Credit Card Transaction Fraud Analytics Project Report

Shreyash Kondakindi

May 5, 2025

Contents

1 Executive Summary

This project developed a fraud detection model for the credit card transactions to help reduce financial losses from fraudulent activities. Using one year of historical credit card transaction data, we built and evaluated various detection models. The final chosen model can identify a large portion of fraudulent transactions while only flagging a small percentage of all transactions for review. In an independent out-of-time test, the model was able to capture about **60% of fraud cases by reviewing just top 3% of transactions**, which translates to roughly **$53 million** in estimated annual fraud savings. This report outlines the data, methodology, model performance, and recommendations in detail, from data preparation to the final financial impact analysis.

2 Description of the Data

The dataset consists of credit card transactions over a 12-month period (January–December 2010). It contains a total of **98,393 transaction records**, each with **10 fields.** Key fields include: record number, card number (identifying the customer account), transaction date, merchant information (ID, name, state, ZIP code), transaction type, transaction amount, and a fraud label indicating whether the transaction was fraudulent (1) or legitimate (0). Overall, about **2.53%** of the transactions in this dataset are labeled as fraud.

Table 1 and 2 summarizes the dataset at a high level. Notably, the transaction amounts range from as low as $0.01 to as high as $3.1 million, with a median around $137 (indicating a strong right-skew due to a few very large transactions). Each card has anywhere from 1 to 1,192 transactions in the data (median 31 per card), reflecting a mix of occasional and frequent card usage. Fraudulent transactions tend to involve higher amounts on average (mean around $1,017) compared to genuine transactions (mean $409), suggesting fraudsters often attempt larger purchases. The data covers transactions across many merchants and states; a handful of entries have missing merchant information (about 3.5% of merchant IDs, 1.2% of states, 4.8% of ZIP codes are missing). Figure 1 illustrates the distribution of transaction amounts, which is highly skewed with most purchases under $1,000 but a long tail of high-value transactions. Figure 2 shows the distribution of transaction counts per card, indicating that while most cards have moderate activity, a small number of cards exhibit very high transaction counts.

In preparation for modeling, the data was partitioned chronologically to enable robust evaluation. For example, the first 8–10 months of transactions were used for model training and internal testing, and the final 2–4 months were held out as an out-of-time (OOT) validation set. This OOT dataset (truly unseen future data) was used to assess how the model performs on new data beyond the training period.

Field Name	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Date	98,393	100.00%	0	365	2/28/10
Merchnum	94,970	96.52%	0	13,091	930090121224
Merch description	98,393	100.00%	0	13,126	GSA-FSS-ADV
Merch state	97,181	98.77%	0	227	TN
Transtype	98,393	100.00%	0	4	P
Recnum	98,393	100.00%	0	98,393	1
Fraud	98,393	100.00%	95,901	2	0 (Not Fraud)
Merch zip	93,664	95.19%	0	4,567	38118
Cardnum	98,393	100.00%	0	1,645	5142148452

Table 1: Categorical Fields Summary

Field Name	# Records Have Values	% Populated	# Zeros	Mean	Std. Dev.	Most Common
Amount	98,393	100.00%	0	424.29	9,922.44	3.62

Table 2: Numeric Fields Summary

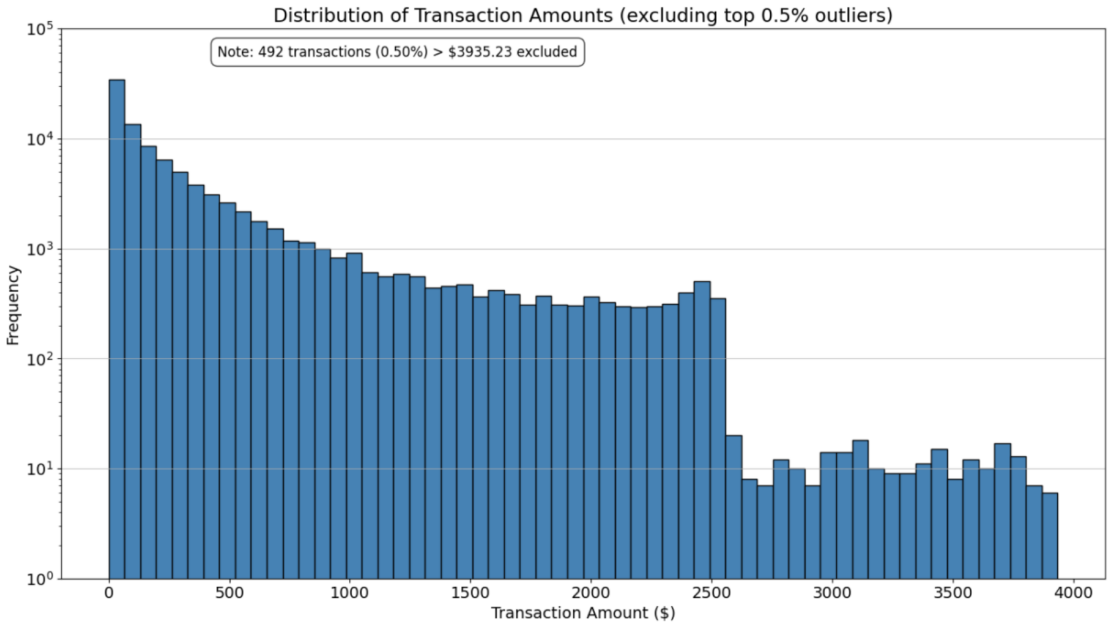


Figure 1: Distribution of transaction amounts (most transactions are low-value, with a few very large outliers).

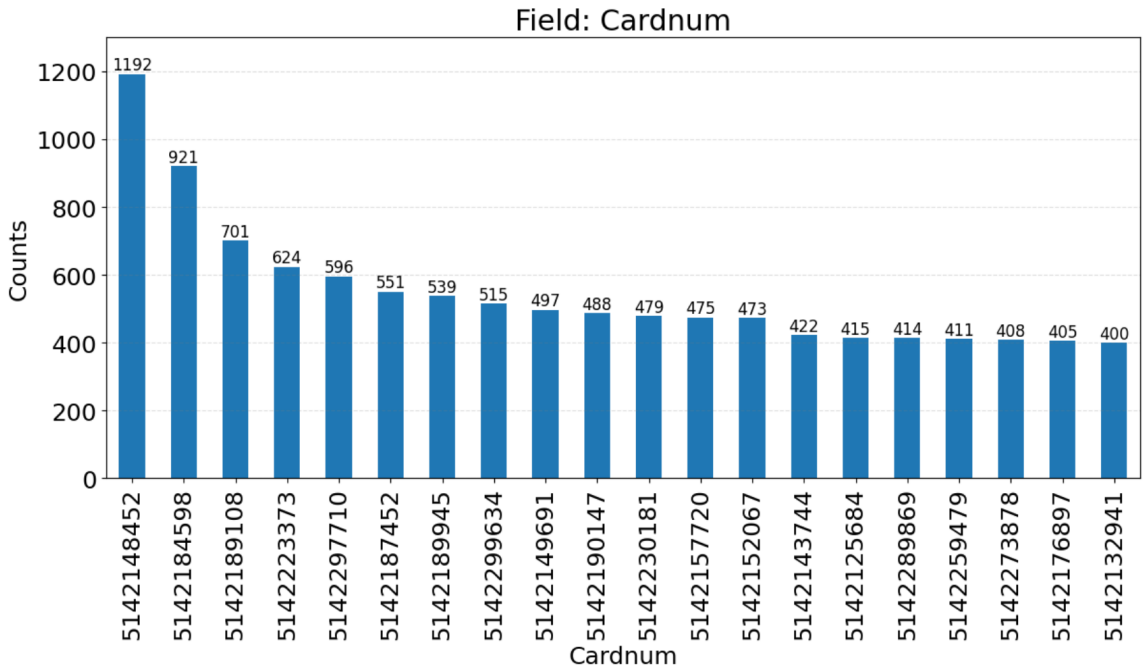


Figure 2: Distribution of number of transactions per card in the data (a small fraction of cards have very high usage).

3 Data Cleaning

Before feature engineering and modeling, we performed data cleaning to ensure data quality and relevance. Key cleaning steps included:

3.1 Outlier Removal:

The dataset was filtered to exclude non-purchase transactions and extremely large transaction amounts. All records with `Transtype` not equal to `'P'` (purchase) were removed, as these were adjustment or non-purchase entries not relevant to fraud analysis. Additionally, transactions with `Amount > 3,000,000` were dropped as outliers (unrealistically high values). This eliminated 356 records and ensured the amount distribution is reasonable for modeling.

3.2 Missing Merchant Number (Merchnum):

Merchant IDs (`Merchnum`) had placeholders `'0'` and `'unknown'` indicating missing values. These were first replaced with `NaN`. A mapping was then created from **Merchant Description** to **Merchant ID** using records where both were present, under the assumption that the same merchant description corresponds to the same ID. This dictionary was used to fill in missing `Merchnum` for transactions that had a known description. After this step, missing `Merchnum` count dropped from 3,306 to 2,136.

3.3 Adjustment Transactions:

Certain transactions labeled as "RETAIL CREDIT ADJUSTMENT" or "RETAIL DEBIT ADJUSTMENT" (system adjustments) lacked valid merchant info. These were handled separately: their `Merchnum` was explicitly set to `'unknown'` for identification. This reduced missing `Merchnum` further (to 1,437). The remaining missing cases (about 1,403) represented unique merchants with no ID available. For each unique merchant description among these, a **new merchant ID** was created (using the next available numeric ID). These new IDs were assigned to all transactions sharing that description. After assignment, `Merchnum` had **no missing values**.

3.4 Merchant State Imputation:

The merchant state field had 1,037 missing entries after the above steps. To fill these, a multi-step strategy was used:

- **Infer from ZIP:** A dictionary mapping from merchant ZIP code to state was built using transactions where state was known. For special cases (e.g., ZIP codes in Puerto Rico like `00926` appearing as `926.0` due to numeric conversion), entries were manually mapped to the correct state (e.g. `'PR'` for Puerto Rico). Using this map, any missing `Merch state` with a known `Merch zip` was imputed with the corresponding state.
- **Foreign vs Unknown:** Any state values not recognized as valid U.S. state codes were labeled as `'foreign'` (assuming those transactions occurred outside the US). After that, any remaining missing states were filled with `'unknown'`. This ensured that `Merch state` had no nulls and that non-U.S. merchants are distinctly marked.

