Credit Card Fraud Detection

ML for Fraud Detection and Business Impact Optimization

Spenderlytics

- Our software company specializes in:
  - Analyzing CC transaction data
  - Provide actionable insights
  - Accurate Forecasting
  - Intelligent dashboards
  - **New Service: ML Modeling to detect Fraud**

Turning Transactions into Trust with Insights, Forecasts, and Fraud Defense!

Founding Team

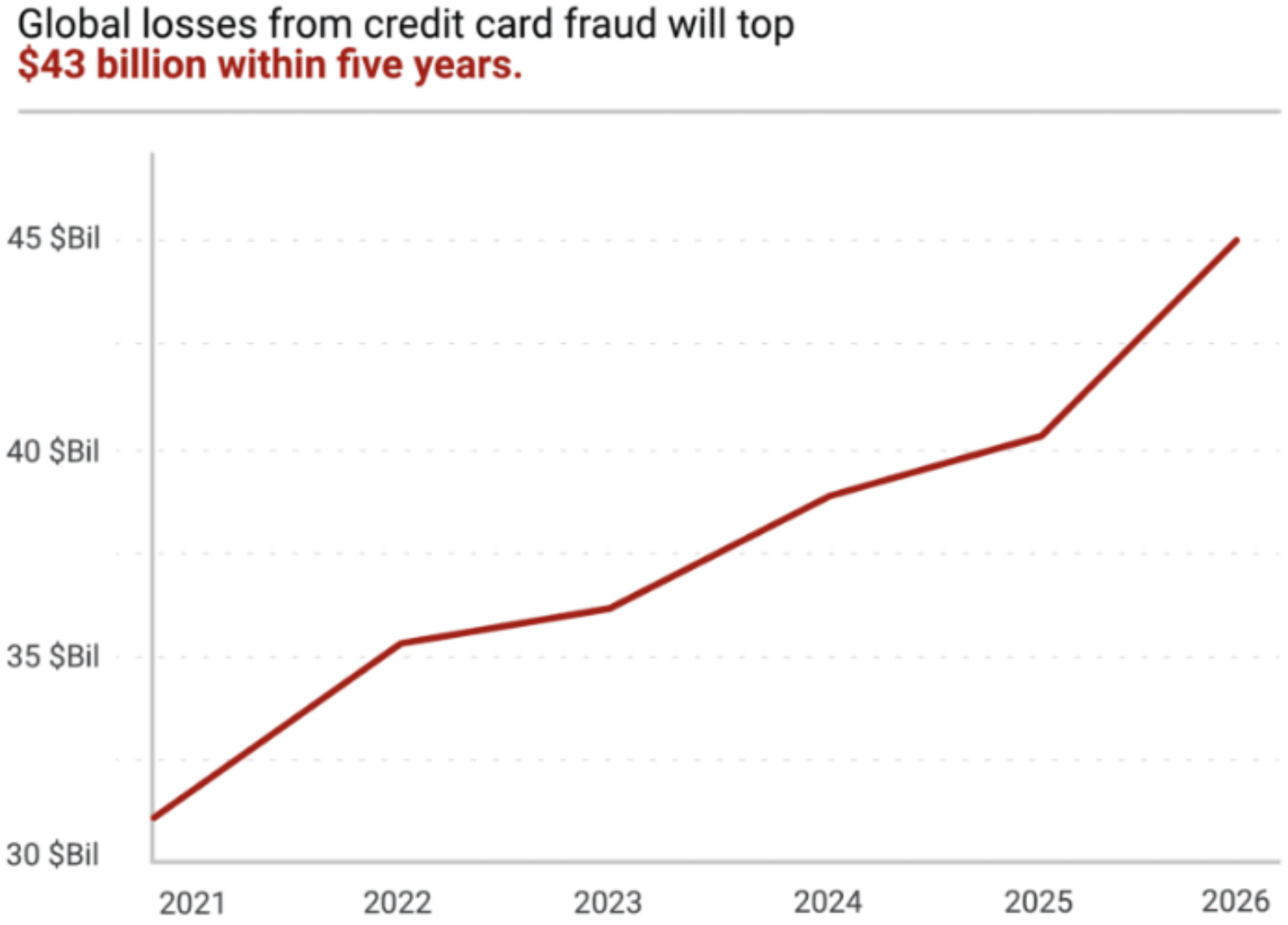
Asif Khan

Jason Brooks

Amit Gaikwad

Kade Thomas

Simranpreet Saini



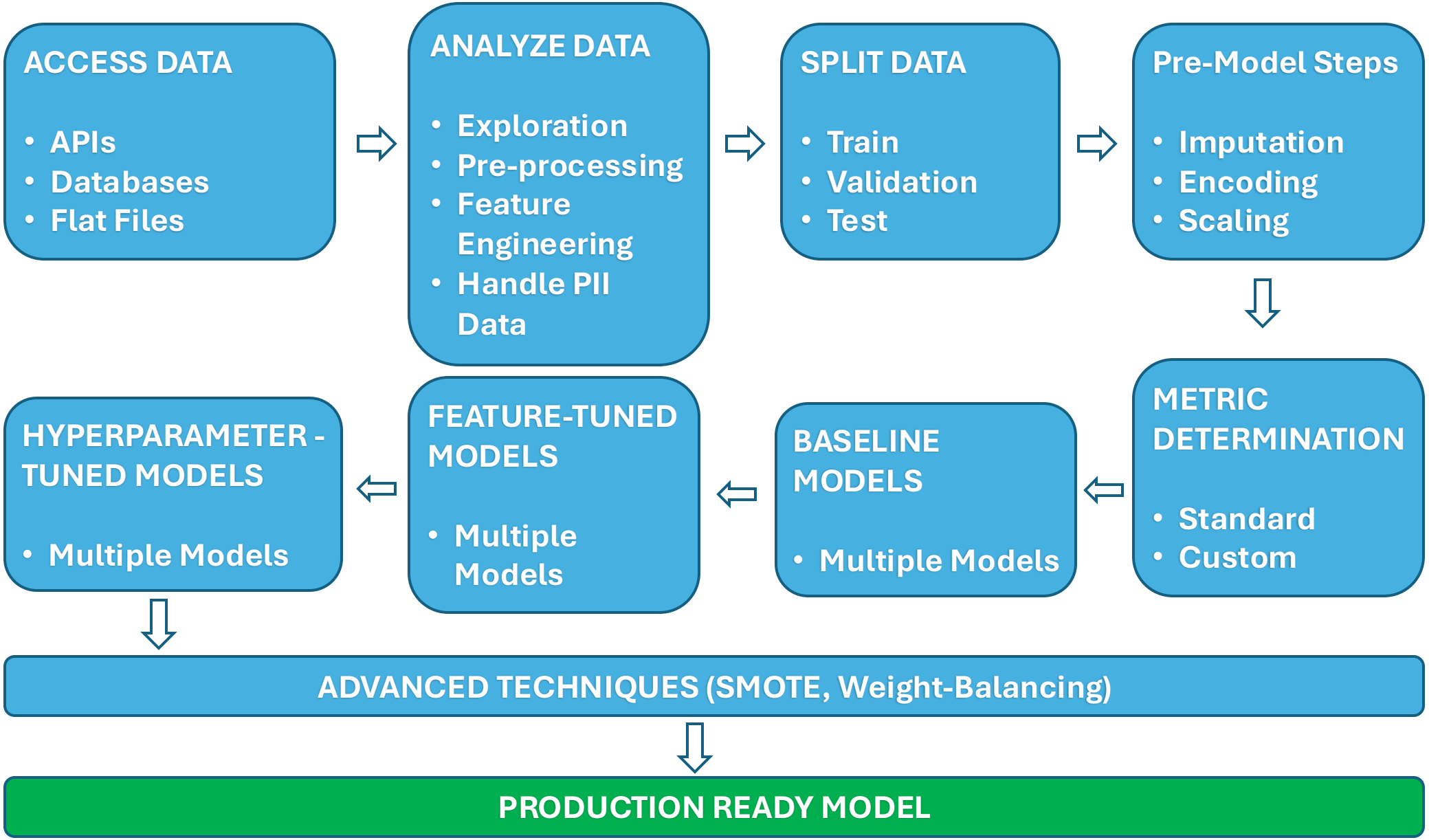
Problem Statement

- CC fraud causes global losses exceeding $32B annually
- Frauds make an extremely imbalanced dataset
- Traditional methods struggle with speed, accuracy and scalability

