PROJECT REPORT

Final Fraud Analytics assignment: Fraud detection for credit card transactions

Facundo Klappenbach

Table of Contents

Executive Summary

Business Description

This project focuses on the development and implementation of a fraud detection model designed to protect the organization's financial assets by identifying fraudulent transactions with high accuracy and minimal impact on legitimate activity. Using advanced data analytics, the model was trained and tested to balance fraud detection with cost efficiency, minimizing false positives while maximizing fraud savings. The solution was calibrated to perform optimally at a fraud detection rate (FDR) of 4% on out-of-time (OOT) data, representing a robust and scalable approach for ongoing fraud risk management.

Statement of Business Impact

The fraud detection model is expected to deliver substantial financial benefits to the company by reducing fraud-related losses. Based on the applied FDR threshold, the model is projected to save approximately $42 million annually by preventing fraudulent transactions while minimizing revenue loss from false positives. This outcome provides a solid foundation for both immediate and future savings, supporting the organization's commitment to efficient and proactive fraud management. By aligning the fraud detection strategy with financial objectives, this project reinforces the organisation's resilience against fraud and enhances overall operational profitability.

Data description

Overview of data

The dataset is **Card Transactions**, which contains **Transactions** of credit cards that occurred in 2010. The data came from a **US government Organization** in Tennessee. There are **10 fields** and **97,852 records**

Statistics tables

Numeric fields table

Field Name	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Stdev	Most Common
Amount	97,852	100.00%	0	0.01	310,2045.53	425.47	9,949.8	3.62

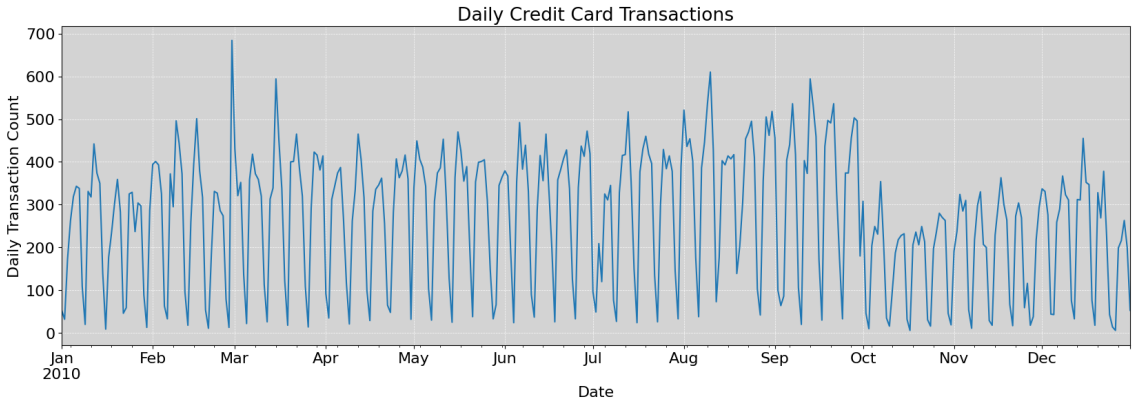
Categorical fields table

Field Name	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Date	97,852	100.0%	0	365	2/28/10
Merchnum	94,455	96.5%	0	13,091	930090121224
Merch description	97,852	100.0%	0	13,126	GSA-FSS-ADV
Merch state	96,649	98.8%	0	227	TN
Transtype	97,852	100.0%	0	4	P
Recnum	97,852	100.0%	0	97,852	1
Fraud	97,852	100.0%	95,805	2	0
Cardnum	97,852	100.0%	0	1,645	5142148452
Merch zip	93,149	95.2%	0	4,567	38118

Distributions examples

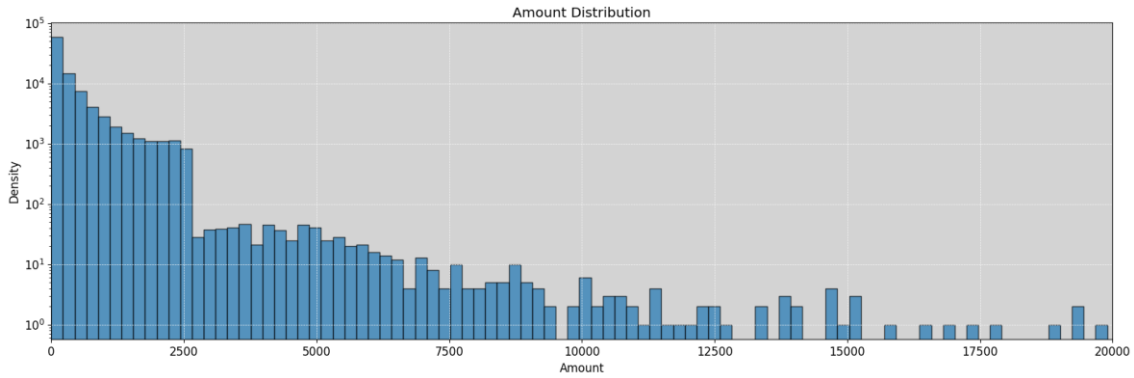
Transactions accros time:

The distribution shows the number of daily transactions across time, with a clearly decreased number after October, which can be due to the fiscal year starting that same month.