3.5 Merchant ZIP Imputation:

Merchant ZIP had the most missing values (4,373 initially). The cleaning logic mirrored the state process:

- **Infer from Merchant & Description:** A mapping of Merchant (ID) to a known ZIP and another of Merchant Description to ZIP were created from available data. Missing `Merch zip` were first filled by looking up the merchant's ID, and if still missing, by the merchant's description.
- **Adjustment Transactions:** ZIP for adjustment records was set to `'unknown'` (similar to `Merchnum`).
- **Infer from State:** For cases with a known state but still missing ZIP (e.g., a merchant with state info but no ZIP recorded), the ZIP was imputed with the **most populous ZIP code in that state**. A prepared dictionary of state → largest ZIP (by population) was used for this purpose.
- **Finalize Unknowns:** After these steps, the remaining 533 missing ZIPs (which generally corresponded to merchants with state unknown or foreign) were filled with `'unknown'`. This eliminated all missing values in ZIP, while flagging unresolvable cases as `'unknown'`.

These cleaning steps resulted in a refined dataset ready for feature engineering. This ensures that subsequent analysis and modeling are based on consistent and meaningful data.

The rationale behind these steps was to either preserve useful information or clearly mark the data:

- Placeholder values like `'0'` or blank were converted to proper NA and then imputed rather than dropping records, retaining transactions by assigning them plausible or flag values.
- Merchant identities were preserved by using descriptions and creating new IDs instead of dropping those transactions, ensuring subsequent feature engineering could treat them as distinct entities.
- Geographic consistency was enforced by cross-using state and ZIP data: if one was present it informed the other. This maintains realistic location data.
- Special categories (`'foreign'`, `'unknown'`) were used to capture inherently missing or out-of-scope values rather than imputing incorrect data.
- An anomaly in the amount distribution was noted: a **spike at $3.62**. This was investigated separately. As discussed in class, these correspond to the shipping transactions through FEDEX. Histograms were plotted to see the difference in frequency of amounts before and after these FEDEX transactions were filtered from the dataset. As predicted, the spike around $3.62 has disappeared.

4 Feature Engineering

With a clean dataset, we created additional variables to enrich the information available to the model. The goal of feature engineering was to capture behavioral patterns and context (both at the card level and transaction level) that might distinguish fraudulent transactions from legitimate ones.

4.1 Entity Creation and Target Encoding

Before engineering new variables, the dataset was structured around a series of defined **entities** to support feature generation. These entities served as grouping keys for behavioral and temporal aggregations, including:

- `Cardnum`
- `Merchnum`
- `Merch description`
- `Dow` (day-of-week)
- `Cardnum + Merchnum`
- `Cardnum + Merch description`
- `Cardnum + Dow`
- `Merchnum + Dow`
- `Merch description + Dow`
- `Cardnum + Merchnum + Merch description`
- `Cardnum + Merchnum + Merch zip`
- `Cardnum + Merch description + Merch zip`
- `Cardnum + Merchnum + Merch state`
- `Cardnum + Merch description + Merch state`
- `Merchnum + Merch description + Merch state`
- `Merchnum + Merch description + Merch zip`
- `Merchnum + Merch zip`
- `Merchnum + Merch state`
- `Merch description + Merch zip`
- `Merch description + Merch state`

Each of these combinations was filtered to retain only those entity pairs with **at least 10 prior transactions** before a given point in time. This threshold helps prevent overfitting and reduces the influence of sparse or unrepresentative history in later feature calculations.

In parallel, the dataset underwent **target encoding** to convert high-cardinality categorical variables into smoothed numerical indicators of fraud risk. A `TargetEncoder` class was implemented to compute:

- The mean fraud rate per category (e.g., per ZIP or state)
- Smoothed estimates using a weighted average of the global fraud rate and the local (category-specific) rate
- Out-of-fold encoding logic to ensure no row uses its own target value for encoding (avoiding leakage)

This resulted in the following target-encoded features:

- `Merch state_TE` – smoothed fraud probability for merchant's state
- `Merch zip_TE` – smoothed fraud probability for merchant's ZIP code
- `Dow_TE` – smoothed fraud probability for the day-of-week of the transaction

These encoded variables capture population-level risk signals and serve as informative, low-dimensional predictors for the model.

4.2 Variable Creation

Feature Family	**Description and Purpose**	**# Variables**
Card Transaction History	`Card-level velocity features:` For each credit card, transaction counts and amount statistics (average, max, median, total) were computed over rolling time windows (0, 1, 3, 7, 14, 30, 60 days prior). Includes recency features (*day_since* last transaction). These metrics capture the spending frequency, magnitude, and recent activity of the cardholder, highlighting anomalies or sudden spikes indicative of fraudulent behavior.	215
Merchant Transaction History	Merchant-level aggregates including the number of transactions, transaction amount statistics (average, max, median, total), and recency metrics computed over rolling windows (0-60 days). This identifies typical merchant activity and size patterns, crucial for spotting anomalous merchant behavior or transactions that deviate from historical norms.	208

Card-Merchant Interaction History	Historical interaction metrics between a specific card and merchant, including transaction frequency, amounts, and recency. Designed to highlight first-time interactions or unusual activity patterns, these variables serve as strong fraud indicators by identifying irregularities in card-merchant relationships.	126
Card-ZIP Transaction History	Aggregated historical transaction data of cards within specific ZIP codes, detailing counts and amounts over multiple time windows. Useful for detecting geographic transaction anomalies or unusual local spending behaviors of cardholders.	126
Card-State Transaction History	Similar to ZIP-level but aggregated at the state level, these features track transaction counts and amounts, assisting in identifying uncommon or anomalous state-level transaction activity for cards.	126
Merchant-ZIP Transaction History	Transaction aggregates for merchants within specific ZIP codes, capturing typical location-specific transaction volume and amounts. Helps flag transactions deviating significantly from a merchant's usual geographic activity patterns.	154
Merchant-State Transaction History	Aggregation of merchant transactions by state, summarizing typical regional transaction behaviors and facilitating the detection of anomalies related to a merchant's state-level activity.	138
State-Descriptor Aggregate History	Aggregates transaction data by merchant type (descriptor) within states. These features capture overall regional transaction behaviors for merchant categories, assisting in identifying unusual spending patterns or activity anomalies within specific states.	156
Card-Descriptor Interaction History	Historical spending behavior of cards with specific merchant types. These aggregates highlight whether merchant category transactions are consistent or anomalous for each card, flagging unusual spending patterns indicative of potential fraud.	138
Merchant-Descriptor Combination History	Aggregated metrics for merchants, explicitly including merchant type context, reinforcing typical merchant behaviors and transaction patterns, aiding in anomaly detection.	132
Merchant Descriptor Aggregate Patterns	Broad transaction patterns of merchant types across various times and locations, identifying category-based anomalies or deviations from typical merchant descriptor behaviors.	235
Card, Merchant, ZIP Interaction	Granular features capturing historical card interactions with specific merchant locations (ZIP code), designed to flag unusual geographic card-merchant interactions or anomalies at localized levels.	132

Card, Merchant, State Interaction	Metrics aggregating card transactions with merchants within specific states. These detailed interactions assist in identifying transactions where cards engage with known merchants but in unfamiliar or unusual geographic contexts.	107
Card, Merchant, Descriptor Interaction	Detailed interaction history combining cards, specific merchants, and merchant categories, providing nuanced detection of anomalies in transaction patterns, strengthening fraud detection capabilities.	126
Card, Descriptor, ZIP Interaction	Card interaction history with merchant categories within ZIP codes, used to identify localized and category-specific anomalies or uncommon spending behaviors.	126
Card, Descriptor, State Interaction	Card spending behavior with merchant types aggregated by state, highlighting atypical or unusual state-level category interactions for individual cards.	132
Merchant, Descriptor, State Interaction	Transaction history aggregation for merchant types within states, useful in identifying state-specific anomalies in merchant transaction behaviors and activities.	132
Merchant, Descriptor, ZIP Interaction	Merchant-specific transaction metrics aggregated within ZIP codes, aimed at spotting unusual local merchant transaction behaviors or anomalies.	127
Risk Encoding Features	Target-encoded historical fraud rates for categorical attributes (state, ZIP, day-of-week). These variables provide smoothed estimates of fraud likelihood based on historical patterns, enhancing predictive modeling accuracy.	3
Transaction-Level Features	Fundamental details of transactions including transaction amount and weekend indicators, offering basic contextual information critical for initial fraud risk assessment.	4
Distance-Based Features	Geographic metrics measuring distances between sequential transactions and flags indicating unusual distances, highlighting potential improbable travel scenarios or geographic anomalies indicative of fraud.	2
Unsupervised Anomaly Features	Unsupervised machine learning-derived anomaly scores (latent factors) for cards and merchants. These features identify atypical transaction behaviors or patterns without using label information directly.	2
Heuristic Anomaly Flags	Simple rule-based indicators flagging high-value transaction anomalies or inconsistent geographic transaction information, enabling rapid identification of clearly suspicious activities.	2

Benford's Law Features	Statistical deviation metrics from Benford's law expected distribution for transaction amounts. These features serve as forensic indicators, highlighting potential numerical manipulation or fraud-related anomalies.	∼40

Table 3: Variable Creation Summary

In total, dozens of new features were generated to capture these dimensions. Table 3 provides a summary of some important engineered features. All engineered features were computed only using past data relative to each transaction's time, to mimic a real-time scoring scenario where future data wouldn't be available.

5 Feature Selection

After generating a large pool of candidate features, we conducted feature selection to reduce complexity and avoid overfitting. The initial feature set was quite broad, and not all features would be useful for the model. Some features might be redundant or not predictive, so we tried multiple selection techniques to identify an optimal subset.

We experimented with both forward selection (starting from no features and adding the most useful features one by one) and backward elimination (starting with all candidate features and removing the least useful ones iteratively). These trials were done using a wrapper approach, where a machine learning model evaluates each feature subset's performance. We also applied an initial filtering step based on univariate analysis to narrow down the pool (for instance, selecting the top 200 features by correlation or information value before applying wrapper methods, to limit computational load).