IMPACT

- Financial Loss
- Customer Dissatisfaction

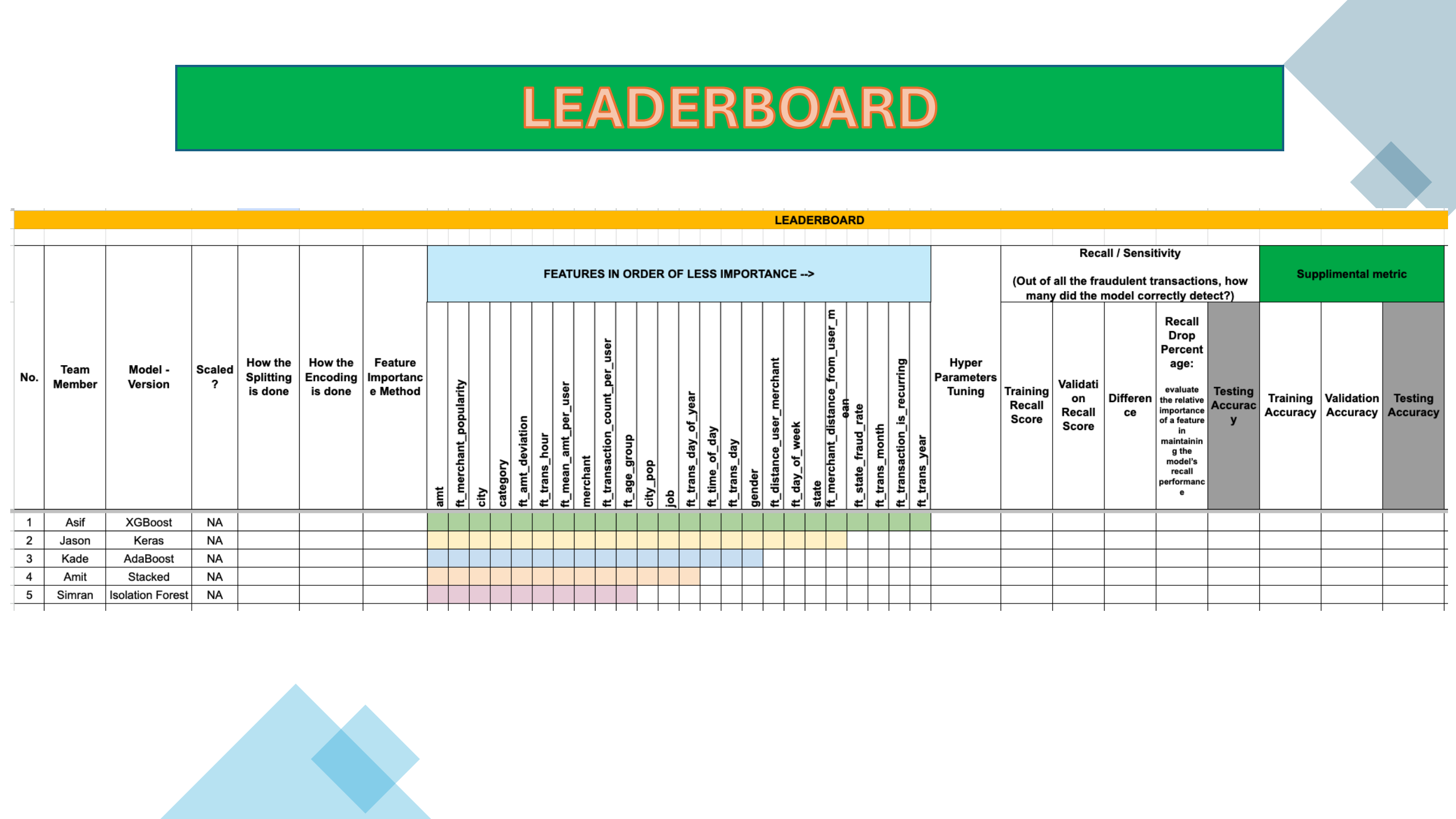
Our Approach



LEADERBOARD

LEADERBOARD

No.	Team Member	Model - Version	Scaled ?	How the Splitting is done	How the Encoding is done	Feature Importance Method	FEATURES IN ORDER OF LESS IMPORTANCE -->																								Hyper Parameters Tuning	Recall / Sensitivity (Out of all the fraudulent transactions, how many did the model correctly detect?)					Supplimental metric		
							amt	ft_merchant_popularity	city	category	ft_amt_deviation	ft_trans_hour	ft_mean_amt_per_user	merchant	ft_transaction_count_per_user	ft_age_group	city_pop	job	ft_trans_day_of_year	ft_time_of_day	ft_trans_day	gender	ft_distance_user_merchant	ft_day_of_week	state	ft_merchant_distance_from_user_mean	ft_state_fraud_rate	ft_trans_month	ft_transaction_is_recurring	ft_trans_year		Training Recall Score	Validation Recall Score	Difference	Recall Drop Percentage: evaluate the relative importance of a feature in maintaining the model's recall performance	Testing Accuracy	Training Accuracy	Validation Accuracy	Testing Accuracy
1	Asif	XGBoost	NA																																			
2	Jason	Keras	NA																																			
3	Kade	AdaBoost	NA																																			
4	Amit	Stacked	NA																																			
5	Simran	Isolation Forest	NA																																			



- **Data Source**:
  - **Kaggle** (Credit Card Transactions Dataset)

- **Data Columns**:
  - Date: Transaction date & Time
  - Credit Card Number
  - Amount: Transaction amount.
  - Merchant
  - Category: Spending category (e.g., Groceries, Dining, Travel, etc.)
  - Geo Information: State/City/Population
  - User Info: Gender/Job/Age

- **Data Size:**
  - 1.5 years
  - 983 Unique Credit Cards
  - 1.29M records
  - Highly imbalanced only .5% Frauds!

Comprehensive Modeling

Conducted **170 iterations** leveraging different variables and advanced techniques

Enabled precise evaluation and comparison of models to select the best-performing approach for both fraud detection and cost optimization.

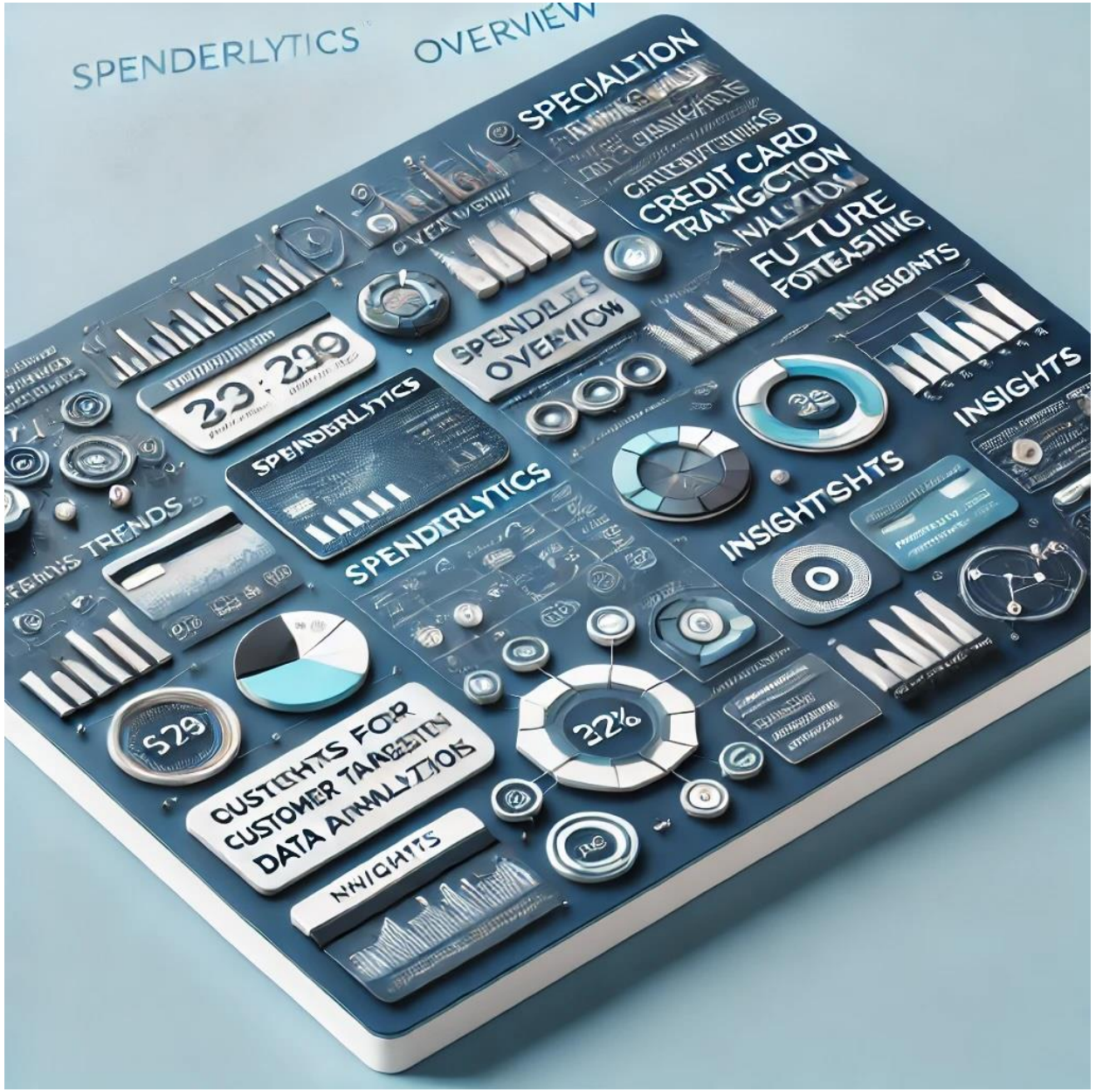
XGBClassifier

XGBClassifier with XGBoost Feature Ranking

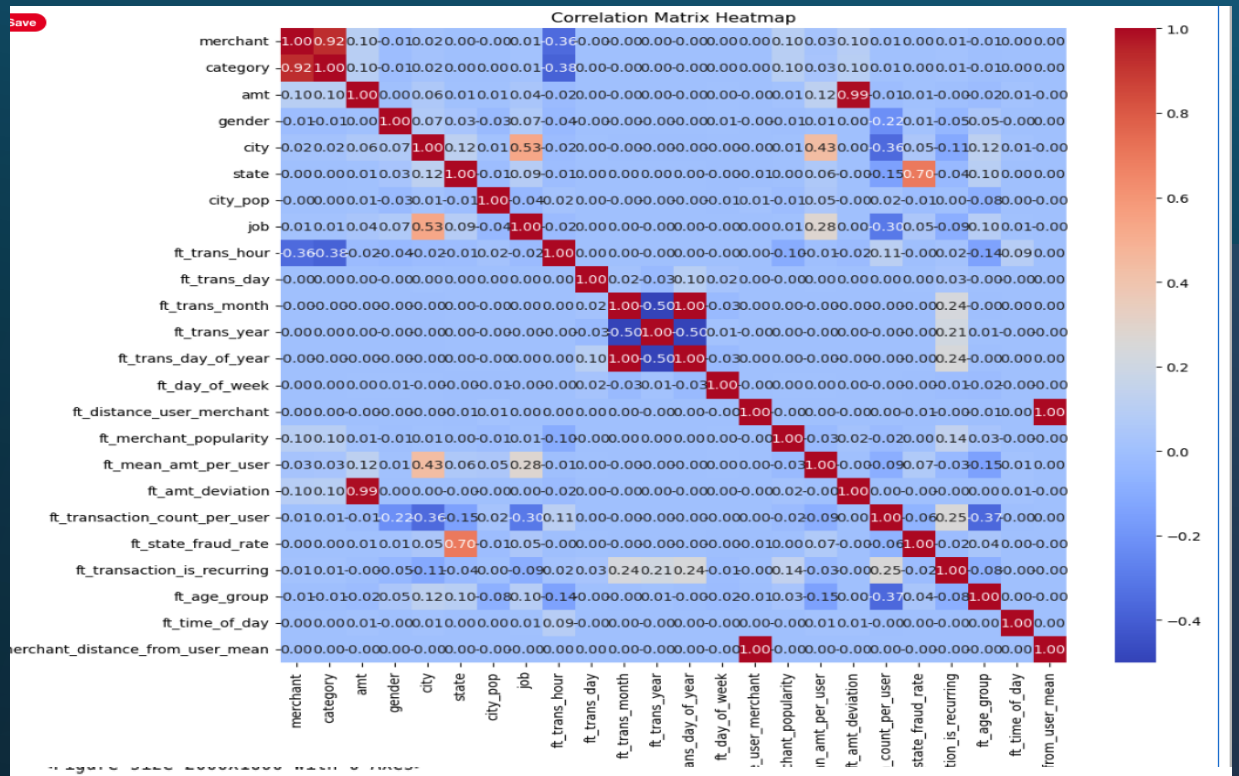
Stacked – XGBClassifier, BalancedRandomForest, LogisticRegression

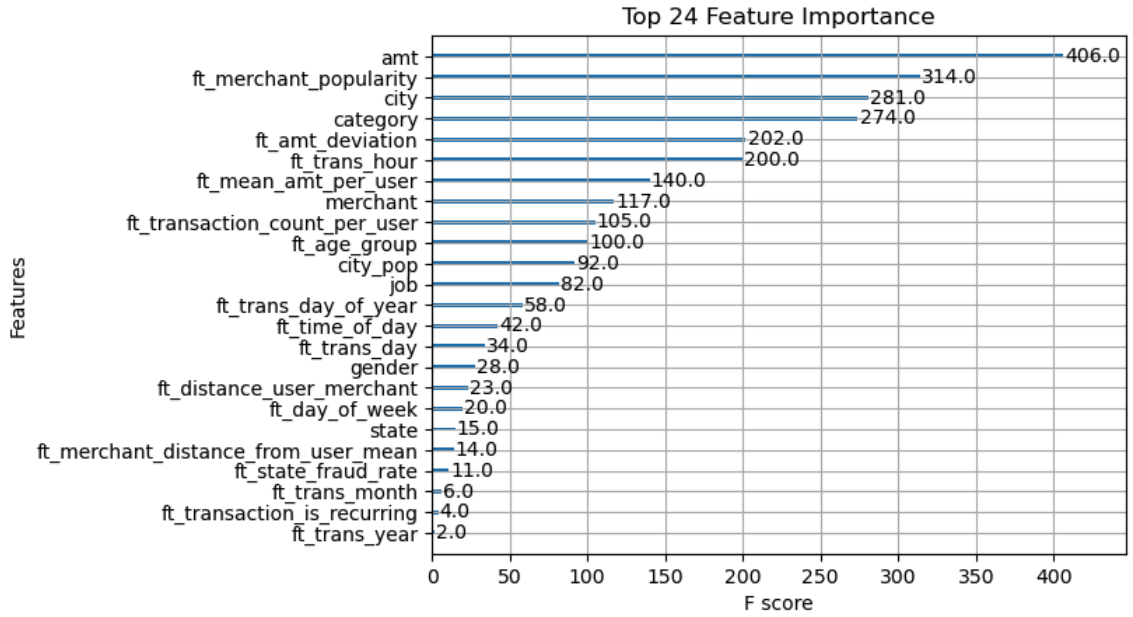
XGBClassifier with AdaBoost Feature Ranking