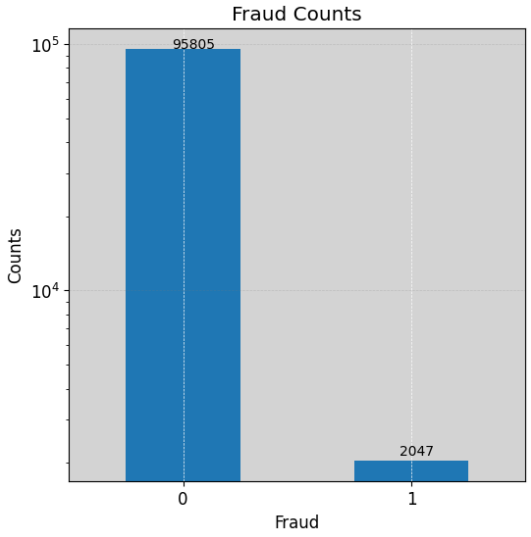
Transactions' amount:

The distribution shows the histogram of Transactions' Amount. There is a discontinuity in the count of transactions with transaction amounts between 2,500 to 3,000. Probably due to limits on the maximum amount of transactions on credit cards.



Fraud distribution:

Fraud identification label. Fraud = 0 (Not fraudulent), Fraud =1 (Fraud identified). The total count of fraud_label = 0 is 95,805. The total count of fraud_label = 1 is 2,047.



Data cleaning

Exclusions

In the Exclusions step, we remove irrelevant data to keep the dataset focused on transactions relevant to fraud detection, such as excluding non-purchase transactions.

Upon reviewing the transaction data, we identified four distinct transaction types: P (97,497 instances), A (181), D (173), and Y (1). The vast majority of transactions are type P. While the exact meanings of these types are unclear, our business manager speculates that type P likely represents purchases, while the others may indicate authorizations without purchases, declines, or similar non-purchase actions.

To maintain consistency and focus on completed purchase transactions, we were advised to exclude all records other than type P. This exclusion ensures that the dataset is aligned with our project objectives, allowing our analysis to more accurately reflect actual purchase behavior.

With this technique, we excluded 355 records. From 97,852 to 97,497

Outliers

Outliers are unusual data points that could distort results. They're examined to see if they represent fraud or legitimate transactions, and may be removed if unrepresentative.

During the initial data analysis, we identified a transaction outlier with an amount of $3,000,000—an unusually high figure compared to the next highest transaction, which was $47,900. Further investigation revealed that this transaction occurred through a Mexican retailer and was confirmed as a legitimate, non-fraudulent purchase.

Due to its extreme value and lack of similar transactions in the dataset, including this outlier could introduce significant bias, potentially skewing model results and impacting the detection of typical fraudulent patterns. After consulting with our business manager, we determined that excluding this transaction from our analysis would better support the model's accuracy and reliability.

With this technique, we excluded 1 record. From 97,497 to 97,496

Methods for imputation

In the Imputations step, we fill in missing values to ensure a complete dataset, enabling accurate pattern detection without gaps.

Clean and impute Merchnum

To begin with, we evaluate how many "Merchnum" records are null, so we can understand if our imputation process is improving the quality of the data. The total of null records is 3,279

The first imputation is performed by mapping and matching the "Merch description" field from those records that have "Merchnum" null to those that are not null. In this way, we can add the "Merchnum" of those records that match the "Merch description" and there is information about its "Merchnum". Now, the new total of null records is 2,115.

The second imputation is performed by setting the “Merchnum” to “unknown” when the “Merch description” is equal to “RETAIL CREDIT ADJUSTMENT” or “RETAIL DEBIT ADJUSTMENT”. That reduces the total number of null records to 1,421.

The last imputation is performed by intelligently identifying that out of the 1,421 null records for the “Merchnum” field, there are 515 unique “Merch description”. Then, for those 515 unique descriptions, we are going to impute a unique new “Merchnum”. The final number of null records for “Merchnum” is 0.

Clean and impute State

To begin with, we evaluate how many “State” records are null, so we can understand if our imputation process is improving the quality of the data. The total of null records is 1,028.

First, we create a dictionary named zip_state to map the “State” corresponding to that Zip number that is extracted from the web. The new total of null records is 954.

Second, we map the fields of “merchnum” and “Merch description” of the records that are not null in “State” with the null ones. This step reduces only two records. The new total of null is 952.

Third, we identify that there are transactions with “Merch description” equal 'RETAIL CREDIT ADJUSTMENT' or 'RETAIL DEBIT ADJUSTMENT' that don’t belong to any “State”. Therefore we case them as “Unknown”. The total of null records is 952.

Fourth, we case the states that are outside the US as “Foreign” to improve Fraud detection given that those transactions are more likely to be fraudulent ones.

Finally, for the remaining 297 cases without State, we case them as “unknown” given that we don’t have useful information to provide an accurate value.