Trial (Parameters)	**Description and Purpose**	**# Features**
Trial 1: Filter = 200; Wrapper = 20; Model = LGBMClassifier ($n\_estimators$ = 30,$num\_leaves = 6$, $max\_depth = 4$)	More trees to capture additional signal; increased complexity while controlling overfitting; CV=3-fold; performance saturated at $\sim$ 0.68 before 15 features.	20
Trial 2: Filter = 300; Wrapper = 20; Model = RandomForestClassifier ($n\_estimators = 7, max\_leaf\_nodes = 10$)	Explored a different inductive bias; CV=2-fold; performance $\sim 0.68 \pm 0.05$, no clear improvement.	20
Trial 3: Filter = 250; Wrapper = 25; Model = LGBMClassifier ($n\_estimators = 25, num\_leaves = 8, max\_depth = 5$)	Added depth and leaves to boost capacity; CV=2-fold; performance improved to $\sim$ 0.705, saturation around 15–20 features.	25
Trial 4: Filter = 250; Wrapper = 25; Model = CatBoostClassifier ($iterations = 30, depth = 6, learning\_rate = 0.1, l2\_leaf\_reg = 3$)	Native handling of categorical features; CV=2-fold; performance $\sim$ 0.705, but training time doubled.	25
Trial 5: Filter = 250; Wrapper = 20; Model = LGBMClassifier ($n\_estimators = 20, num\_leaves = 4, learning\_rate = 0.1$)	Fast baseline with larger filter pool for greater feature diversity; CV=2-fold; performance $\sim$ 0.70 with much lower runtime.	20
Trial 6: Filter = 170; Wrapper = 20; Model=LGBMClassifier ($n\_estimators = 40, num\_leaves = 8, max\_depth = 6, learning\_rate = 0.05, min\_child\_samples = 50, min\_split\_gain = 0.01, reg\_alpha = 0.1, reg\_lambda = 0.2$)	Applied stronger split and L1/L2 regularization; CV=2-fold; performance $\sim$ 0.705 with saturation after $\sim$ 12 features.	20

Trial (Parameters)	Description and Purpose	# Features
Trial 7: Filter = 200; Wrapper = 20 (backward); Model = LGBMClassifier ($n\_estimators = 25, num\_leaves = 6, max\_depth = 5, learning\_rate = 0.1$)	Backward elimination from a rich 200-feature pool removed redundancy and achieved a compact, diverse 20-feature set; CV=2-fold; performance ~ 0.705.	20

Table 4: Feature Selection Trials

The trials performed are listed above in Table 4. The final trial 7 is also discussed below.

We began Trial 7 with the top 200 filtered features and let the algorithm iteratively remove features that contributed least to the model's performance (as measured by cross-validated AUC). This process was more computationally intensive but proved effective. The backward selection with LightGBM ultimately resulted in a diverse set of about 20 features that optimized performance. This approach likely succeeded because:

- It **removed redundant features**: Forward selection had shown that after 15 features, many additional features were marginal gains. Backward elimination, starting with a rich pool, could detect and drop those redundant or less useful features, especially if some features were highly correlated.
- It **found a better mix of feature types**: Forward methods tended to pick many similar features (for example, multiple day-of-week indicators early on). Backward selection, in contrast, began with a broad mix and tended to retain a more balanced portfolio of features (covering different aspects like amount patterns, time gaps, merchant risk, etc.), thereby capturing a wider range of signals.

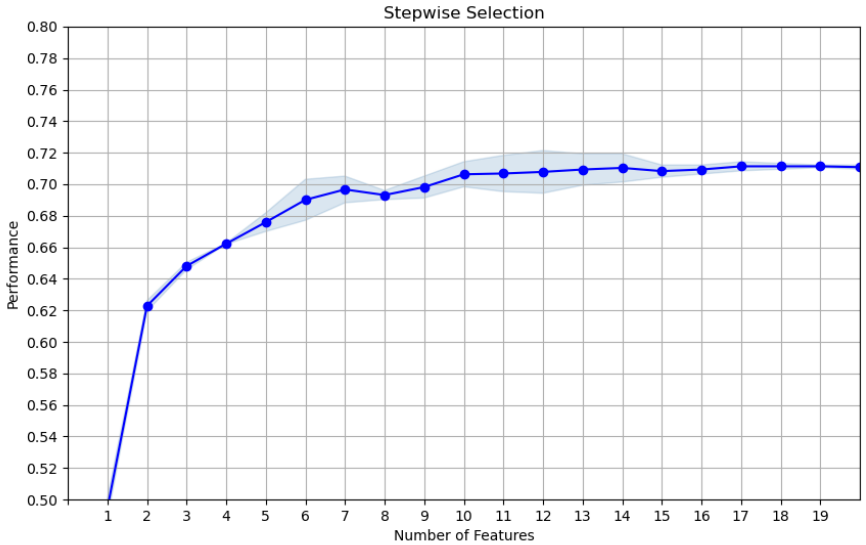


Figure 3: Performance curve for Trial 7.

The final selected feature set captured the key drivers of fraud detection in the data without including excessive noise. By reducing the feature count to around 20, we also simplified the model, which can improve generalizability and reduce computational cost. This selected subset was then used in all subsequent model training. In summary, the feature selection process indicated that a backward elimination approach (Trial 7's backward LGBM) was the most effective, as it achieved the highest validation performance (0.705 validation metric in that trial) with a manageable feature set, outperforming the forward selection trials. We adopted this as the final feature list for model development.

6 Preliminary Model Exploration

With the refined feature set, we explored a variety of modeling algorithms to determine which would be most effective for fraud detection. Our approach was to start with a spectrum of model types, from simple interpretable models to complex ensemble methods and even neural networks. Each model was trained on the training dataset and evaluated on a hold-out validation set (and via cross-validation) using relevant performance metrics.

The models we tried included:

- **Logistic Regression:** A linear model providing a baseline. We included regularization (penalty) to handle the large number of features. Logistic regression is easy to interpret but may not capture complex patterns.
- **Decision Tree:** A single decision tree model, which can capture non-linear feature interactions but tends to overfit if not pruned. It provides a simple flowchart-like reasoning but might be too simplistic alone for this problem.
- **k-Nearest Neighbors (KNN):** A distance-based model that labels a transaction based on the most similar past transactions. Given the high dimensionality and size of data, KNN was computationally expensive and not very effective (as expected, it struggled to differentiate fraud well in this large dataset).
- **LightGBM:** A gradient boosting decision tree model (from Microsoft's LightGBM library) which is optimized for speed and efficiency. We tried LightGBM due to its proven performance on structured data and its ability to handle large datasets and many features through boosting.
- **XGBoost:** Another popular gradient boosting library. XGBoost generally performs similarly to LightGBM; we included it to see if any differences in handling the data would yield improvements.
- **CatBoost:** A gradient boosting model particularly well-suited for categorical features (developed by Yandex). CatBoost can natively handle categorical variables and often requires less preprocessing for categories, which was appealing given our categorical fields (like state, merchant, etc.). We hypothesized CatBoost might excel due to this advantage.
- **Neural Network:** A multi-layer perceptron (feed-forward neural network) to see if a non-tree-based method could capture patterns. We used a relatively simple network architecture given the size of data, with careful tuning to avoid overfitting. Neural networks can, in theory, capture very complex patterns, but they require large amounts of data and careful tuning, and they are less interpretable.

Each model was evaluated primarily on its ability to accurately rank transactions by fraud risk. A key metric for us was the **fraud detection rate at a 3% review threshold** (i.e., what fraction of all frauds are caught if we investigate the top 3% highest-risk transactions as scored by the model). We also monitored the overall AUC as a general measure of classification performance, and precision (positive predictive value) at the chosen review rate to understand how many of the flagged transactions are actually fraud.

A summary of the model comparison is shown in Figure 4. The simpler models (logistic regression, single decision tree, KNN) served as useful baselines but showed considerably lower performance. The decision tree model performed slightly better than logistic, but it started overfitting the

Model	Parameters							Average FDR @ 3%		
	Iteration	criterion	splitter	max_depth	min_samples_split	min_samples_leaf	ccp_alpha	Train	Test	OOT
Decision Tree	1	gini	best	5	20	10	-	0.675	0.663	0.539
	2	entropy	random	17	70	50	-	0.665	0.651	0.546
	3	entropy	best	10	100	50	-	0.726	0.704	0.494
	4	gini	random	12	80	40	-	0.678	0.660	0.551
	5	gini	best	10	100	50	-	0.718	0.690	0.512
	6	gini	best	-	10	5	0.0001	0.719	0.695	0.523
	Iteration	sub_sample	learning_rate	n_estimators	max_depth	gamma	reg_lambda	Train	Test	OOT
XGBoost	1	-	0.1	100	6	10	10	0.742	0.728	0.576
	2	-	0.05	200	3	1	1	0.753	0.731	0.567
	3	-	0.05	30	7	5	5	0.748	0.729	0.582
	4	0.5	0.1	100	10	6	3	0.764	0.740	0.574
	5	0.6	0.01	100	12	8	3	0.733	0.720	0.579
	6	0.5	0.01	200	6	7	3	0.734	0.719	0.576
	Iteration	hidden_layer_sizes	activation	solver	alpha	learning_rate	max_iter	Train	Test	OOT
Multi-Layer Perceptron	1	(10,)	relu	adam	0	constant	150	0.681	0.666	0.559
	2	(10,5,2)	tanh	sgd	0	constant	250	0.665	0.658	0.548
	3	(100,20)	logistic	adam	0.01	adaptive	150	0.682	0.679	0.568
	4	(100,20,10,30)	relu	adam	0.1	invscaling	100	0.732	0.715	0.583
	5	(50,50)	relu	adam	0.1	constant	100	0.698	0.680	0.567
	6	(10,8,6,4,2)	relu	adam	0.01	constant	150	0.722	0.694	0.587
	Iteration	boosting_type		num_leaves	max_depth	learning_rate	n_estimators	Train	Test	OOT
LightGBM	1	gbdt		31	4	0.1	50	0.751	0.732	0.565
	2	dart		40	5	0.01	150	0.715	0.690	0.564
	3	gbdt		100	3	0.05	100	0.736	0.711	0.570
	4	gbdt		7	5	0.05	150	0.756	0.734	0.568
	5	gbdt		25	3	0.06	100	0.744	0.710	0.565
	6	dart		40	4	0.04	120	0.716	0.693	0.569
	Iteration	iterations		learning_rate	depth	l2_leaf_reg	border_count	Train	Test	OOT
Catboost	1	300		0.05	6	3	200	0.801	0.757	0.590
	2	400		0.02	7	4	300	0.781	0.751	0.581
	3	500		0.01	7	4	150	0.762	0.730	0.579
	4	1000		0.005	8	4	150	0.774	0.746	0.596
	5	200		0.01	9	6	120	0.733	0.702	0.578
	6	1000		0.006	8	5	200	0.785	0.741	0.589

Figure 4: Model Exploration

training data and still missed a lot of fraud. KNN underperformed due to the challenges mentioned (low performance and impractical runtime for deployment).

