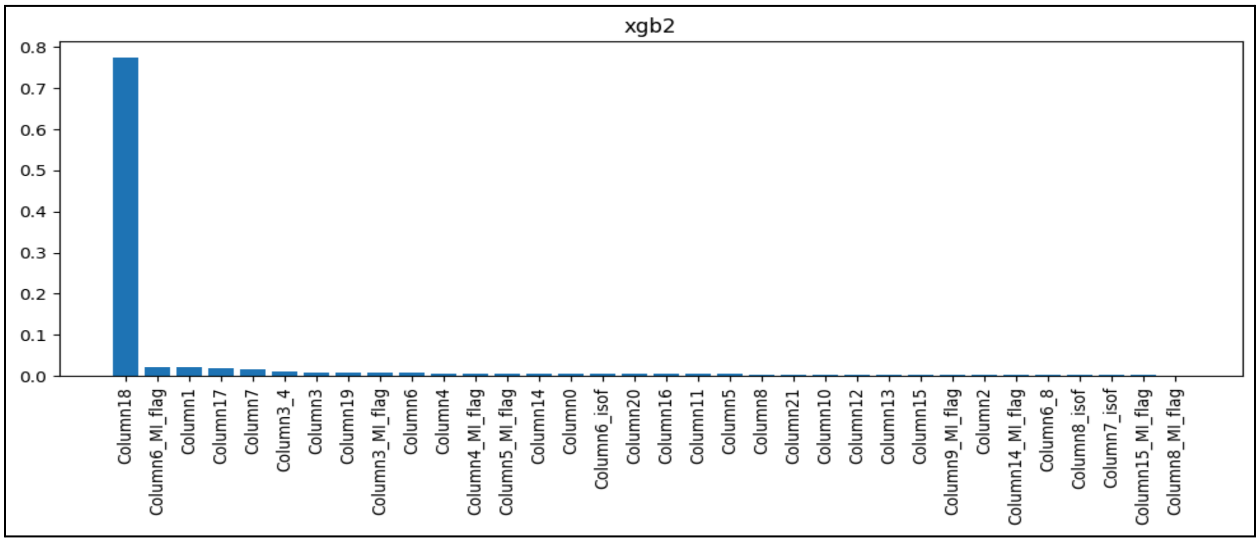

catb2
0
10
20
30
40
50
60
70
Column18
Column1
Column17
Column7
Column2
Column3_4
Column14
Column3
Column8
Column6_8
Column4
Column5
Column0
Column6
Column19
Column15
Column12
Column20
Column5_MI_flag
Column11
Column15_MI_flag
Column13
Column3_MI_flag
Column21
Column16
Column10
Column6_MI_flag
Column9_MI_flag
Column4_MI_flag
Column8_isof
Column8_MI_flag
Column14_MI_flag
Column6_isof
Column7_isof