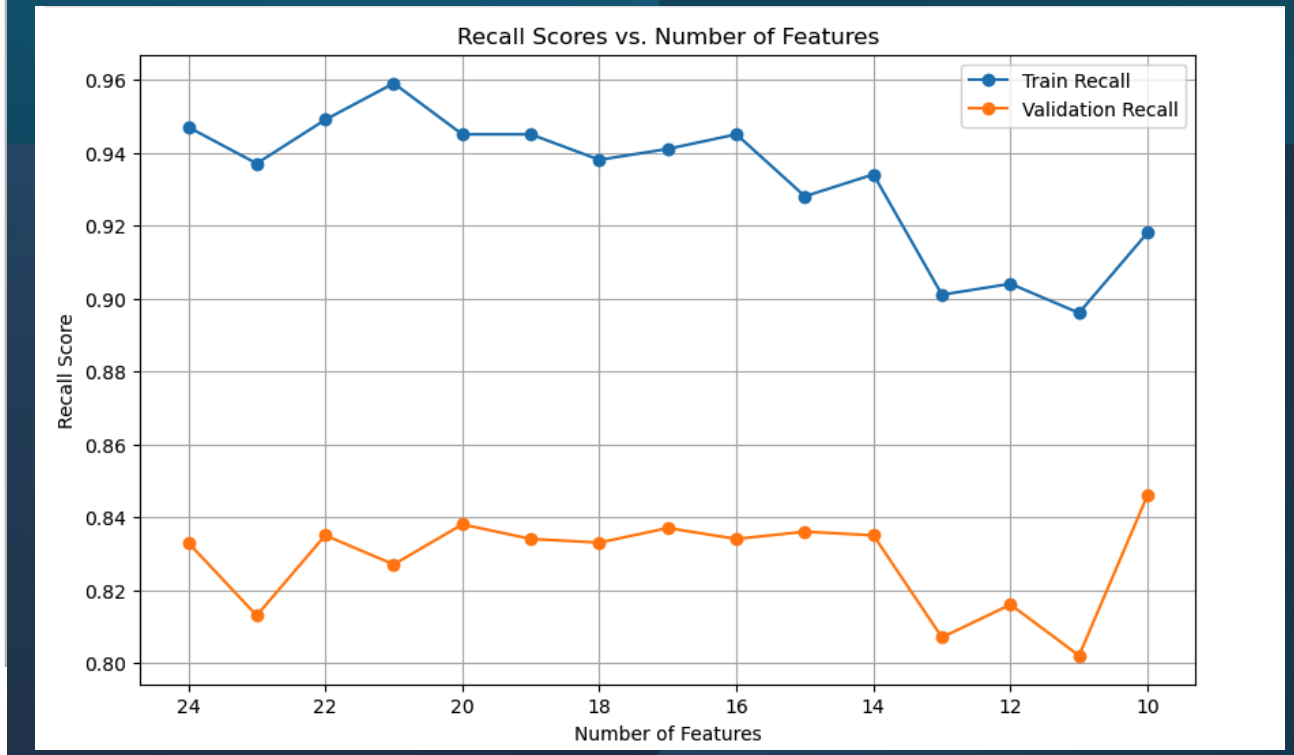
XGBClassifier with advanced data balancing techniques – SMOTE, Class Weighting & Focal Loss



XGBClassifier







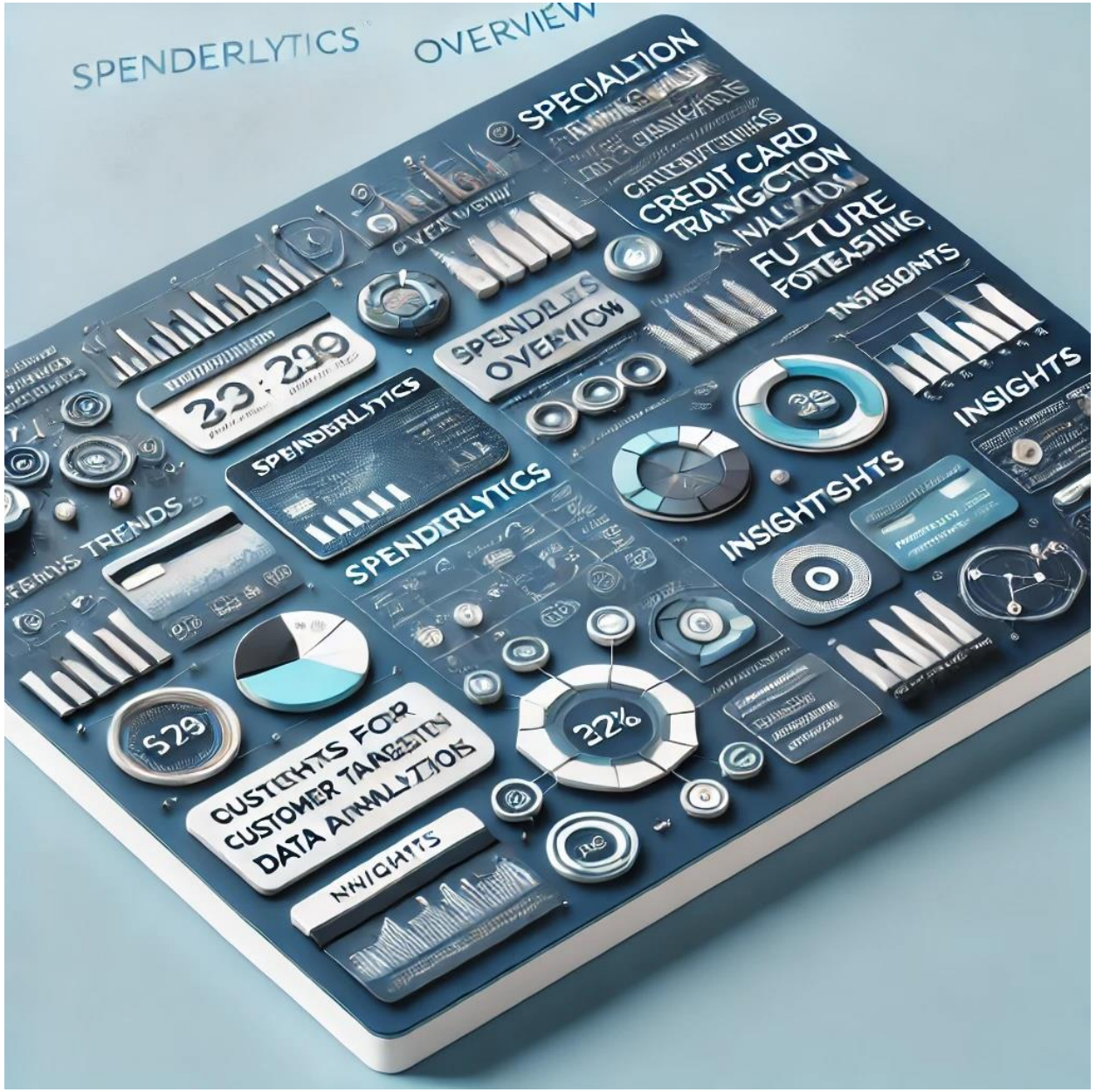
- Poor Test Recall Scores: 0.153 – 0.186
- XGBClassifier with AdaBoost ranking, Test Recall Score improved:
  - Train Recall: 0.707
  - Validation Recall: 0.689
  - Test Recall: 0.688

Poor Performance!!

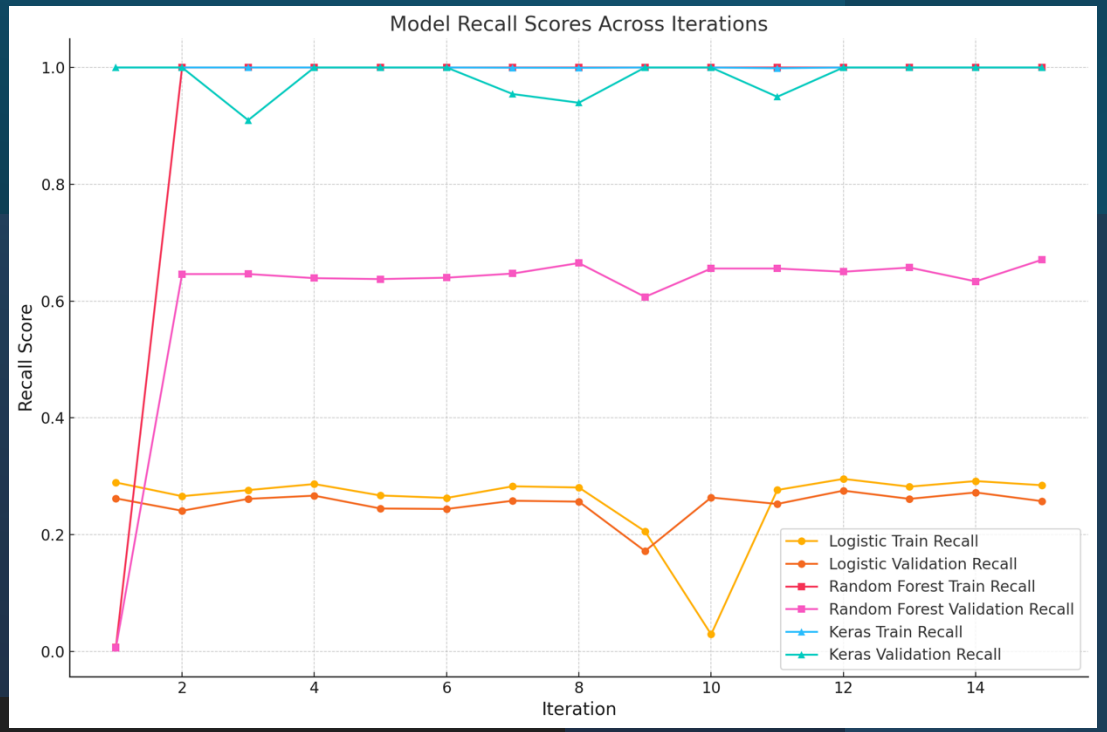
Logistic Regression

Random Forest

Keras



Model Results Comparison



Comparative Insights

Model	Overfitting Trend	Reason
Logistic Regression	**Low Overfitting**	Both training and validation recall remain consistently close.
Random Forest	**High Overfitting**	Training recall is always 1.0, while validation recall is significantly lower.
Keras	**Minimal to Moderate Overfitting**	Validation recall is very close to training recall, indicating better generalization.