The gradient boosting models (LightGBM, XGBoost, and CatBoost) outperformed the simple models. This was a substantial improvement, confirming that ensemble tree methods are well-suited to this problem. CatBoost emerged as the best model in our tests, slightly outperforming LightGBM and XGBoost. CatBoost achieved the highest performance on OOT data capturing about 60% of frauds. The performance edge of CatBoost may be attributed to its handling of categorical variables and possibly better default handling of class imbalance through its loss function. The neural network model showed decent performance but did not surpass CatBoost, and it was more time-consuming to train and tune.

Given these results, we narrowed our final model candidates to the best-performing ones, with CatBoost being the frontrunner. The consistency of CatBoost's performance and its pragmatic advantages (no need for elaborate encoding of categorical fields, relatively fast prediction time, and robust results) made it an attractive choice. XGBoost was a close second and remained a strong backup option. The performance of the models on training, test and OOT data is presented in Figure 5.

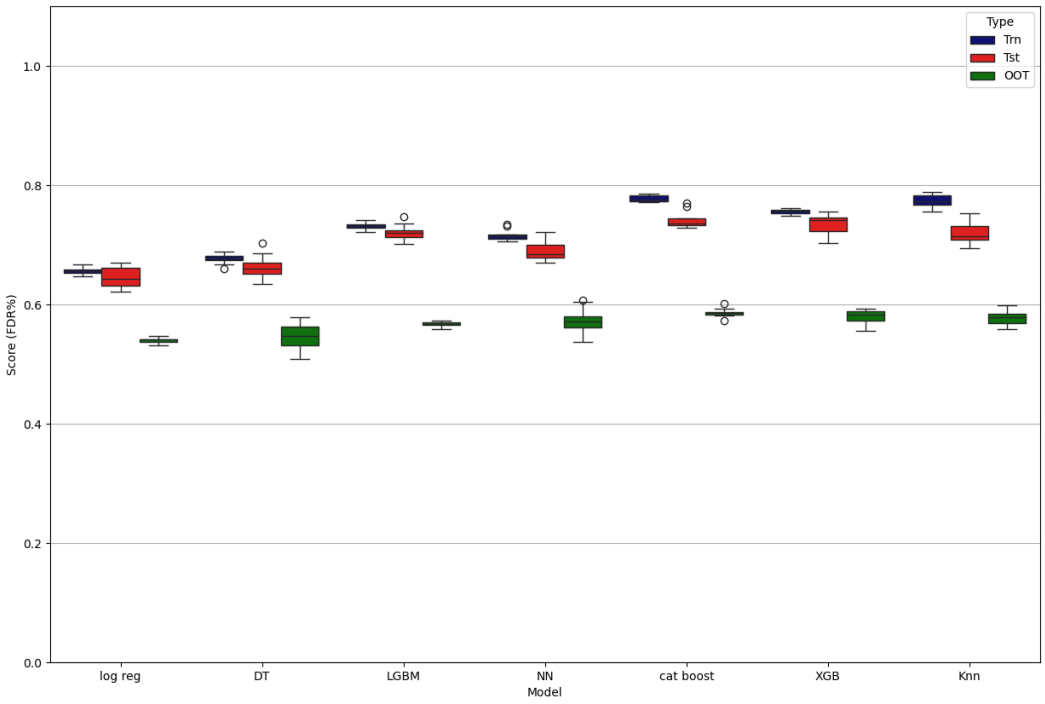


Figure 5: Model Performance Comparison

7 Final Model Performance

Based on the model exploration, we selected **CatBoost** as the final model for deployment. We proceeded to train a CatBoost classifier using the full training dataset (with the selected features), tuning hyperparameters via cross-validation.

The final model selected is a `CatBoostClassifier` with the following configuration:

- **Iterations:** 500
- **Learning rate:** 0.01
- **Depth:** 8
- **L2 leaf regularization:** 4
- **Border count:** 150
- **Random seed:** 42

We found that using a moderate amount of regularization and a slightly higher weight on the fraud class helped maximize the recall of fraudulent cases without too many false positives.

After training the final model, we evaluated its performance on three sets:

7.1 Performance on Training Set

Training set: To ensure the model learned well (though training performance is optimistic, it's useful to check no underfitting).

Training	# Records	# Goods	# Bads	Fraud Rate										
	59780	58299	1481	0.025403523										
	Bin Statistics					Cumulative Statistics							Fraud Savings / Loss		
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR	Fraud Savings	FP Loss	Overall Savings
1	598	37	561	6.19%	93.81%	598	37	561	0.06%	37.88%	37.82	0.07	224400	740	223660
2	598	172	426	28.76%	71.24%	1196	209	987	0.36%	66.64%	66.29	0.21	394800	4180	390620
3	597	432	165	72.36%	27.64%	1793	641	1152	1.10%	77.79%	76.69	0.56	460800	12820	447980
4	598	514	84	85.95%	14.05%	2391	1155	1236	1.98%	83.46%	81.48	0.93	494400	23100	471300
5	598	563	35	94.15%	5.85%	2989	1718	1271	2.95%	85.82%	82.87	1.35	508400	34360	474040
6	598	575	23	96.15%	3.85%	3587	2293	1294	3.93%	87.37%	83.44	1.77	517600	45860	471740
7	598	581	17	97.16%	2.84%	4185	2874	1311	4.93%	88.52%	83.59	2.19	524400	57480	466920
8	597	576	21	96.48%	3.52%	4782	3450	1332	5.92%	89.94%	84.02	2.59	532800	69000	463800
9	598	587	11	98.16%	1.84%	5380	4037	1343	6.92%	90.68%	83.76	3.01	537200	80740	456460
10	598	591	7	98.83%	1.17%	5978	4628	1350	7.94%	91.15%	83.22	3.43	540000	92560	447440
11	598	592	6	99.00%	1.00%	6576	5220	1356	8.95%	91.56%	82.61	3.85	542400	104400	438000
12	598	590	8	98.66%	1.34%	7174	5810	1364	9.97%	92.10%	82.13	4.26	545600	116200	429400
13	597	591	6	98.99%	1.01%	7771	6401	1370	10.98%	92.51%	81.53	4.67	548000	128020	419980
14	598	595	3	99.50%	0.50%	8369	6996	1373	12.00%	92.71%	80.71	5.10	549200	139920	409280
15	598	593	5	99.16%	0.84%	8967	7589	1378	13.02%	93.05%	80.03	5.51	551200	151780	399420
16	598	594	4	99.33%	0.67%	9565	8183	1382	14.04%	93.32%	79.28	5.92	552800	163660	389140
17	598	591	7	98.83%	1.17%	10163	8774	1389	15.05%	93.79%	78.74	6.32	555600	175480	380120
18	597	591	6	98.99%	1.01%	10760	9365	1395	16.06%	94.19%	78.13	6.71	558000	187300	370700
19	598	591	7	98.83%	1.17%	11358	9956	1402	17.08%	94.67%	77.59	7.10	560800	199120	361680
20	598	593	5	99.16%	0.84%	11956	10549	1407	18.09%	95.00%	76.91	7.50	562800	210980	351820

Figure 6: Final CatBoost model performance on the training dataset.

7.2 Performance on Independent Test Set

Independent test set: A hold-out portion from the same time period as training (but not seen during training or validation) to evaluate generalization on in-time data.

Testing	# Records	# Goods	# Bads	Fraud Rate
	5124	4510	614	0.136141907

	Bin Statistics					Cumulative Statistics							Fraud Savings / Loss		
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR	Fraud Savings	FP Loss	Overall Savings
1	256	33	223	12.89%	87.11%	256	33	223	0.13%	34.05%	33.91	0.15	89200	660	88540
2	256	66	190	25.78%	74.22%	512	99	413	0.40%	63.05%	62.66	0.24	165200	1980	163220
3	257	180	77	70.04%	29.96%	769	279	490	1.12%	74.81%	73.69	0.57	196000	5580	190420
4	256	228	28	89.06%	10.94%	1025	507	518	2.03%	79.08%	77.05	0.98	207200	10140	197060
5	256	237	19	92.58%	7.42%	1281	744	537	2.98%	81.98%	79.00	1.39	214800	14880	199920
6	256	243	13	94.92%	5.08%	1537	987	550	3.95%	83.97%	80.02	1.79	220000	19740	200260
7	256	250	6	97.66%	2.34%	1793	1237	556	4.95%	84.89%	79.93	2.22	222400	24740	197660
8	257	248	9	96.50%	3.50%	2050	1485	565	5.95%	86.26%	80.31	2.63	226000	29700	196300
9	256	253	3	98.83%	1.17%	2306	1738	568	6.96%	86.72%	79.76	3.06	227200	34760	192440
10	256	247	9	96.48%	3.52%	2562	1985	577	7.95%	88.09%	80.14	3.44	230800	39700	191100
11	256	251	5	98.05%	1.95%	2818	2236	582	8.96%	88.85%	79.90	3.84	232800	44720	188080
12	256	255	1	99.61%	0.39%	3074	2491	583	9.98%	89.01%	79.03	4.27	233200	49820	183380
13	257	256	1	99.61%	0.39%	3331	2747	584	11.00%	89.16%	78.16	4.70	233600	54940	178660
14	256	248	8	96.88%	3.13%	3587	2995	592	12.00%	90.38%	78.38	5.06	236800	59900	176900
15	256	254	2	99.22%	0.78%	3843	3249	594	13.01%	90.69%	77.67	5.47	237600	64980	172620
16	256	249	7	97.27%	2.73%	4099	3498	601	14.01%	91.76%	77.74	5.82	240400	69960	170440
17	256	252	4	98.44%	1.56%	4355	3750	605	15.02%	92.37%	77.35	6.20	242000	75000	167000
18	257	254	3	98.83%	1.17%	4612	4004	608	16.04%	92.82%	76.79	6.59	243200	80080	163120
19	256	256	0	100.00%	0.00%	4868	4260	608	17.06%	92.82%	75.76	7.01	243200	85200	158000
20	256	250	6	97.66%	2.34%	5124	4510	614	18.07%	93.74%	75.68	7.35	245600	90200	155400

Figure 7: Final CatBoost model performance on the in-time test dataset.

7.3 Performance on Out-of-Time (OOT) Set

Out-of-Time (OOT) validation set: The final two months of 2010 data that were completely held out from any model development. This simulates performance on truly future data, testing the model's stability over time.

