Credit Card Fraud Detection

Exploration and Advanced Modeling

Presented by Ankit, Lazare and Rylan

Dataset Exploration

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

The dataset comprises 284,807 credit card transactions collected over two days in September 2013 by European cardholders. This real-world dataset contains 31 feature columns derived through Principal Component Analysis (PCA) transformation, ensuring privacy while maintaining analytical utility.

	Time	V1	V2	V3	V4	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.84807
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.6040
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.3802
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.1374
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.9159
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.4335
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.1192
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.4801

8 rows × 31 columns

Features

28 PCA-transformed variables (V1-V28) plus Time, Amount, and Class

Transactions

284,807 records spanning 48 hours of real-world activity

Privacy

PCA anonymization protects sensitive cardholder information

Data Quality Assessment

Datatype Analysis

All 31 columns are numerical (float64), providing a clean foundation for machine learning algorithms. The PCA-transformed features V1 through V28 maintain consistent data types, while Time and Amount retain their original scales for interpretability.

The binary target variable 'Class' indicates fraud (1) or legitimate (0) transactions, creating a supervised learning classification problem.

Missing Value Check

Data integrity assessment reveals zero missing values across all features, eliminating the need for imputation strategies. This completeness is critical for fraud detection where missing data could mask suspicious patterns.

```
Missing values per column:
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