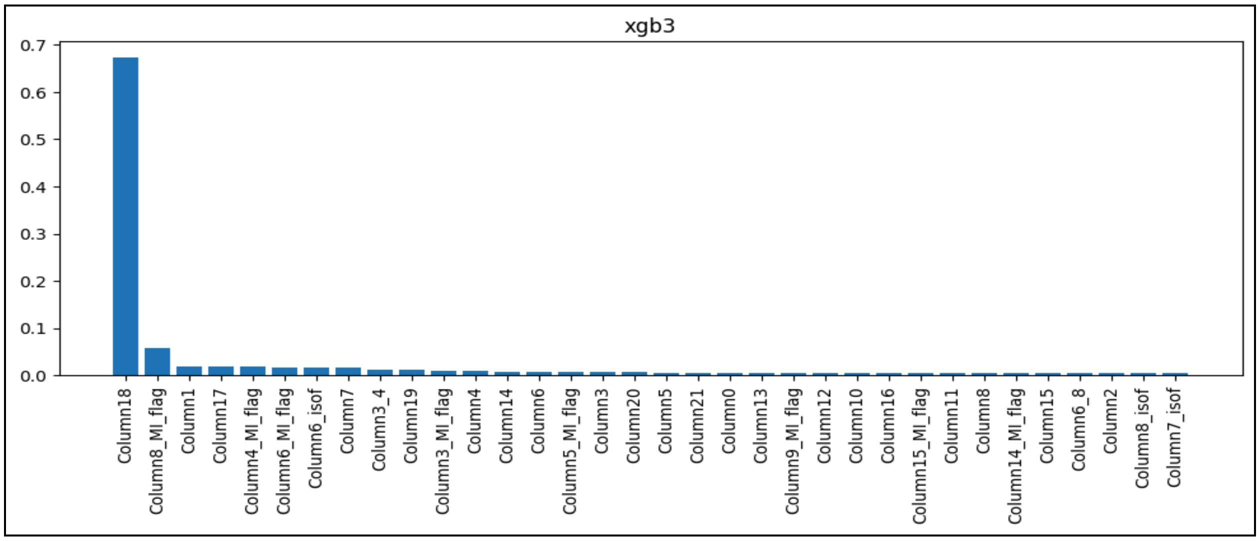

catb3
0
10
20
30
40
50
60
70
Column18
Column1
Column17
Column7
Column3_4
Column2
Column14
Column3
Column6
Column4
Column0
Column5
Column8
Column6_8
Column19
Column5_MI_flag
Column12
Column13
Column15
Column11
Column20
Column8_MI_flag
Column3_MI_flag
Column4_MI_flag
Column15_MI_flag
Column6_MI_flag
Column7_isof
Column9_MI_flag
Column6_isof
Column21
Column10
Column8_isof
Column16
Column14_MI_flag


lgbm1
0
200
400
600
800
1000
Column1
Column2
Column7
Column3_4
Column8
Column14
Column4
Column3
Column6_8
Column5
Column6
Column0
Column15
Column18
Column17
Column19
Column20
Column11
Column12
Column10
Column15_MI_flag
Column13
Column6_MI_flag
Column5_MI_flag
Column16
Column8_isof
Column14_MI_flag
Column4_MI_flag
Column8_MI_flag
Column21
Column3_MI_flag
Column6_isof
Column7_isof
Column9_MI_flag

lgbm2
0
200
400
600
800
1000
Column1
Column2
Column3_4
Column7
Column14
Column3
Column4
Column6_8
Column8
Column5
Column6
Column0
Column17
Column18
Column15
Column19
Column12
Column11
Column13
Column10
Column5_MI_flag
Column15_MI_flag
Column6_MI_flag
Column20
Column8_isof
Column16
Column8_MI_flag
Column6_isof
Column3_MI_flag
Column7_isof
Column14_MI_flag
Column21
Column4_MI_flag
Column9_MI_flag

rf1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Column18
Column1
Column17
Column3_4
Column4_MI_flag
Column19
Column5_MI_flag
Column3_MI_flag
Column2
Column7
Column3
Column8
Column4
Column6_8
Column7_isof
Column14_MI_flag
Column20
Column6
Column5
Column0
Column15_MI_flag
Column12
Column21
Column13
Column11
Column6_isof
Column8_isof
Column15
Column8_MI_flag
Column10
Column6_MI_flag
Column9_MI_flag
Column16
Column14


rf2
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Column18
Column1
Column3_4
Column4
Column4_MI_flag
Column3_MI_flag
Column3
Column2
Column17
Column7
Column14_MI_flag
Column8
Column6_8
Column7_isof
Column6
Column19
Column5
Column5_MI_flag
Column0
Column12
Column6_isof
Column20
Column13
Column11
Column15
Column8_MI_flag
Column10
Column6_MI_flag
Column8_isof
Column21
Column15_MI_flag
Column9_MI_flag
Column14
Column16


xgb1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Column18
Column7_isof
Column1
Column17
Column4_MI_flag
Column7
Column3_4
Column3_MI_flag
Column20
Column5_MI_flag
Column15_MI_flag
Column21
Column6_MI_flag
Column8_MI_flag
Column12
Column4
Column14
Column6_isof
Column3
Column19
Column6
Column9_MI_flag
Column8
Column5
Column14_MI_flag
Column6_8
Column0
Column16
Column8_isof
Column11
Column13
Column10
Column2
Column15


xgb2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Column18
Column6_MI_flag
Column1
Column17
Column7
Column3_4
Column3
Column19
Column3_MI_flag
Column6
Column4
Column4_MI_flag
Column5_MI_flag
Column14
Column0
Column6_isof
Column20
Column16
Column11
Column5
Column8
Column21
Column10
Column12
Column13
Column15
Column9_MI_flag
Column2
Column14_MI_flag
Column6_8
Column8_isof
Column7_isof
Column15_MI_flag
Column8_MI_flag



6. References

1. GSTN Analytics Hackathon 2024 Dataset. Provided by GSTN Analytics. https://event.data.gov.in/event/gst-analytics-hackathon/
2. Chen, T. & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). https://doi.org/10.1145/2939672.2939785
3. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., & Gulin, A. (2018). *CatBoost: unbiased boosting with categorical features*. In Advances in Neural Information Processing Systems, 31. https://arxiv.org/abs/1706.09516
4. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.Y. (2017). *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. In Advances in Neural Information Processing Systems 30 (NIPS 2017), 3146-3154. https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
5. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2623-2631. https://doi.org/10.1145/3292500.3330701