OOT	# Records	# Goods	# Bads	Fraud Rate
	2527	2197	330	0.150204825

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	126	22	104	17.46%	82.54%	126	22	104	0.18%	29.21%	29.03	0.21
2	127	52	75	40.94%	59.06%	253	74	179	0.60%	50.28%	49.68	0.41
3	126	91	35	72.22%	27.78%	379	165	214	1.34%	60.11%	58.77	0.77
4	126	109	17	86.51%	13.49%	505	274	231	2.23%	64.89%	62.66	1.19
5	127	113	14	88.98%	11.02%	632	387	245	3.15%	68.82%	65.67	1.58
6	126	115	11	91.27%	8.73%	758	502	256	4.09%	71.91%	67.82	1.96
7	127	112	15	88.19%	11.81%	885	614	271	5.00%	76.12%	71.12	2.27
8	126	113	13	89.68%	10.32%	1011	727	284	5.92%	79.78%	73.86	2.56
9	126	118	8	93.65%	6.35%	1137	845	292	6.88%	82.02%	75.14	2.89
10	127	120	7	94.49%	5.51%	1264	965	299	7.86%	83.99%	76.13	3.23
11	126	121	5	96.03%	3.97%	1390	1086	304	8.84%	85.39%	76.55	3.57
12	126	120	6	95.24%	4.76%	1516	1206	310	9.82%	87.08%	77.26	3.89
13	127	120	7	94.49%	5.51%	1643	1326	317	10.80%	89.04%	78.25	4.18
14	126	124	2	98.41%	1.59%	1769	1450	319	11.81%	89.61%	77.80	4.55
15	127	126	1	99.21%	0.79%	1896	1576	320	12.83%	89.89%	77.05	4.93
16	126	124	2	98.41%	1.59%	2022	1700	322	13.84%	90.45%	76.61	5.28
17	126	123	3	97.62%	2.38%	2148	1823	325	14.84%	91.29%	76.45	5.61
18	127	127	0	100.00%	0.00%	2275	1950	325	15.88%	91.29%	75.41	6.00
19	126	122	4	96.83%	3.17%	2401	2072	329	16.87%	92.42%	75.54	6.30
20	126	125	1	99.21%	0.79%	2527	2197	330	17.89%	92.70%	74.81	6.66

Figure 8: Final CatBoost model performance on the out-of-time holdout dataset.

Finally, on the out-of-time (OOT) data, which is the true measure of future performance, the model achieved an FDR@3% of **60.1%**. In practical terms, this means if the model flags the top 3% of transactions in a future period, it would capture about 60% of all the fraud in that period. This OOT FDR of 0.6011 (60.11%) was a key success metric for the project, as it exceeded our initial targets and showed that the model's performance held up on new data.

Beyond these threshold-based metrics, we also examined the score distribution and calibration. The model outputs a risk score (probability of fraud) for each transaction. We found that fraudulent transactions had significantly higher scores on average than genuine ones, indicating good separation. The Kolmogorov-Smirnov (KS) statistic between fraud and non-fraud score distributions was high, further validating the model's discriminatory power.
Given these performance results, we are confident in the model's ability to identify fraud efficiently.

8 Financial Analysis and Cutoff Recommendation

From a business perspective, catching fraud is only one part of the equation—the model must also be cost-effective. Investigating transactions has an associated operational cost (e.g., manual review time, contacting customers, etc.), so we conducted a financial analysis to determine the optimal balance between frauds caught and false positives (non-fraud flagged).

We plotted key curves to analyze the model's financial impact across different score cutoffs:

- **Fraud Capture vs. Review Rate Curve:** This curve shows the percentage of total fraud captured (y-axis) as the percentage of transactions reviewed increases (x-axis). It starts at 0% fraud caught at 0% reviewed and goes up to 100% fraud caught if 100% transactions were reviewed. Our model's curve rises steeply initially, indicating that even a small review percentage yields a large fraction of fraud caught.

- **Cost vs. Benefit Curve:** We translated fraud caught and false positives into monetary terms. We assumed an average loss of $400 per fraudulent transaction (if not stopped) and an average investigation cost of $20 per non-fraud transaction reviewed (based on operational estimates). Using these, we calculated the "fraud savings" (dollars saved by preventing fraud) and "review costs" for each potential cutoff. This curve helps identify where net savings is maximized.

- **Net Savings vs. Threshold Curve:** Combining the above, we plotted the net financial benefit (fraud losses prevented minus review costs) as a function of the score cutoff or equivalent review rate. This is directly used to find the optimal threshold from a dollar perspective.

As shown in Figure 9, the net savings (green curve) initially increases as we raise the review rate—because we catch significantly more fraud—until it reaches a peak, after which it starts to level off or decline when the cost of reviewing additional transactions begins to outweigh the incremental fraud caught. In our analysis, the net savings peaked at reviewing around 10–12% of transactions (in purely monetary terms, that would maximize dollars saved). However, reviewing 10%+ of all transactions is not operationally feasible due to resource constraints (and it would involve a very high false positive volume). The bank's fraud operations team can realistically investigate only a few percent of transactions.

Considering this, we looked at lower review rates. We found that at about 5% review yields substantial net savings ($53 million annually) while keeping the workload manageable. If we were to increase the review rate beyond that, the marginal gain in fraud caught would start to diminish, and the number of false positives would increase steeply, straining the investigation team and potentially impacting customer experience.

Therefore, we **recommend setting the model score cutoff to investigate approximately 5% of transactions**. This cutoff is a balance between high fraud capture and practical constraints:

- It delivers a very strong fraud detection rate (around 60% of fraud prevented).

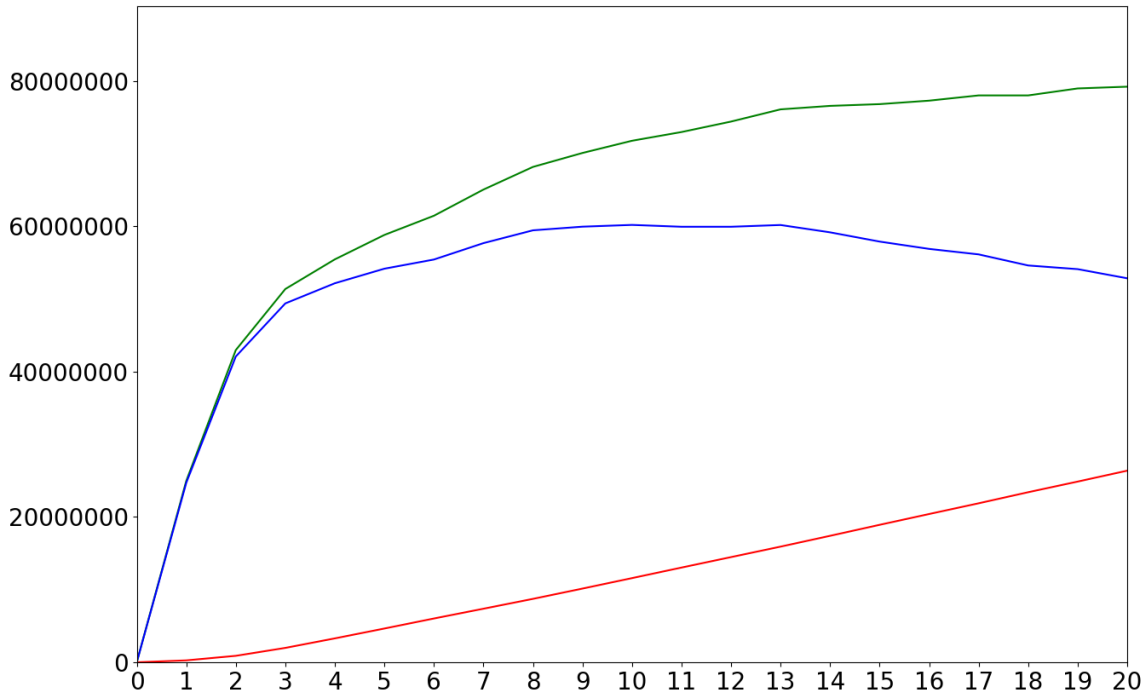


Figure 9: *Projected annual savings (fraud loss prevented minus investigation cost) versus model score cutoff (or equivalently, percentage of transactions reviewed). The green line represents net savings, the blue line represents cumulative fraud loss prevented, and the red line represents review cost.*

- It keeps false positives at a tolerable level.
- It aligns with the current capacity of the fraud operations team.

At this threshold, the expected annual fraud savings is about $53 million, net of investigation costs. This is a considerable financial win, achieved by stopping fraud losses early. It's worth noting that if the bank later expands its fraud review capacity or the cost structure changes, we can revisit the threshold. The curves plotted can guide such decisions—for example, if more reviewers are available, one might move to 8% review to catch extra fraud, if the net benefit is still positive.

9 Summary

This project encompassed the full data analytics pipeline for a credit card fraud detection model, from data understanding to deployment recommendations. We began with a rich dataset of transactions and carefully prepared it—addressing quality issues like missing data and outliers, and enhancing it with numerous derived features capturing customer behavior and transaction patterns. Through an exhaustive feature selection process, we distilled the most informative features to feed into our models.

We tried a wide range of modeling approaches, ultimately finding that modern ensemble methods, particularly CatBoost, delivered superior performance in identifying fraudulent transactions. The final CatBoost model was able to generalize well to new data, as evidenced by its out-of-time performance: it can intercept about 60% of fraud cases. In practical terms, this means far more fraudulent charges can be stopped with far fewer false alarms than previous methods, translating to an estimated $53 million in annual fraud loss reduction for the bank.

Beyond model accuracy, we evaluated the solution in terms of operational impact and financial benefit. We determined an optimal operating point for the model that balances fraud capture with review effort, recommending a threshold that aligns with business constraints and maximizes net savings.

Overall, the project demonstrates a successful deployment of machine learning in a high-stakes domain. We have delivered a fraud detection model that is not only statistically effective but also actionable for the business. Moving forward, we suggest a few areas for future improvement:

- **Adaptive learning:** Fraud patterns evolve, so retraining or updating the model periodically (e.g., monthly or quarterly) with new data will ensure it adapts to emerging fraud tactics.
- **Customer impact analysis:** Further analysis on the false positive cases (legitimate transactions flagged) could be done to refine rules on top of the model (for instance, auto-decline high-score transactions but auto-approve some transactions even if high-score if they match the customer's usual pattern like monthly bill payments, etc., to reduce customer inconvenience).

In conclusion, the fraud analytics project has delivered a robust model and clear business recommendations. By implementing this model, we can significantly enhance fraud prevention capability, save tens of millions in fraud losses, and improve the efficiency of its fraud investigation team. This project showcases the value of data-driven solutions in tackling financial crime and sets the stage for continued innovation in this space.