The absence of nulls enables direct modeling without preprocessing overhead, though the extreme class imbalance presents a more significant challenge.

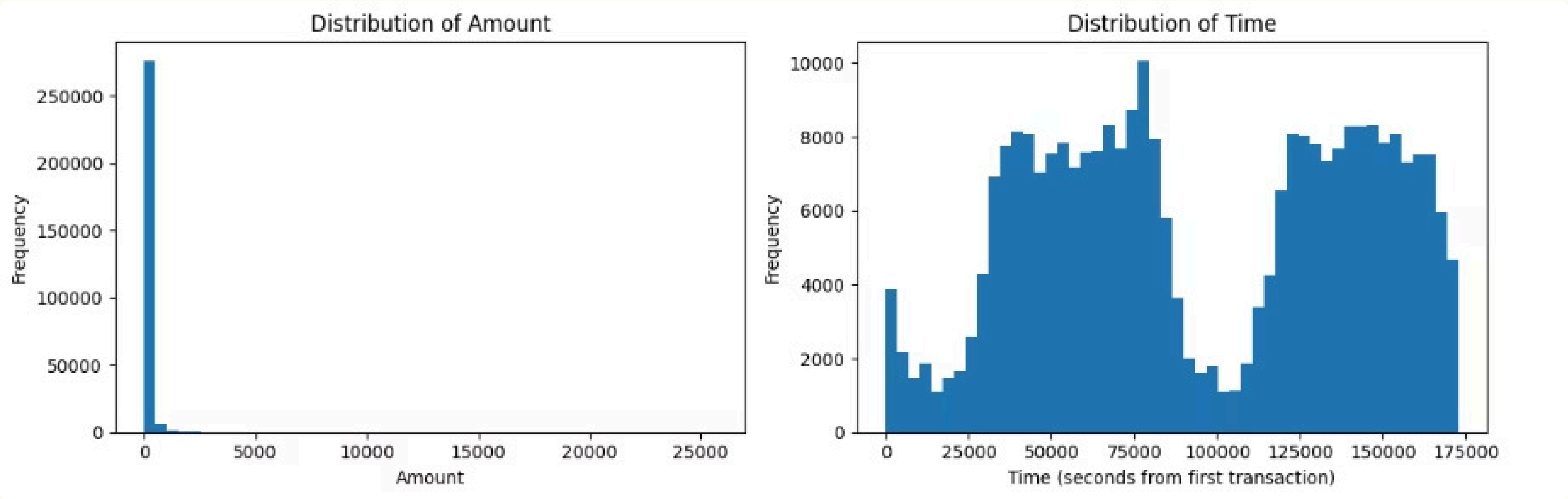
Distribution Patterns in Key Features

Amount Distribution

Transaction amounts exhibit significant right skew, with most transactions clustering below $100 while rare high-value transactions extend the tail. This pattern suggests the need for log transformation or robust scaling to prevent model bias toward large transactions.

Temporal Patterns

The Time feature reveals distinct cyclic patterns corresponding to daily transaction rhythms. Clear peaks and valleys suggest fraud behavior may vary by time of day, making temporal feature engineering a valuable strategy for improving detection accuracy.



These distribution insights drive our preprocessing strategy: scaling Amount to normalize the range, and engineering Time into cyclical features that capture hour-of-day patterns. Both transformations enhance model sensitivity to fraud indicators.

The Class Imbalance Challenge

Extreme Rarity

Out of 284,807 transactions, only 492 are fraudulent—representing just 0.17% of the dataset. This severe imbalance poses a critical challenge: naive models can achieve 99.8% accuracy by simply predicting all transactions as legitimate.

Standard accuracy metrics become meaningless in this context. A model that never detects fraud still appears highly accurate, making precision, recall, and F1-score essential evaluation metrics.

Voting Classifier Performance

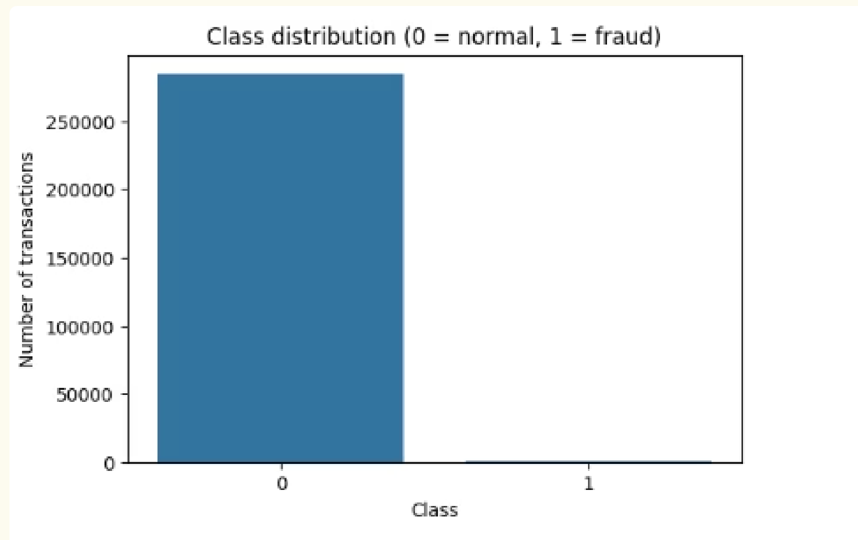
The confusion matrix from our ensemble voting classifier demonstrates the real-world implications. While the model correctly identifies most legitimate transactions, the focus must be on the fraud detection rate and false positive balance.

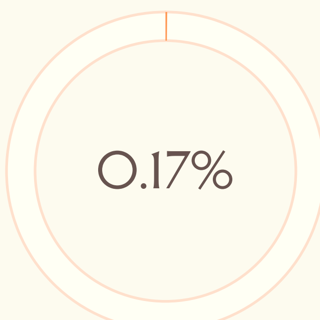
This imbalance necessitates specialized techniques: SMOTE oversampling, class weights, and threshold tuning to optimize for recall while managing alert fatigue.