Model Comparison

1. Logistic Regression

- Training recall and validation recall are relatively close across all iterations.
- Example (Iteration 1): Training recall = 0.2893, Validation recall = 0.262
- Overfitting **Trend**: Minimal overfitting, as the training and validation recalls are consistently close.

2. Random Forest

- Training recall is often **1.0**, while validation recall is significantly lower.
- Example (Iteration 2): Training recall = 1.0, Validation recall = 0.6462
- Overfitting **Trend**: Strong overfitting. The model perfectly captures fraudulent cases in the training set but struggles on unseen data.

3. Keras

- Training recall is generally **1.0**, and validation recall is often very close.
- Example (Iteration 1): Training recall = 1.0, Validation recall = 1.0
- Example (Iteration 7): Training recall = 0.9996, Validation recall = 0.9545
- Overfitting **Trend**: Minimal to moderate overfitting. Even when validation recall drops slightly, it remains very high and closely matches training recall.

Model Comparison

Logistic Regression Best Iteration:

"Logistic Regression Iteration 12: The Standout Choice"

Summary: Iteration 12 achieves the highest training recall score (0.2954) and validation recall score (0.2753) among all iterations. This indicates the best trade-off between performance on the training set and generalization to unseen data.

Random Forest Best Iteration:

"Random Forest Iteration 15: The Top Performer"

Summary: Iteration 15 emerges as the best-performing model with a perfect training recall of 1.0000 and the highest validation recall of 0.6706. This iteration strikes the optimal balance between overfitting and generalization.

Keras Best Iteration:

"Keras Iteration 7: Balancing Performance and Regularization"

Summary: While many Keras iterations achieve perfect training and validation recall, Iteration 7 stands out with a training recall of 0.9996 and a validation recall of 0.9545. This iteration demonstrates strong performance while avoiding overfitting, making it a well-balanced option for fraud detection.

Model Comparison

Updated Test Recall Scores

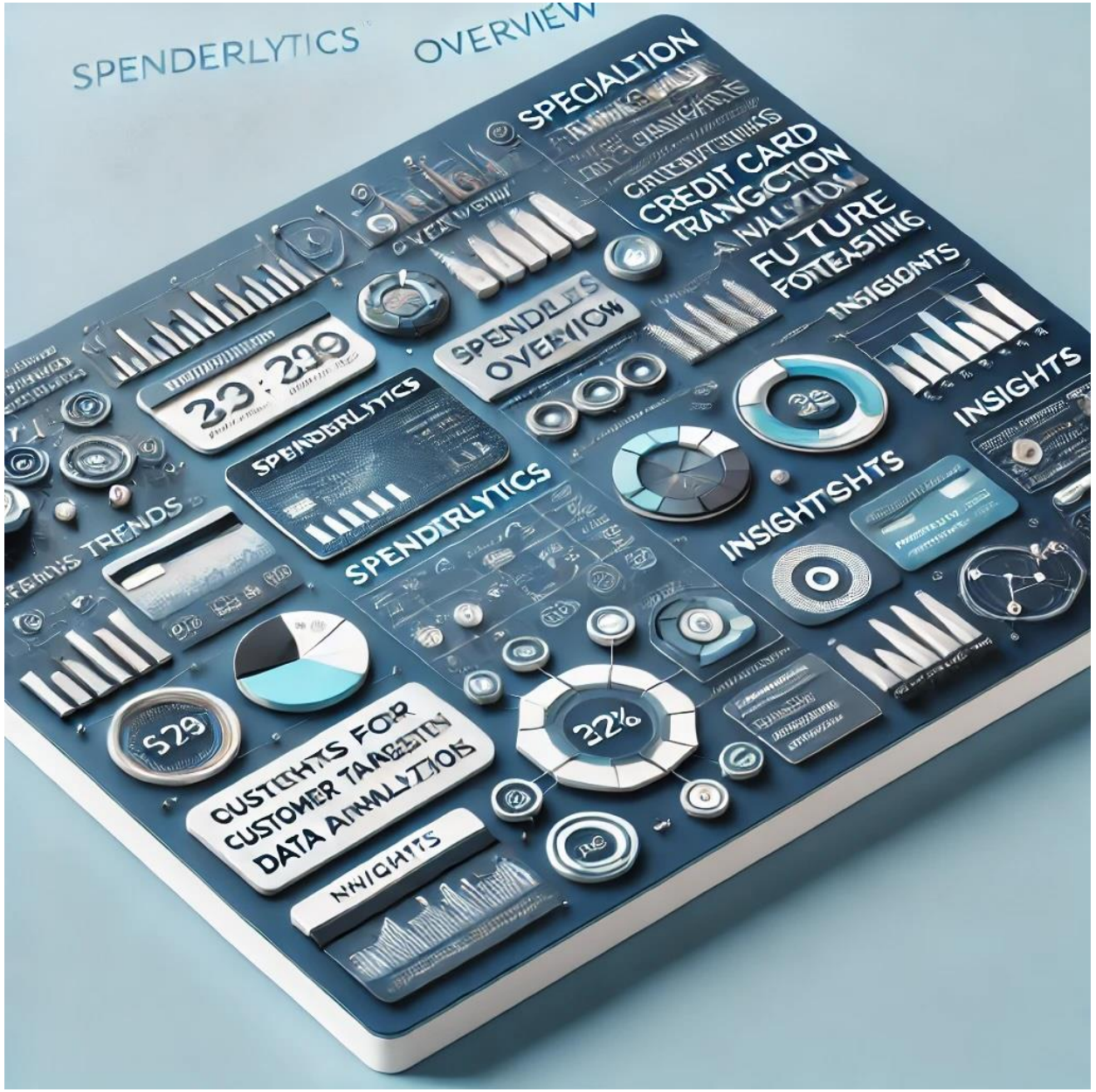
Model	Iteration	Test Recall
Logistic Regression	12	0.0803
Random Forest	15	0.2851
Keras Neural Network	7	0.2507

Updated Analysis

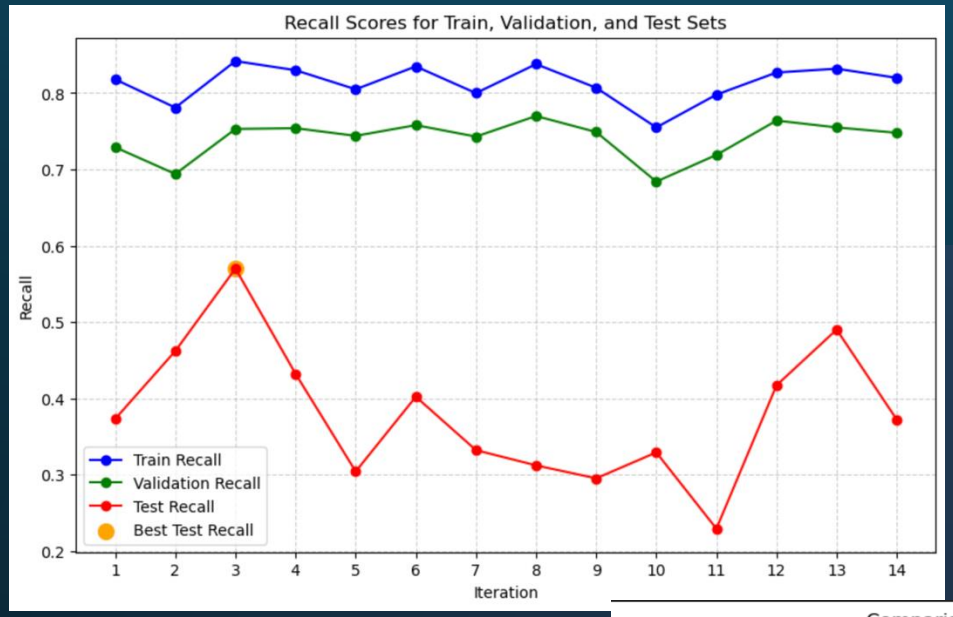
1. **Logistic Regression (Iteration 12)**:
   - **Test Recall**: 0.0803
   - **Performance**: Very low recall, making it unsuitable for identifying fraud cases effectively.
2. **Random Forest (Iteration 15)**:
   - **Test Recall**: 0.2851
   - **Performance**: The best recall among the three models, indicating better generalization and the ability to detect fraud cases.
3. **Keras Neural Network (Iteration 7)**:
   - **Test Recall**: 0.2507
   - **Performance**: Slightly below Random Forest but significantly better than Logistic Regression. Could be a viable option with further tuning.