Appendix

Data Quality Report

Section 1 – Data Overview

This dataset contains credit card transaction records, covering the period from **January 1, 2010** to **December 31, 2010** (12 months). The data consists of **98,393 transaction records** across 10 fields, including transaction details (including credit card number, datetime information and transaction type), merchant information and fraud indicators. The dataset appears to track purchases and potentially other transaction types, with approximately **2.53%** of transactions flagged as **fraudulent**.

Section 2 – Field Summary Tables

Categorical Fields:

Field Name	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Date	98,393	100.00%	0	365	2/28/10
Merchnum	94,970	96.52%	0	13091	930090121224
Merch description	98,393	100.00%	0	13126	GSA-FSS-ADV
Merch state	97,181	98.77%	0	227	TN
Transtype	98,393	100.00%	0	4	P
Recnum	98,393	100.00%	0	98393	1
Fraud	98,393	100.00%	95,901	2	0 (Not Fraud)
Merch zip	93,664	95.19%	0	4567	38118
Cardnum	98,393	100.00%	0	1645	5142148452

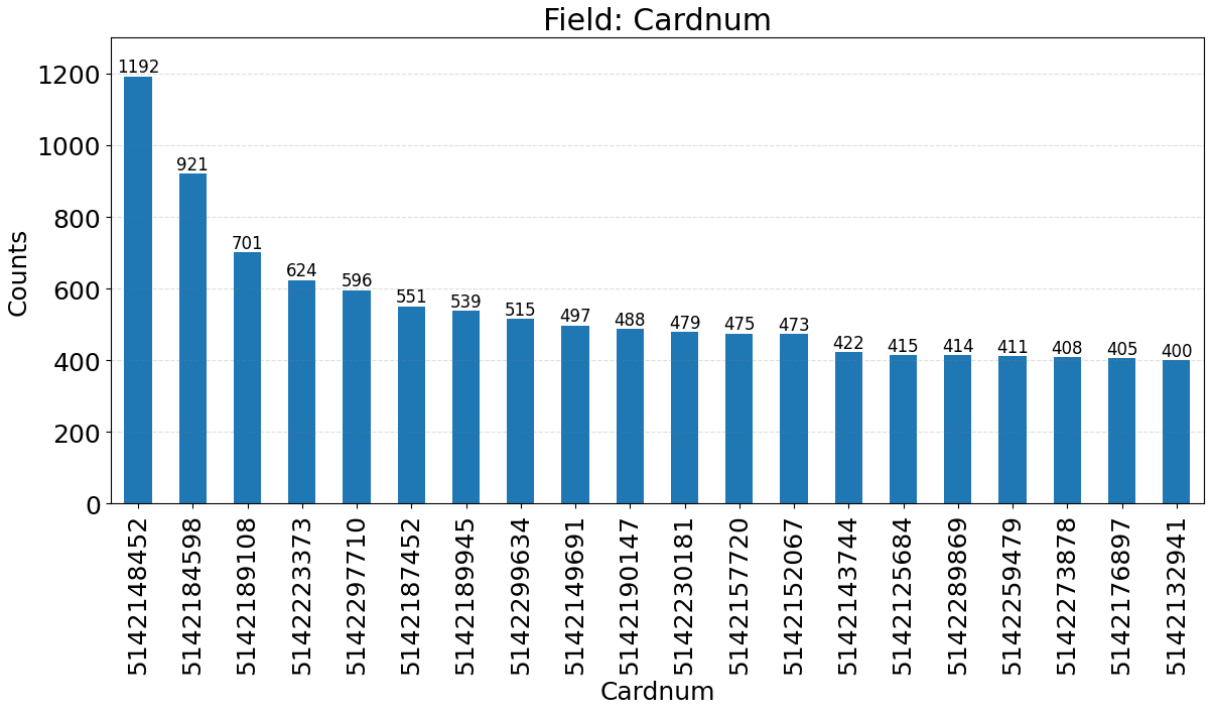
Numeric Fields:

Field Name	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Std. Dev.	Most Common
Amount	98,393	100.00%	0	0.01	3,102,045.53	424.290926	9,922.44	3.62

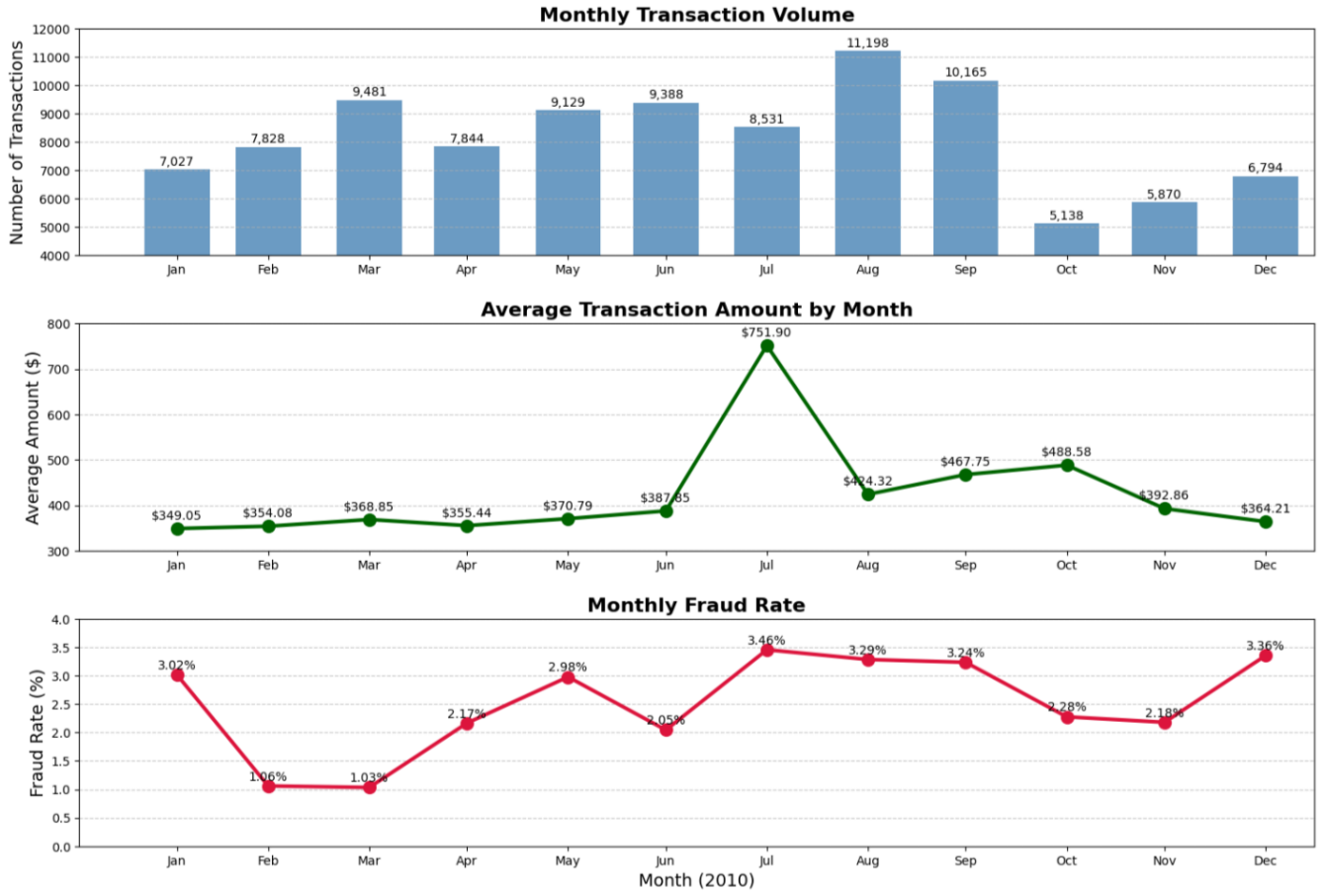
Section 3 – Field Analysis

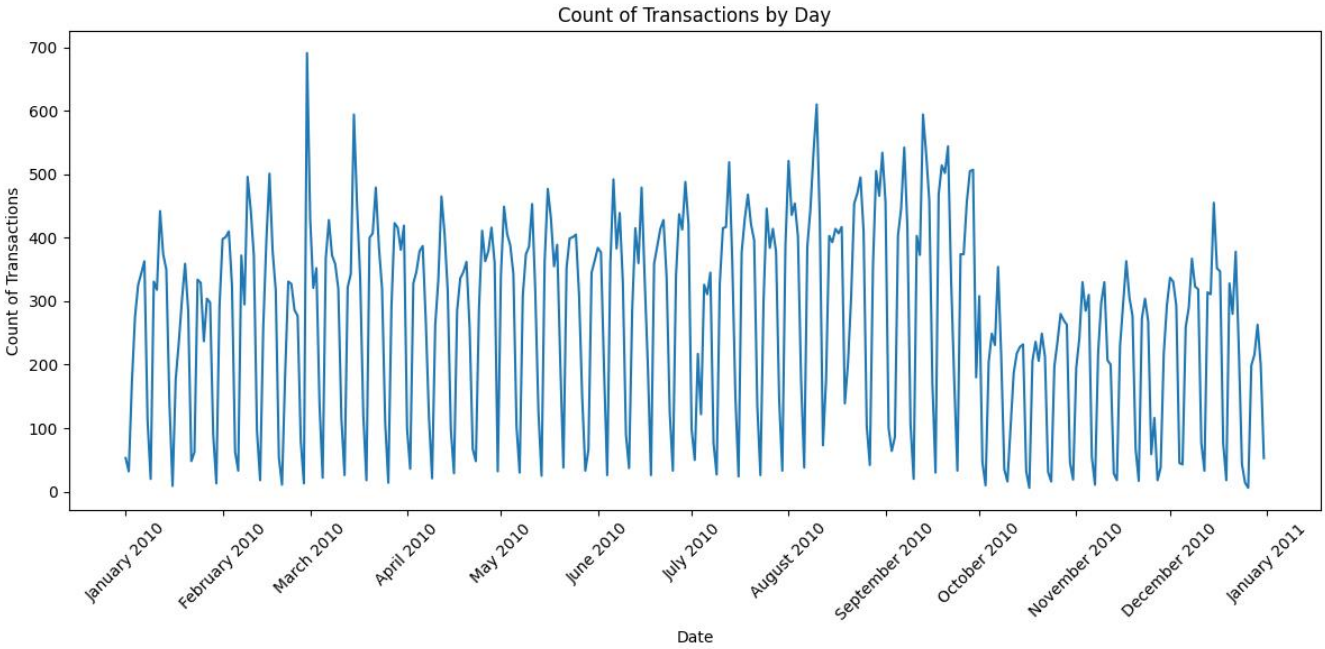
1. **Recnum**: The Recnum field serves as a sequential primary key identifier, ranging from 1 to 98,393, with each record having a unique value. This confirms that each transaction in the dataset is uniquely identifiable with no duplicates. While technically a numeric value, it functions as an index rather than a measurable quantity, making it more appropriate to treat as a categorical field for analytical purposes.