```
Class distribution:
            count
   Class
Legit(0)   284315
Fraud(1)      492

Class percentages (%):
 Class
0    99.83
1     0.17
Name: proportion, dtype: float64

Majority-class baseline accuracy: 99.83%
Fraud rate: 0.00173 (~0.173%)
```





Fraud Rate

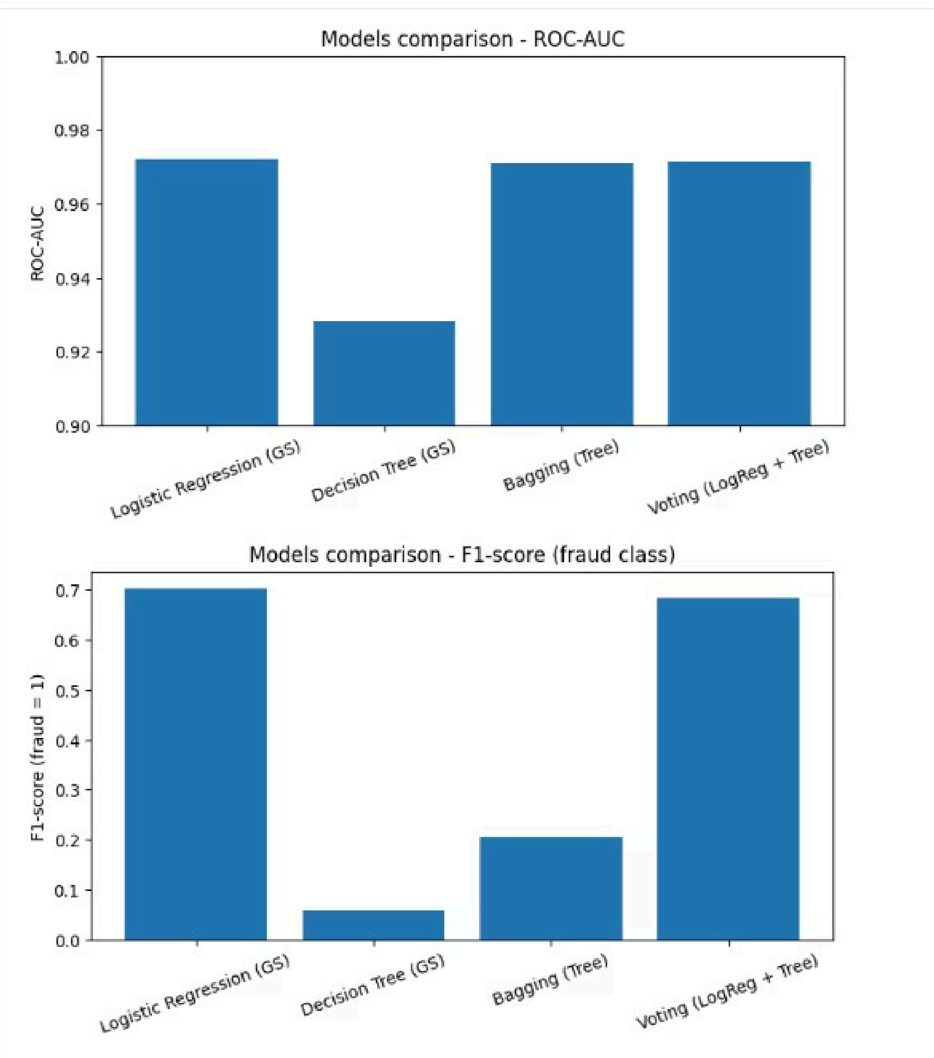
Only 492 fraudulent transactions in the entire dataset

Legitimate

The overwhelming majority are normal transactions

Baseline Model Experimentation

We evaluated multiple classification algorithms to establish performance benchmarks across the imbalanced dataset. Each model was tested with default parameters and class weights to understand their baseline capabilities for fraud detection.



01

Logistic Regression

Simple linear baseline with interpretable coefficients, tested with balanced class weights

02

Decision Tree

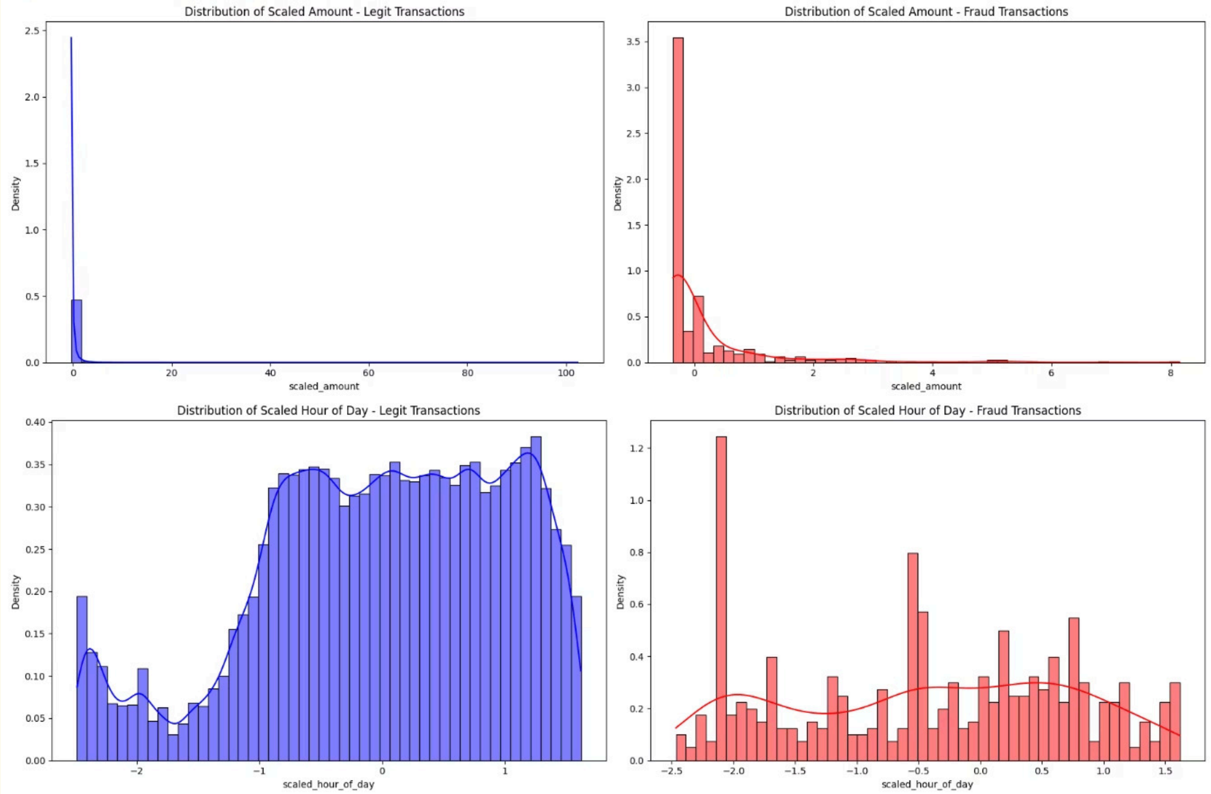
Non-linear classifier capturing complex decision boundaries, prone to overfitting

The baseline results revealed significant performance variation across algorithms, with tree-based ensemble methods showing particular promise for handling the class imbalance inherent in fraud detection scenarios.

Feature Engineering Strategy

	Time	hour_of_day	scaled_hour_of_day	Amount	scaled_amount	Class
0	0.0	0.000000	-2.486373	149.62	0.244964	0