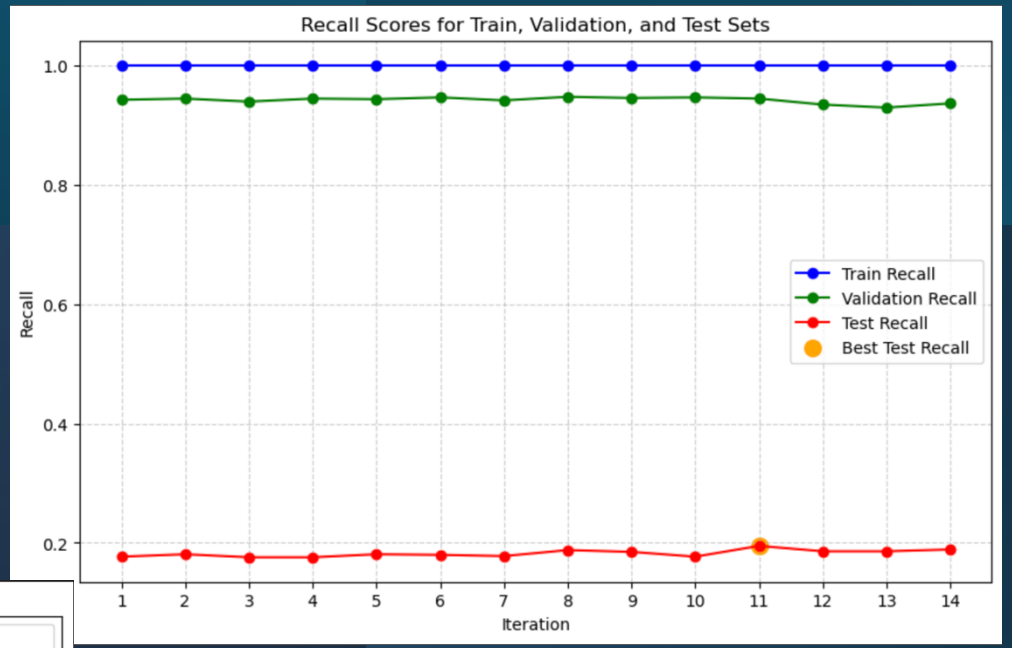
LGBM Classifier

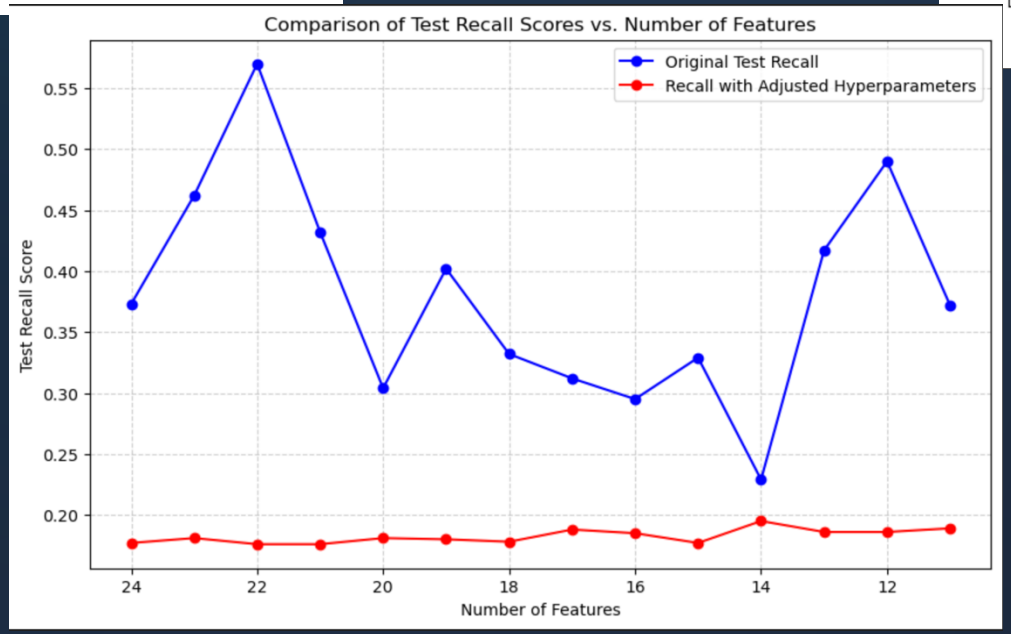
CatBoost Classifier



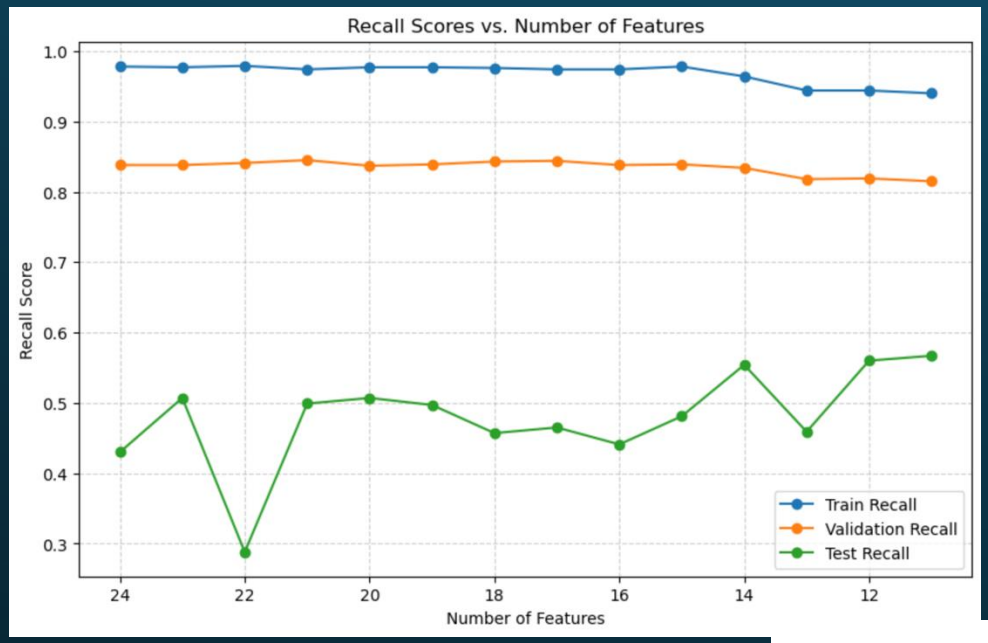
LGBM Classifier

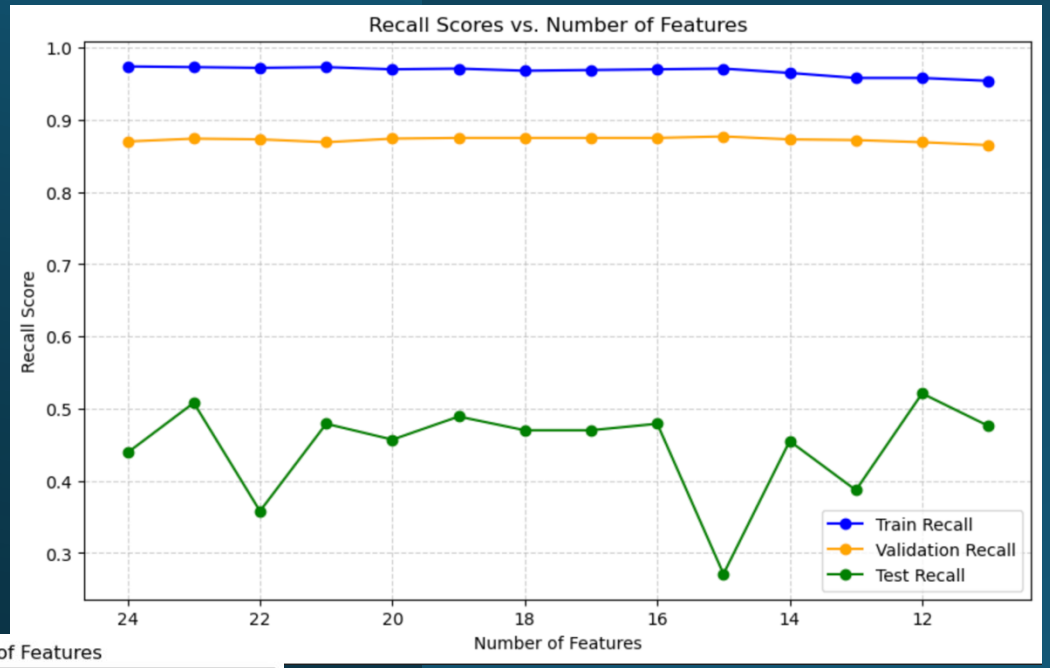


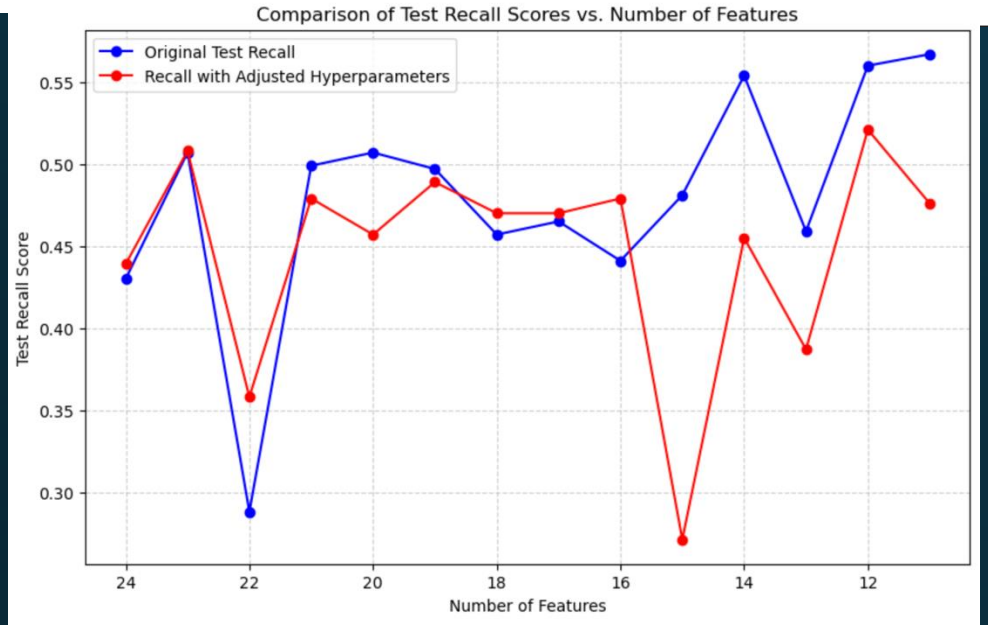




CatBoost Classifier







Stacking Classifier Model

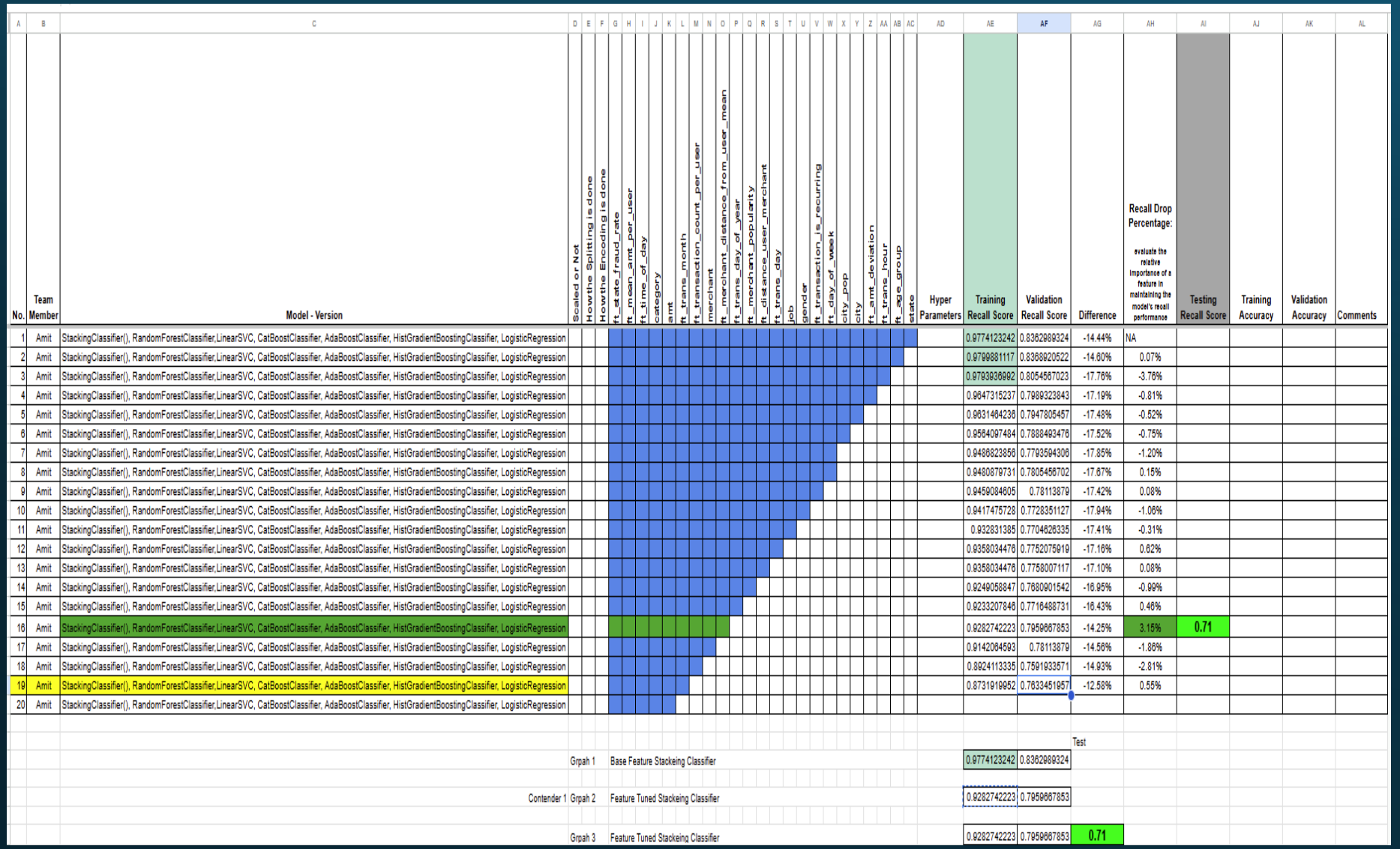
Stacked Estimators

- RandomForestClassifier
- LinearSVC
- CatBoostClassifier
- AdaBoostClassifier
- HistGradientBoostingClassifier