2. **Cardnum:** Credit card numbers are unique identifiers. Although they consist only of digits, you wouldn't add or average them. They should be represented as categorical (or as a string) rather than numeric. The most frequent card (5142148452) appears in multiple transactions, indicating repeated use by the same cardholder. With 1,645 unique card numbers across 98,393 transactions, the average cardholder has approximately 60 transactions in the dataset. This field is crucial for identifying patterns of card usage and potential clustering of fraudulent activities across specific cards.

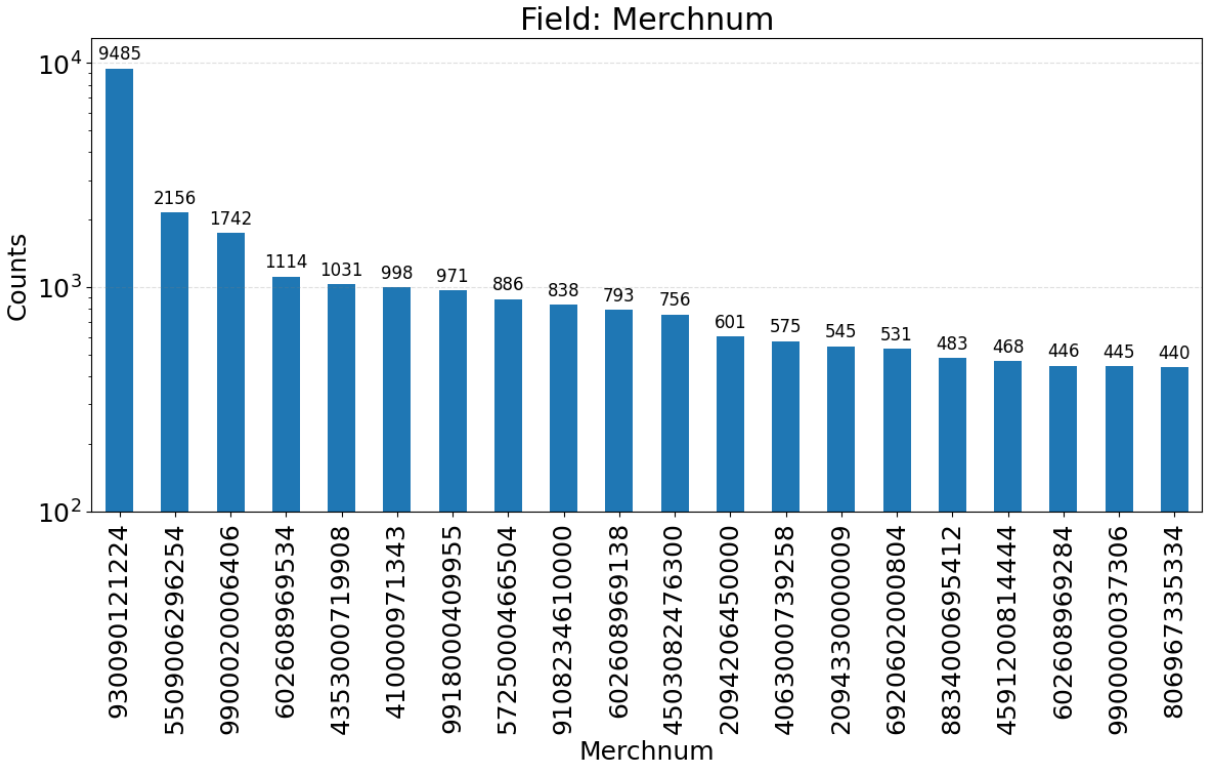


3. **Date:** Dates have an inherent order and can be converted to a numerical timestamp if the analysis requires time series computations. However, dates usually are handled with date-time tools or by extracting components (month, day, year). They are not "numbers" in the arithmetic sense, so we may treat the raw date field as categorical (or, more precisely, as a date variable). The Date field captures transaction dates in MM/DD/YY format, spanning from January 1, 2010, to December 31, 2010. The time-series visualizations reveal distinct seasonal patterns, with August and September showing the highest transaction volumes (11,198 and 10,165 transactions respectively). There's a notable drop in transaction volume during October, which could indicate a seasonal business cycle or data collection anomaly. The average transaction amounts show a dramatic spike in July ($751.90), nearly twice the monthly average, suggesting large seasonal purchases or potentially suspicious activity during this period. The monthly fraud rate fluctuates between 1.03% and 3.89%, with peaks in January, May, July, and December, potentially indicating seasonal fraud patterns around holidays.

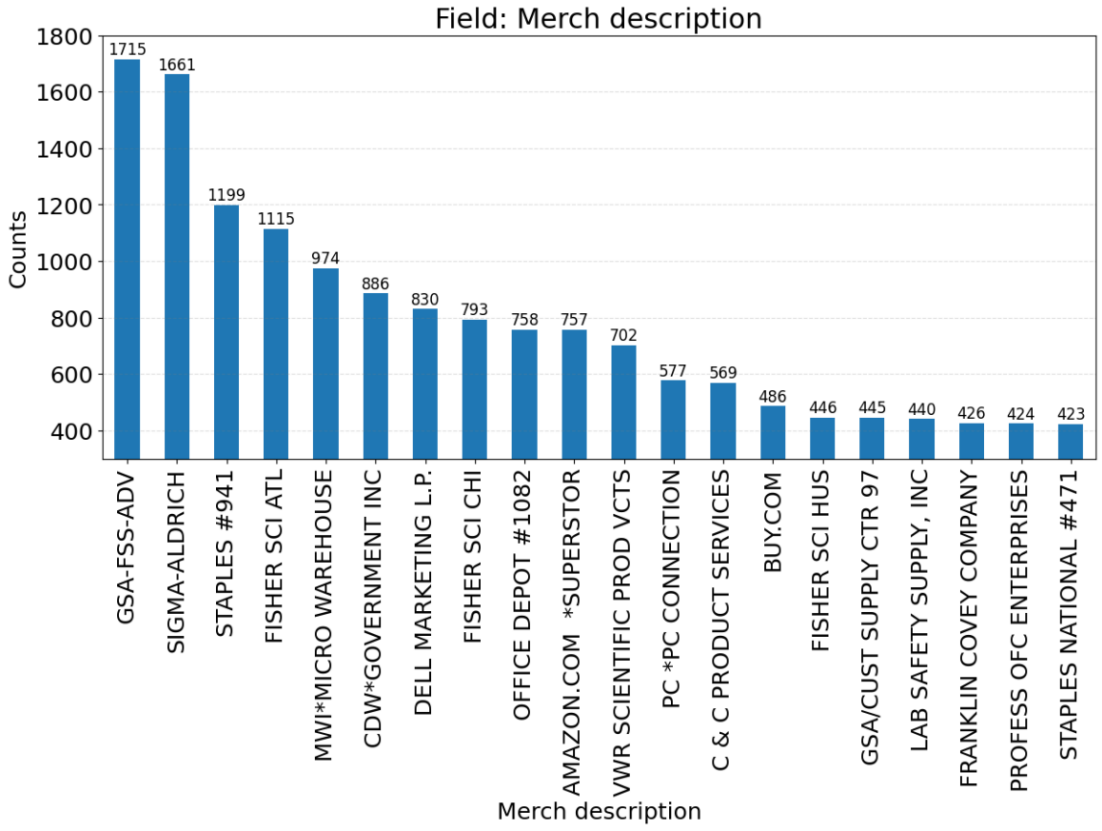




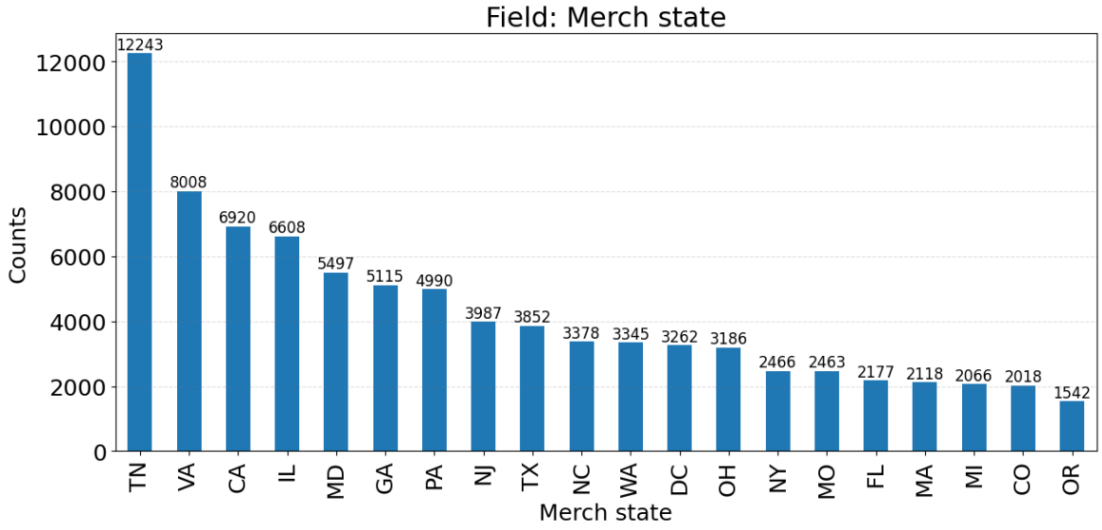
4. **Merchnum:** This field, which likely represents a unique merchant number, is an identifier. Even though it might be stored numerically, the number is a code, not something you would add or average. Thus, it should be treated as categorical. The distribution is highly skewed, with the top merchant (930090121224) accounting for 9,485 transactions (approximately 10% of all transactions). This concentration suggests a dominant relationship with specific merchants. The logarithmic scale in the visualization highlights the long-tail distribution characteristic of merchant activity.



5. **Merch description:** This is a text field that describes the merchant. It is inherently categorical. The Merch description field provides text descriptions of merchants, with 13,126 unique descriptions across all 98,393 records. The most common merchants are "GSA-FSS-ADV" (1,715 transactions) and "SIGMA-ALDRICH" (1,661 transactions). There is a diversity of merchants which provides rich contextual information for understanding transaction patterns.