1	0.0	0.000000	-2.486373	2.69	-0.342475	0
2	1.0	0.000278	-2.486326	378.66	1.160686	0
3	1.0	0.000278	-2.486326	123.50	0.140534	0
4	2.0	0.000556	-2.486278	69.99	-0.073403	0

-----------------------Visualize---



Temporal Transformation

Converted elapsed seconds into hour-of-day features to capture fraud patterns tied to specific time windows.

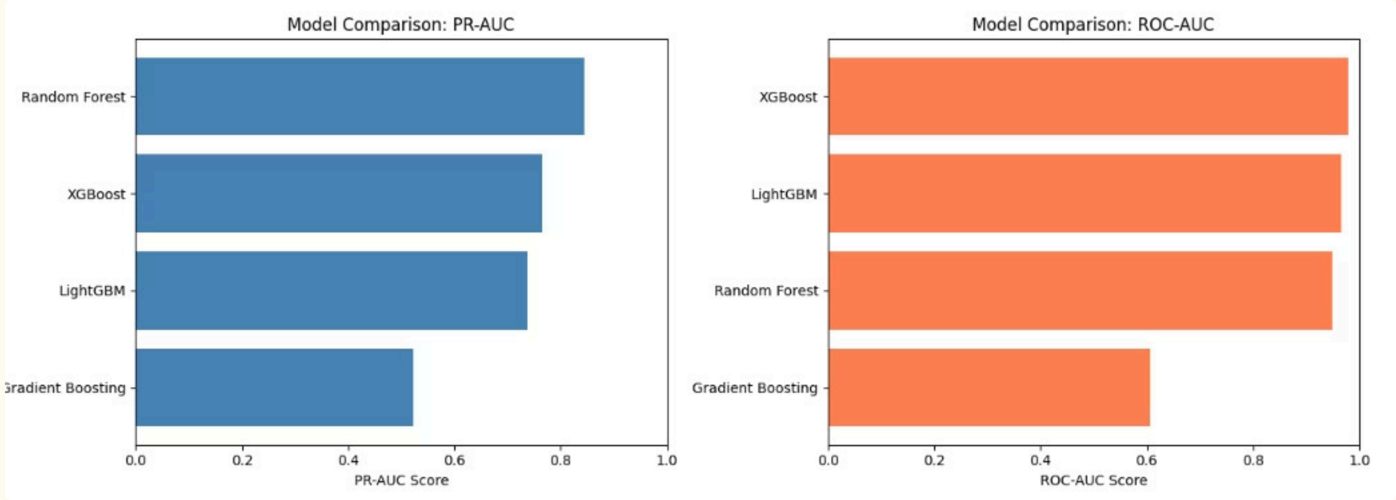
Feature Scaling

Applied StandardScaler to Amount and Time features to normalize their distributions and align with the PCA-transformed V1-V28 variables. Scaling prevents high-magnitude features from dominating distance-based algorithms and gradient descent optimization.

Impact on Performance

These transformations improved model sensitivity to fraud patterns across recall metrics. The hour-of-day features proved particularly valuable, revealing that fraud attempts concentrate during specific time windows.

Model Selection Recommendations



Random Forest for Precision

To minimize false alerts and reduce investigation workload, Random Forest offers the best precision-recall balance. This approach reduces alert fatigue while maintaining acceptable fraud detection rates.

Use case: Operational efficiency focus, limited investigation resources, or scenarios where false positives create significant customer friction.

XGBoost for Maximum Recall