Final Estimator

- LogisticRegression

Stacking Classifier



No.	Team Member	Model - Version	Scaled or Not	How the Splitting is done	How the Encoding is done	ft_state_fraud_rate	ft_mean_amt_per_user	ft_time_of_day	category	amt	ft_trans_month	ft_transaction_count_per_user	merchant	ft_merchant_distance_from_user_mean	ft_trans_day_of_year	ft_merchant_popularity	ft_distance_user_merchant	ft_trans_day	job	gender	ft_transaction_is_recurring	ft_day_of_week	city_pop	city	ft_amt_deviation	ft_trans_hour	ft_age_group	state	Hyper Parameters	Training Recall Score	Validation Recall Score	Difference	Recall Drop Percentage: evaluate the relative importance of a feature in maintaining the model's recall performance	Testing Recall Score	Training Accuracy	Validation Accuracy	Comments
1	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9774123242	0.8362988324	-14.44%	NA				
2	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9799681117	0.8368920522	-14.60%	0.07%				
3	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9793936992	0.8054567023	-17.76%	-3.76%				
4	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9647315237	0.7989323843	-17.19%	-0.81%				
5	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9631464236	0.7947805457	-17.48%	-0.52%				
6	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9564097484	0.7888493476	-17.52%	-0.75%				
7	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9486823856	0.7793584306	-17.85%	-1.20%				
8	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9480879731	0.7805456702	-17.67%	0.15%				
9	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9459084605	0.78113879	-17.42%	0.08%				
10	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9417475728	0.7728351127	-17.94%	-1.06%				
11	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.932831385	0.7704626335	-17.41%	-0.31%				
12	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9358034476	0.7752075919	-17.16%	0.62%				
13	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9358034476	0.7758007117	-17.10%	0.08%				
14	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9249058847	0.7680901542	-16.95%	-0.99%				
15	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9233207846	0.7716488731	-16.43%	0.46%				
16	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9282742223	0.7959667853	-14.25%	3.15%	0.71			
17	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.9142064593	0.78113879	-14.56%	-1.86%				
18	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.8924113335	0.7591933571	-14.93%	-2.81%				
19	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																											0.8731919952	0.7633451957	-12.58%	0.55%				
20	Amit	StackingClassifier(), RandomForestClassifier,LinearSVC, CatBoostClassifier, AdaBoostClassifier, HistGradientBoostingClassifier, LogisticRegression																																		

			Test	
Grpah 1	Base Feature Stackeing Classifier	0.9774123242	0.8362988324	
Contender 1 Grpah 2	Feature Tuned Stackeing Classifier	0.9282742223	0.7959667853	
Grpah 3	Feature Tuned Stackeing Classifier	0.9282742223	0.7959667853	0.71

Data Processing

Preprocessing & Feature Engineer : New features are derived from existing ones.

Category Encoding: Merchant, Category, Gender, City, State, Job, Time_of_day, Age_group, Day_of_week columns are encoded.

Data splitting: Splitting data in train, test and validation sets.

Correlationship with Fraud Indicator: Prioritized weightage of features from low to high

Feature Reduction: The train and validation sets are fed to gauge model score against feature set. Marking model score in each iteration and reducing feature one by one to achieve least drop in recall score. Thus, feature set is shortlisted.

Validation of Model performance: With selected features, the model performance is validated against test set.

Cost Of Model

```
def cost_of_model(x, y_real, y_prediction):
# cost of correct prediction
cost_correct_pred = (number of correction prediction) X
(sum of transaction amount for correct prediction)
# cost of false positive prediction
cost_fp_pred = 0.25 X
(number of false positive prediction) X
(sum of transaction amount for false positive prediction)
# cost of false negative prediction
cost_fn_pred = (number of false negative prediction) X
(sum of transaction amount for false negative prediction)

return {'error_cost' :
cost_fn_pred/ (cost_correct_pred + cost_fn_pred),
'customer_experience_cost' : 1-
(cost_correct_pred/(cost_correct_pred + cost_fp_pred))
}
```

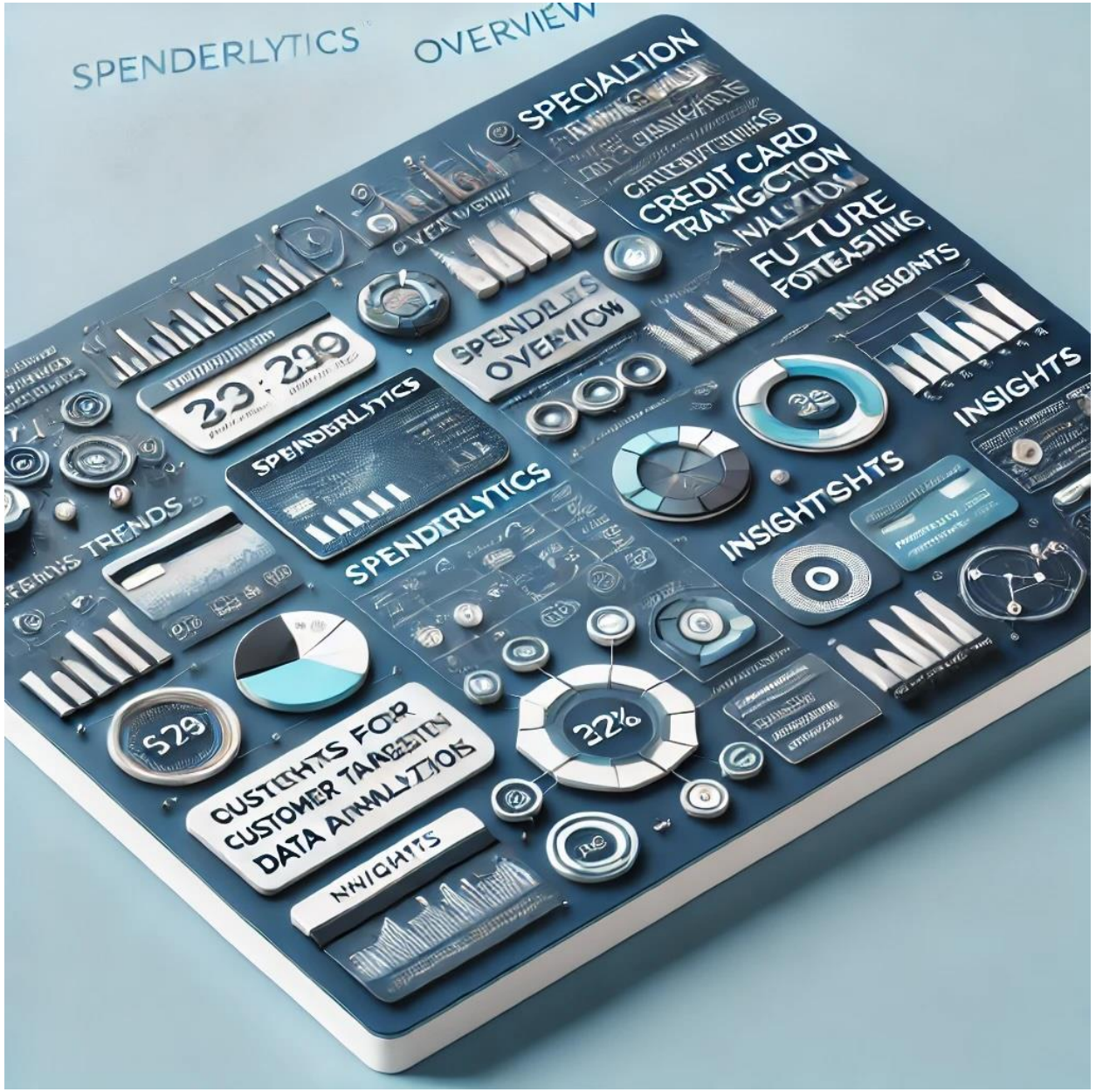
Accuracy Of Model

Recall_Score

Number of True Positive / (Number of True Positive + Number of False Negative)