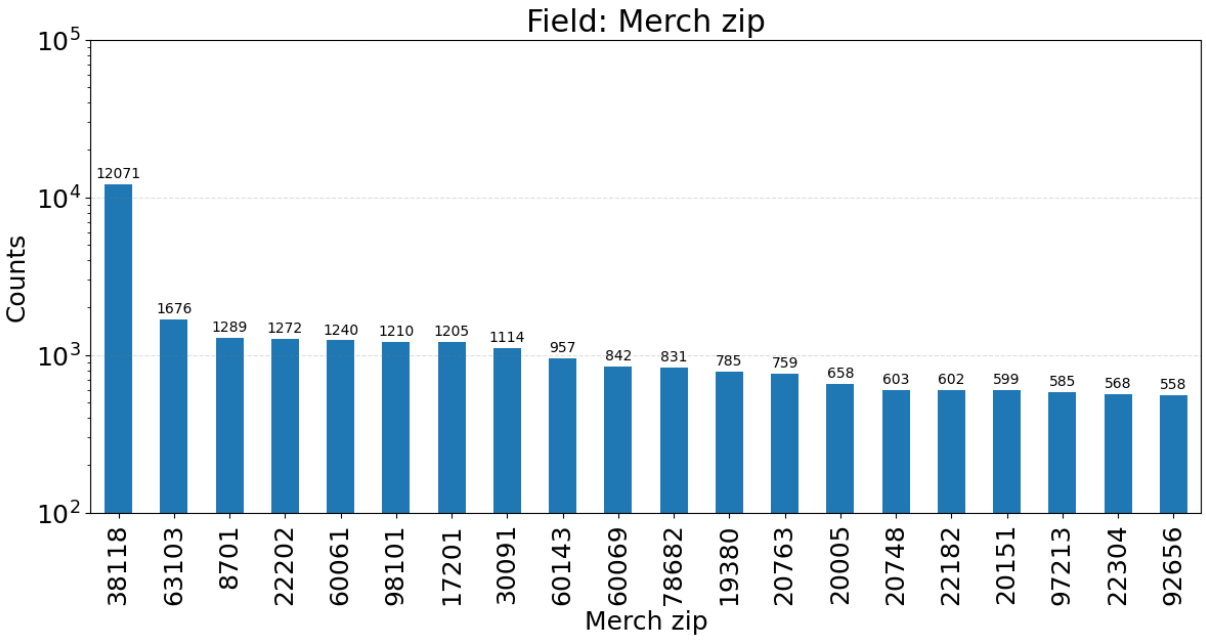


6. **Merch state:** Merch state indicates the U.S. state where merchants are located, with 227 unique values across 97,181 records (98.77% of transactions). Tennessee (TN) dominates the distribution

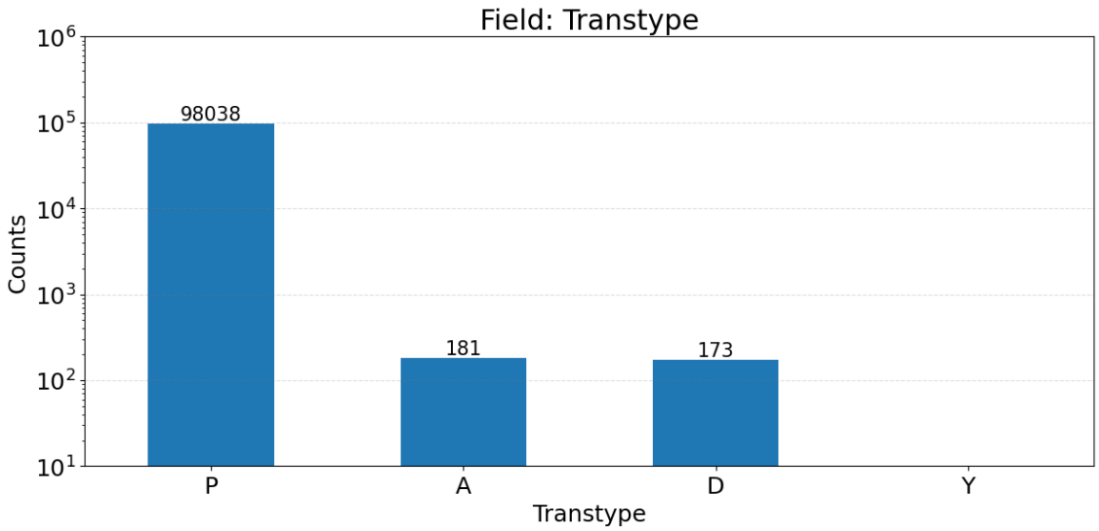


with 12,243 transactions, followed by Virginia (VA) with 8,008 and California (CA) with 6,920. This strong geographic concentration suggests the cardholder organization likely has significant operations or relationships in these states. The presence of 227 unique values (exceeding the 50 U.S. states plus territories) indicates potential data quality issues, such as non-standard state codes or international locations being captured in this field. The geographic distribution could help identify regional fraud patterns or unusual transaction locations that might indicate compromised cards.

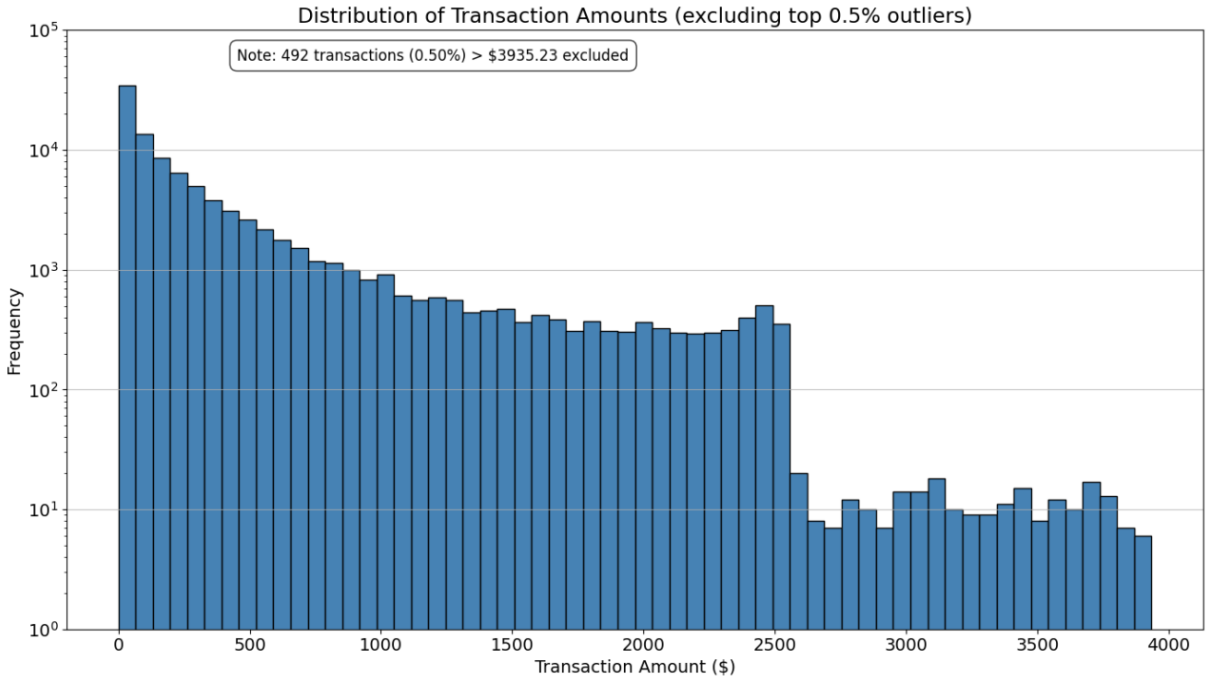
7. **Merch zip:** Merch zip contains ZIP codes for merchant locations, populated in 93,664 records (95.19% of transactions) with 4,567 unique values. The distribution shows extremely high concentration in certain ZIP codes, with 38118 (Memphis, TN) accounting for 12,071 transactions (nearly 13% of all transactions with ZIP codes). This corresponds with the high frequency of Tennessee in the Merch state field. The log-scale visualization reveals clusters of activity in specific regions, providing granular geographic information beyond state-level analysis. The 4.81% of missing ZIP codes, combined with the 1.23% of missing state information, suggests some geographic data quality issues that might impact location-based fraud analysis.



8. **Transtype:** Transtype categorizes the type of transaction with four unique values across all 98,393 records. It is a label and should be treated as categorical. The overwhelming majority (98,038 or 99.6%) are coded as "P" (likely indicating Purchases), with minimal occurrences of "A" (181 transactions) and "D" (173 transactions), and a single "Y" transaction. The extreme imbalance in this distribution indicates the dataset predominantly captures purchase transactions rather than other activities like refunds, cash advances, or balance transfers. Understanding the meaning of these codes is essential for transaction analysis, as different transaction types carry different risk profiles and behavioral patterns. The rare transaction types ("A", "D", and "Y") might require special attention in fraud analysis as unusual patterns.



9. **Amount:** Amount represents the monetary value of each transaction, ranging from $0.01 to $3,102,045.53. It is a continuous value on which calculations (sums, averages, etc.) are meaningful; therefore, it should be numeric. The distribution is heavily right-skewed, with 99.5% of transactions below $3,935.23 (the remaining 0.5% being extreme outliers). The log-scale visualization shows distinct transaction behavior patterns, with a sharp decline after $2,500, potentially reflecting authorization limit policies or natural spending thresholds. The most common transaction amount is $3.62, occurring in numerous transactions and likely representing a standard fee or small purchase. Large transactions significantly skew the average amount upward. The presence of multi-million dollar transactions requires some investigation.



10. **Fraud:** Although Fraud is recorded as 0 or 1, it is a binary indicator (whether the transaction is fraudulent or not). This field is used as the target variable in classification. It can be stored as numeric (binary) for some algorithms, but conceptually it is categorical because the "0" and "1" represent classes rather than measurable quantities. Only 2,492 transactions (2.53%) are flagged as fraudulent, creating a significant class imbalance typical in fraud detection problems. The time-series visualization of fraud counts by day reveals temporal patterns, with several significant fraud spikes in January, July, and August 2010. These spikes could indicate coordinated fraud attacks or periods of enhanced fraud detection activity. Daily fraud counts range from 0 to over 50 transactions, suggesting considerable volatility in fraud occurrence.