When catching every fraud case is paramount—even at the cost of more false positives—XGBoost delivers superior recall performance. This model excels in high-stakes scenarios where missing a fraudulent transaction carries severe consequences.

Use case: High-value transactions, new account monitoring, or regulatory compliance requirements where recall is the primary metric.

Security Priority

Choose XGBoost when fraud prevention is critical

Operational Balance

Deploy Random Forest to optimize alert management

The optimal choice depends on business context: financial institutions must weigh the cost of missed fraud against operational burden of false alerts. A/B testing both models in production with appropriate threshold tuning will reveal the best fit for your specific use case.

Why Tree-Based Models Excel at Imbalanced Data

Why We Didn't Balance the Data

Our initial notebook focused on an exploration phase to understand how classical models performed on the raw, imbalanced data. This crucial step allowed us to establish a true baseline before introducing more complex preprocessing.

Balancing techniques like SMOTE, undersampling, or oversampling, while effective in some scenarios, can inadvertently distort PCA-transformed features. This distortion introduces synthetic noise that might obscure subtle fraud patterns, especially with highly engineered features like our V1-V28 variables.

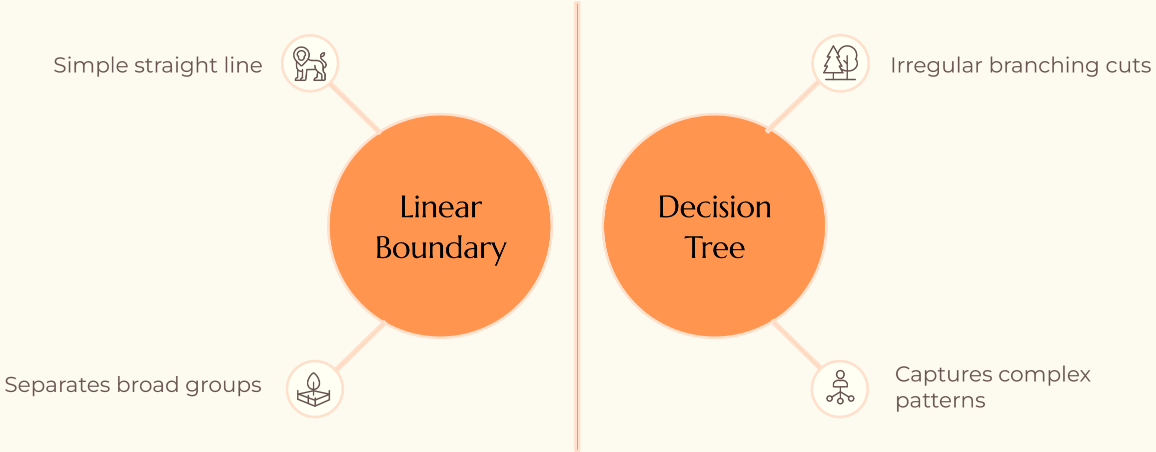
Therefore, baseline experimentation on the original dataset was critical to assess model performance and identify challenges directly, prior to applying advanced data manipulation techniques.

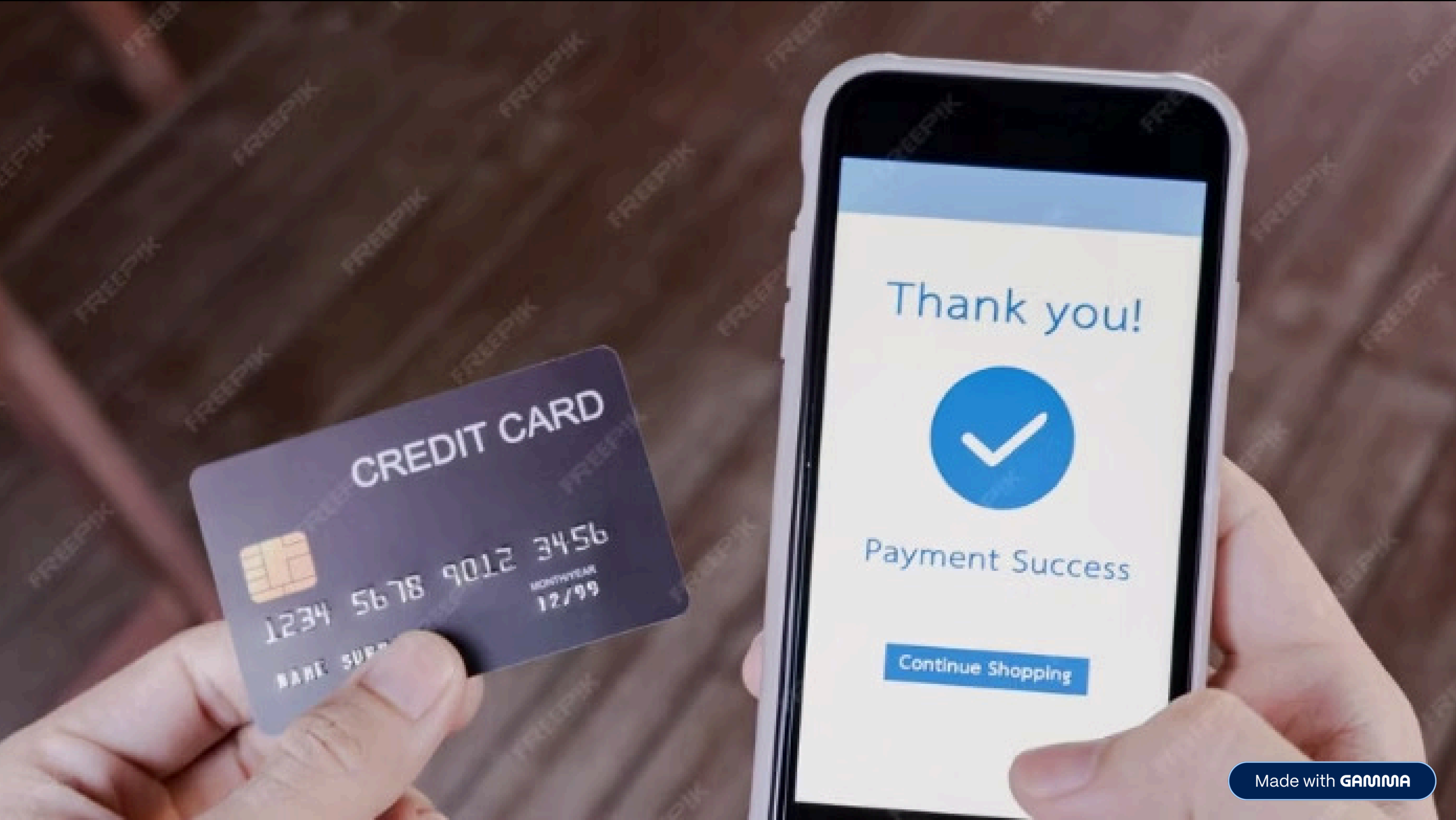
Why Tree-Based Models Work Better

Tree-based models, such as Random Forest and XGBoost, are inherently robust to imbalanced datasets and excel at capturing complex, non-linear patterns that linear models often miss. Unlike linear models that draw a single line to separate classes, tree-based models learn through a series of many small "yes/no" questions about the data.

Even with rare fraud cases, these models are designed to identify and learn the minority class patterns during their tree-building process. They effectively isolate the fraudulent transactions by creating specific branches and rules tailored to their unique characteristics.

This structural advantage means they perform exceptionally well without the need for explicit resampling or other complex data balancing preprocessing steps, naturally handling the class imbalance within their algorithmic design.




CREDIT CARD
1234 5678 9012 3456
12/99
Thank you!
Payment Success
Continue Shopping