AdaBoost

Isolation Forest

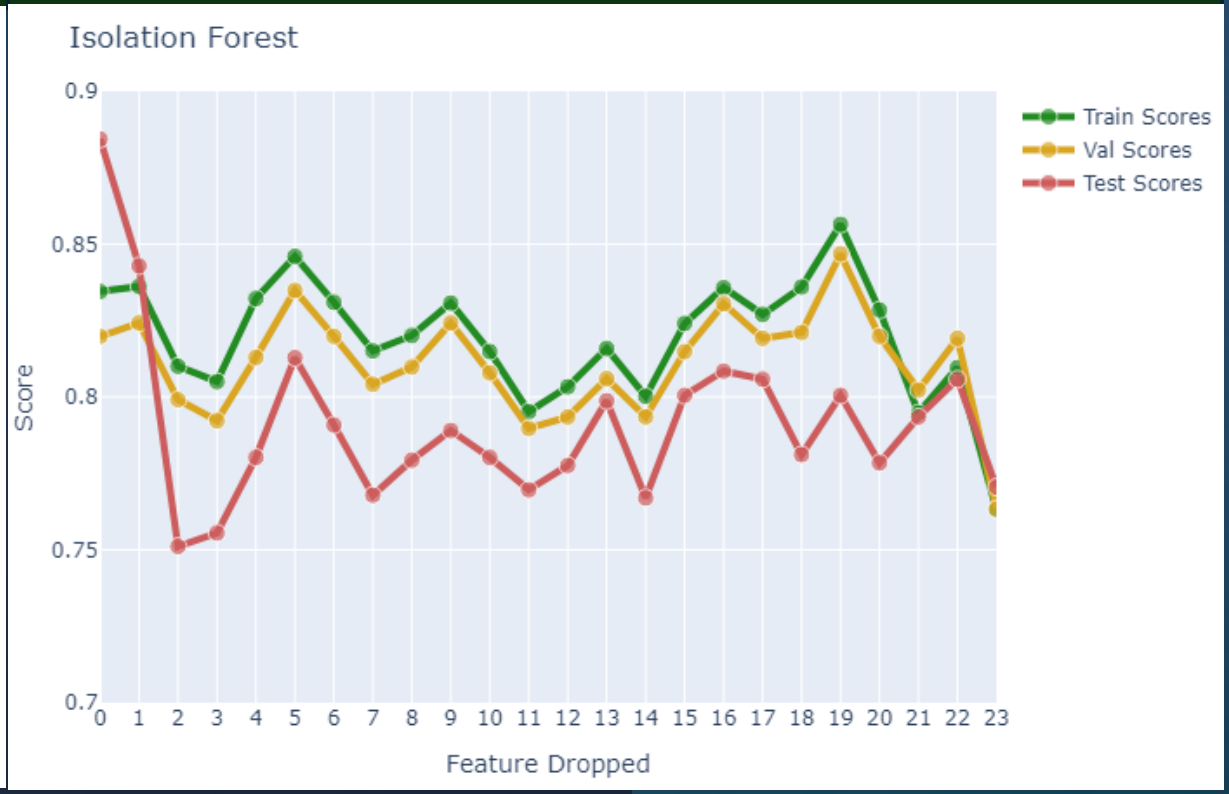


AdaBoost and Isolation Forest

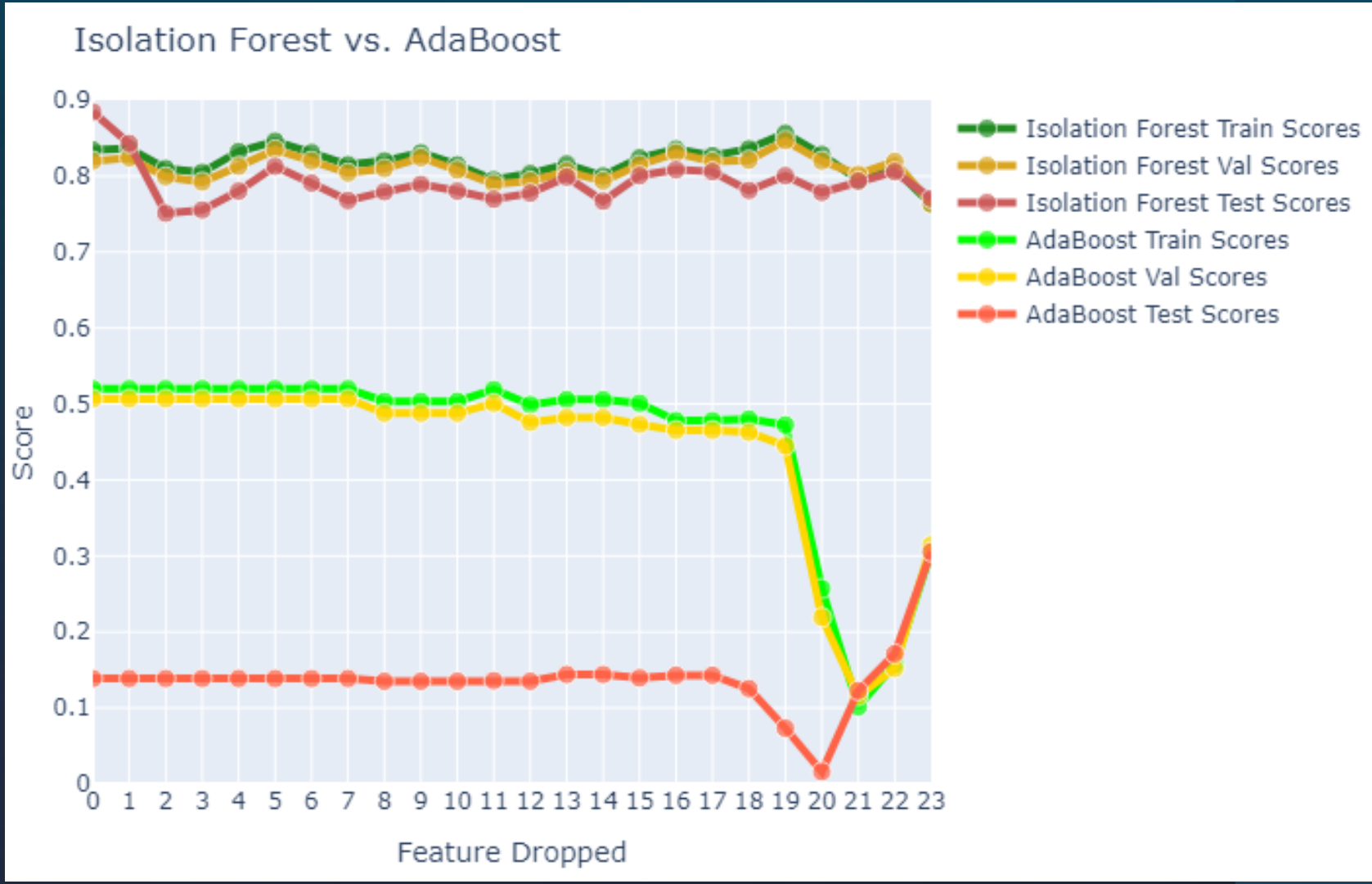
Using the same sorted order of features and dropping one by one

AdaBoost

Isolation Forest

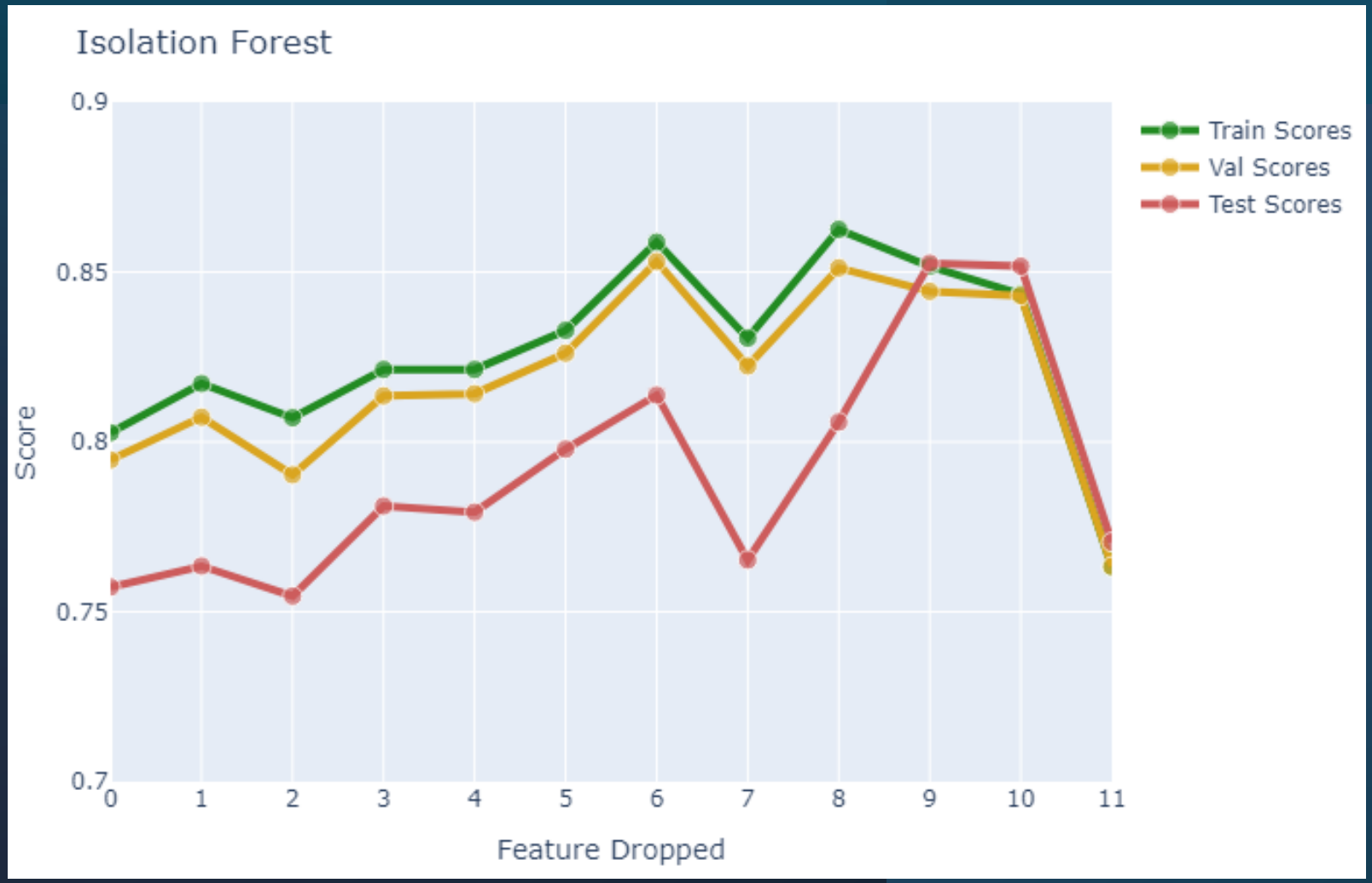


AdaBoost and Isolation Forest



Isolation Forest (AdaBoost Features Importances)

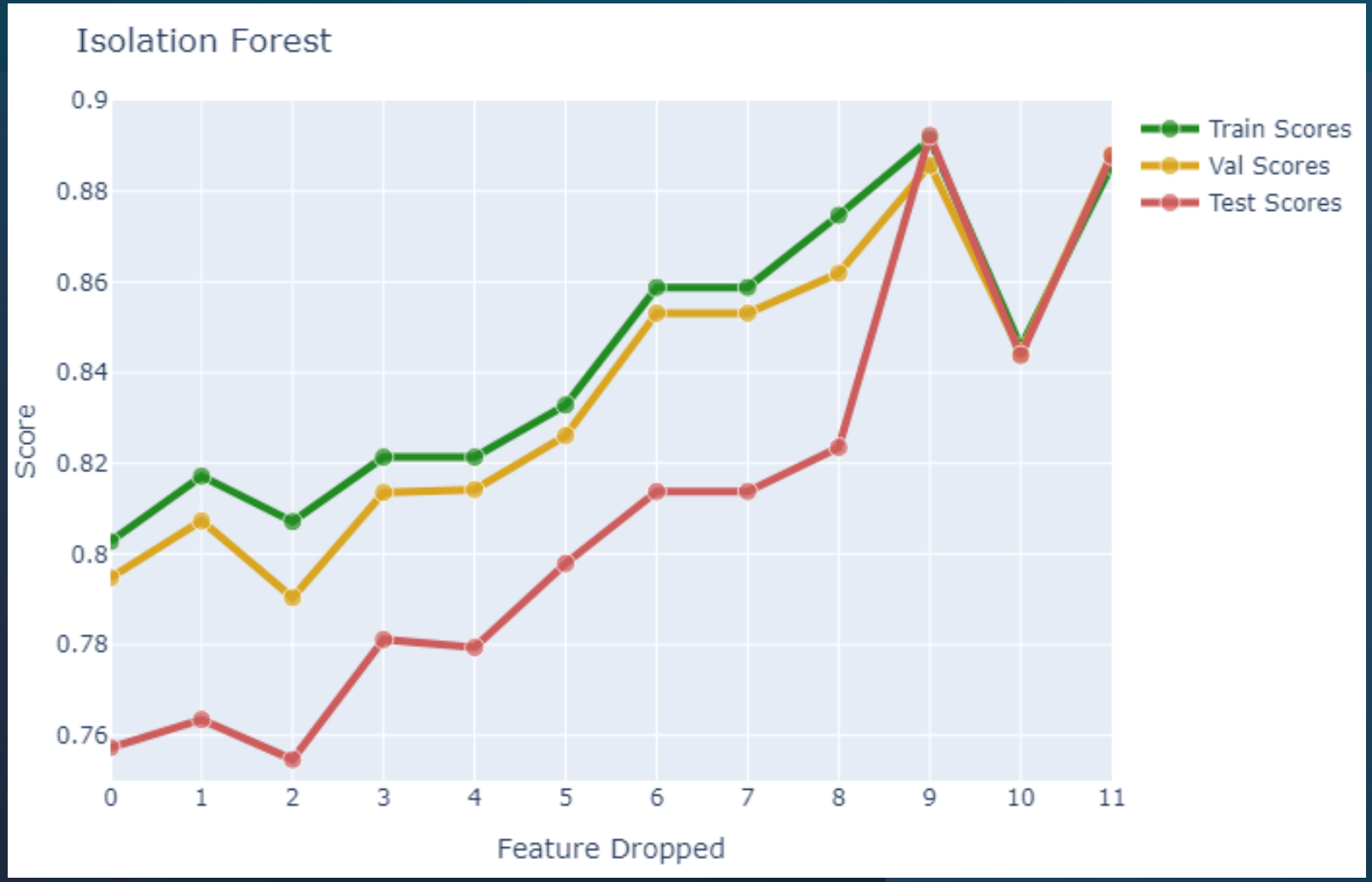
	Features	Values
2	amt	0.34
1	category	0.12
9	ft_trans_hour	0.12
4	city	0.10
0	merchant	0.08
10	ft_time_of_day	0.06
20	ft_amt_deviation	0.06
18	ft_merchant_popularity	0.04
6	city_pop	0.02
7	job	0.02
12	ft_trans_day_of_year	0.02
19	ft_mean_amt_per_user	0.02



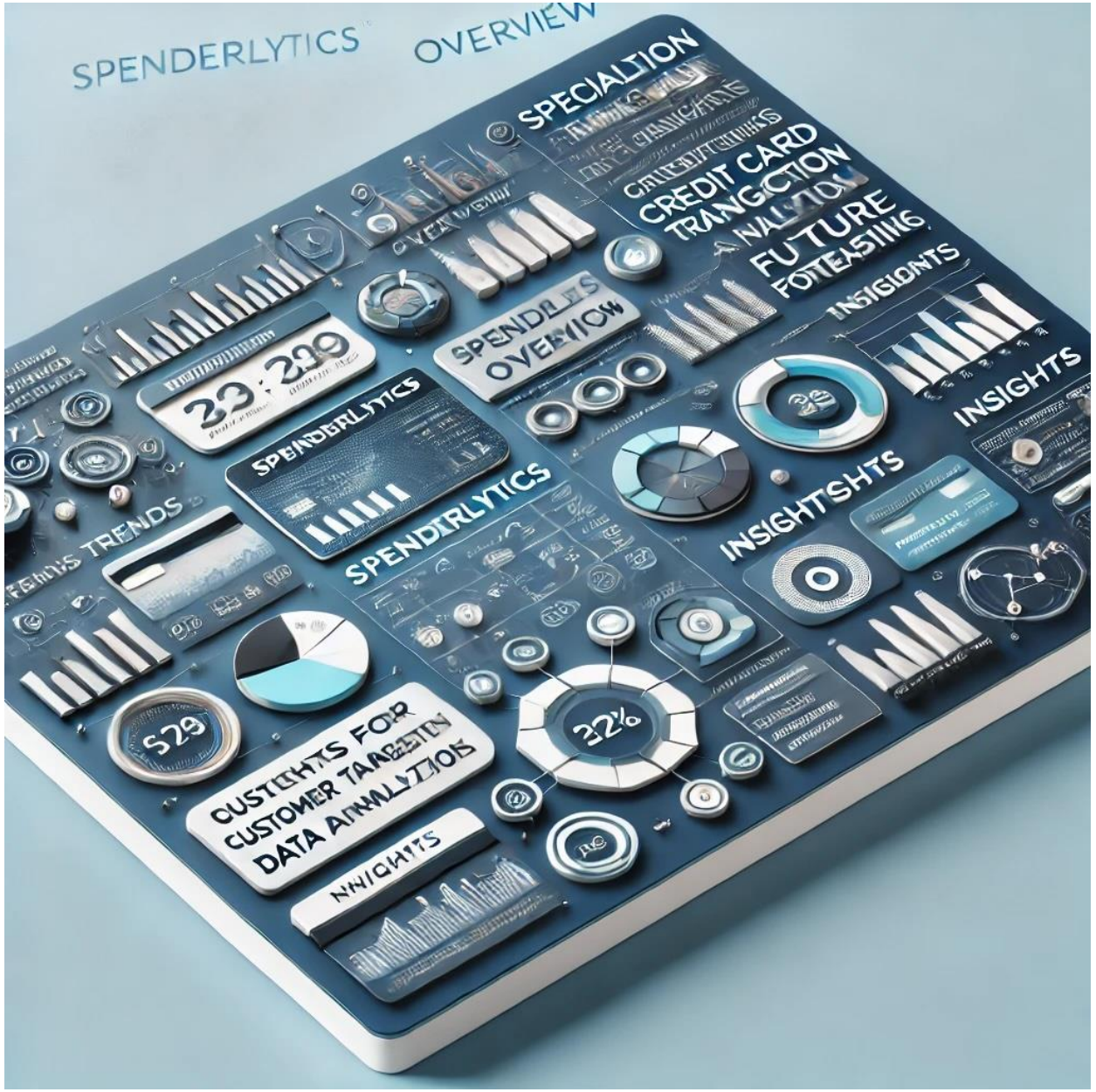
Isolation Forest (AdaBoost Features Importances)

Keeping ft_time_of_day

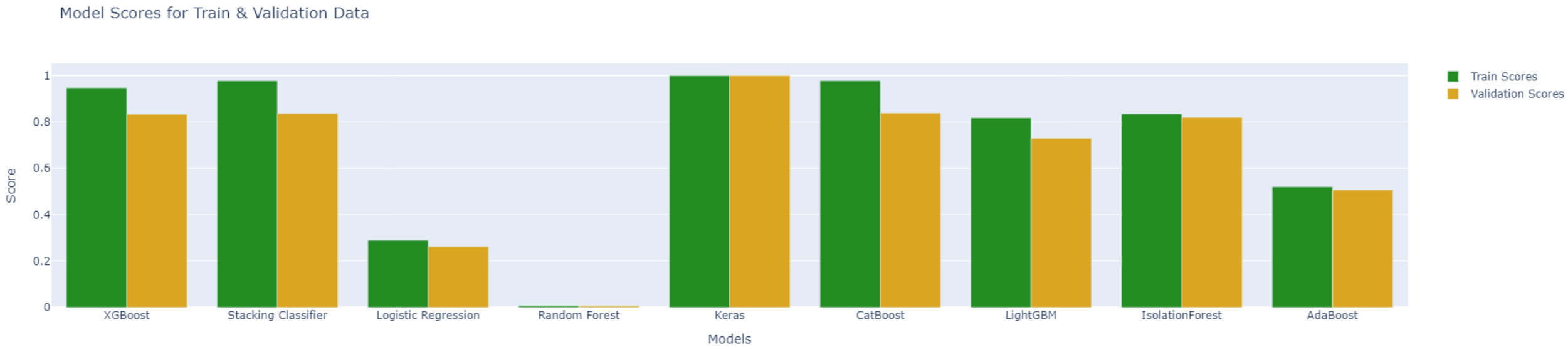
	Features	Values
2	amt	0.34
1	category	0.12
9	ft_trans_hour	0.12
10	ft_time_of_day	0.06



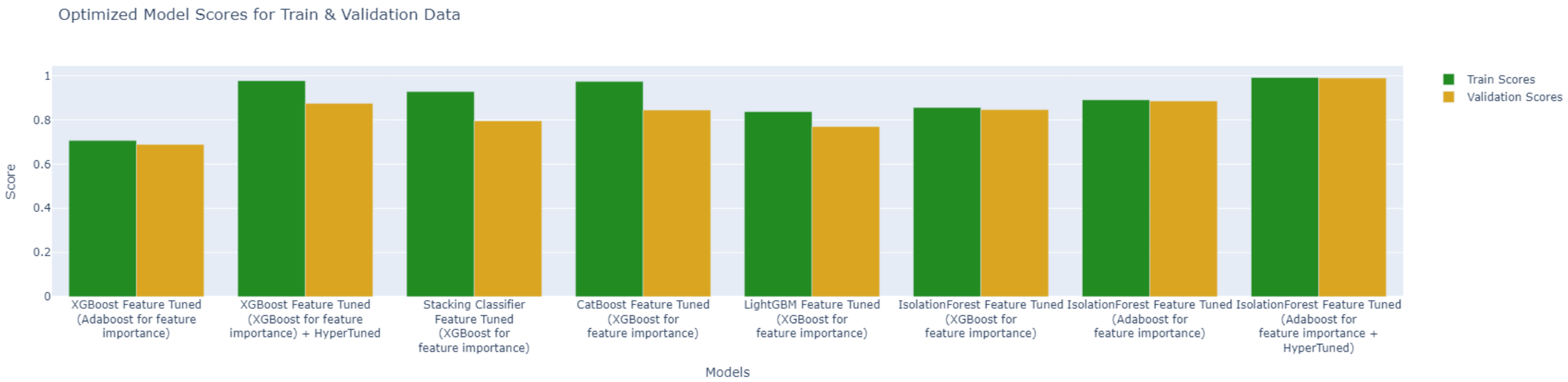
Summary


SPENDERLYTICS
OVERVIEW
SPECIALTION
CREDIT CARD
TRANGCTION
FUTURE
INSIGHTS
INSIGHTS
INSIGHTSHTS
SPENDERLYTICS
SPENDERLYTICS
TRENDS
$29
32%

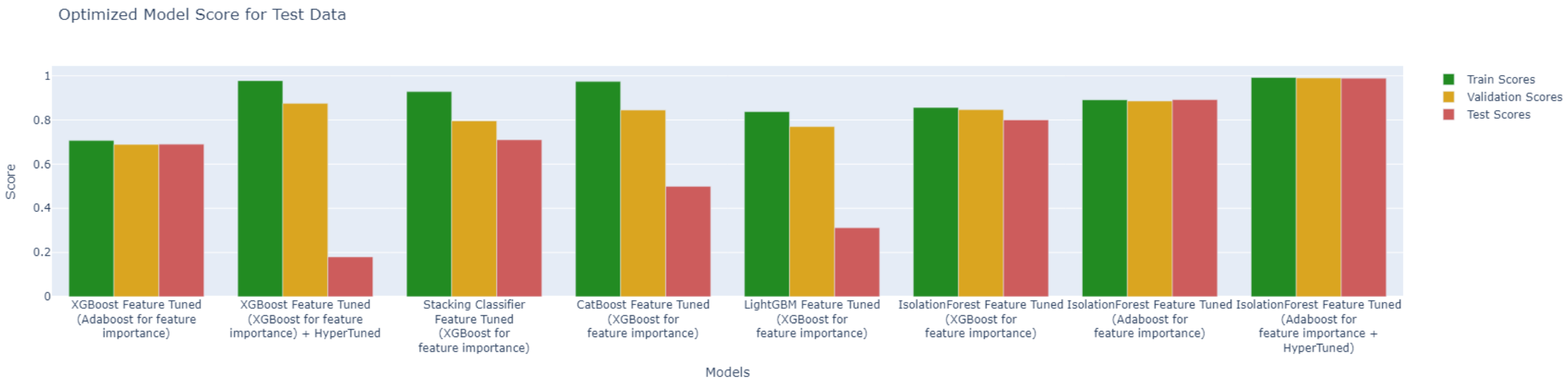
Model Scores for Train & Validation Data



Optimized Model Scores for Train & Validation Data



Optimized Model Scores for Test Data



Challenges

Limited Time: More time would allow for further exploration in terms of model depth and breadth

Limited Computing Power: More computing power would allow for faster model processing and conduct more iterations

Extremely Imbalanced Data: There is a very small percentage of fraud for the models to detect relative to non fraud

Massive Possible Combinations: Between different models, parameters, features, splitting strategies, etc. there are many combinations to test

Designing Custom Metric: Lack industry expertise on the severity and risk of fraud as it relates to business features like cost to the company and customer satisfaction

Future Direction

Real-Time Monitoring: Develop APIs for live transaction analysis.

Scalability: Build scalable architecture for high-velocity data.

Data-Bias Handling: Use advanced sampling techniques to eliminate potential data bias in features like Gender/Loc/Job

Continuous Learning: Use adaptive models to handle evolving fraud patterns.

Thank You!