Clean and impute ZIP

To begin with, the code evaluates how many “Merch Zip” records are null, so we can understand if our imputation process is improving the quality of the data. The total of null records is 4,347

The first imputation is performed by mapping with “Merchnum” and “Merch description” and filling in the zip code for those records that were null. After the first imputation, the number of null zip records is now 2,625.

Second, we are assigning “unknown” values to the zip records that have a “Merch description” equal to 'RETAIL CREDIT ADJUSTMENT' or 'RETAIL DEBIT ADJUSTMENT' as well as we did with the state imputation. The new number of null values is 1,940.

Third, for those records that “State” is known but the zip code is null we are going to impute the most populated zip code in the state. That was performed by using a dictionary to map “State” and impute the zip code. After this imputation, we have only 531 null “Merch zip”.

For the remaining 531 null records in “Merch zip” we assign the value “unknown” for those records.

Variable creation

Fraud detection summary

Credit Card Transaction Fraud Overview

Credit card fraud can occur through various methods by which account information is compromised. A card may be physically lost or stolen, allowing unauthorized individuals to make purchases. Alternatively, fraud can occur at a common point of compromise, where a merchant or third party gains access to multiple card details, sometimes using devices like skimmers at gas stations, ATMs, or other retail locations to create counterfeit cards. Online account hacking is also a prevalent method, where a fraudster gains access to an account's login credentials, enabling them to retrieve card information for fraudulent transactions.

Key Fraud Indicators

Certain transaction patterns serve as strong indicators of potential fraud. These include sudden bursts of activity across multiple merchants, unusually large purchases, usage at unfamiliar or high-risk merchants (such as online, jewelry, or electronics stores), and transactions in geographically distant or unexpected locations. Other signals include increased frequency in "card-not-present" scenarios, as well as infrequent recurring charges of the same amount or at the same merchant. These indicators are critical for developing detection variables and flagging suspicious activities.

New variables creation

With the fraud detection overview in mind, we created 2,634 new variables to better predict fraud behavior.

Main variables creation

Description	# of Variables created
Day of week target encoded: Average percentage fraud of that day	1
Merch state target encoded: Average percentage fraud for that state	1
Merch Zip target encoded: Average percentage fraud for that merch zip	1
Day since variables: number of days since a transaction with that same entity was seen	23
Amount variables: The {Average, max, median, total, Actual/total, Actual/maximum, Actual/median, Actual/total} transaction for each different entity over the past {0, 1, 3, 7, 14, 30, 60} days	1,288
Velocity: Number of transactions for each same entity over the past {0,1,3,7,14,30, 60} days.	161. Day since + Amount + Velocity = 1,472 as stated in the code.
Relative velocity: {Number, Amount} of transactions with the same entity over the past {0, 1} days divided by average {Number, Amount} (same as numerator) of transactions with the same entity over the past {7, 14, 30, 60} days.	368
Velocity change variable: Number of transactions for the same entity over the last {0, 1} days divided by the number of days since last transaction for that same entity.	184
Variability in the Amount paid: Calculates the Avg, Max, Med of the difference between the amount paid in the transaction and {0, 1, 3, 7, 14, 30} previous days	414
Counts by entity group 1: Number of unique counts for each {'Cardnum', 'Merchnum', 'card_merch', 'card_zip', 'card_state'} for each other entity {'Cardnum', 'Merchnum', 'card_merch', 'card_zip', 'card_state'} over the past {1, 3, 7, 14, 30, 60} days. For example, for	120

the ‘Cardnum’ = 1111 there were 10 different ‘Merchnum’ transactions over the last 30 days.	
Counts by entity group 2: Number of unique counts for each {'merch_zip', 'merch_state', 'state_des', 'Card_Merchdesc', 'Card_dow'} for each other entity {'merch_zip', 'merch_state', 'state_des', 'Card_Merchdesc', 'Card_dow'} over the past {1, 3, 7, 14, 30, 60} days. For example, for the ‘_= 1111 there were 10 different ‘Merchnum’ transactions over the last 30 days.	120
Counts by entity group 3: Number of unique counts for each {'Merchnum_desc', 'Merchnum_dow', 'Merchdesc_dow', 'Card_Merchnum_desc', 'Card_Merchnum_Zip'} for each other entity {'Merchnum_desc', 'Merchnum_dow', 'Merchdesc_dow', 'Card_Merchnum_desc', 'Card_Merchnum_Zip'} over the past {1, 3, 7, 14, 30, 60} days.	120
Counts by entity group 4: Number of unique counts for each {'Card_Merchdesc_Zip', 'Merchnum_desc_State', 'Merchnum_desc_Zip', 'merchnum_zip', 'Merchdesc_State', 'Merchdesc_Zip', 'Card_Merchnum_State', 'Card_Merchdesc_State'} for each other entity {'Card_Merchdesc_Zip', 'Merchnum_desc_State', 'Merchnum_desc_Zip', 'merchnum_zip', 'Merchdesc_State', 'Merchdesc_Zip', 'Card_Merchnum_State', 'Card_Merchdesc_State'} over the past {1, 3, 7, 14, 30, 60} days.	336
Velocity change variable: Number of transactions for the same entity squared	184

over the last {0, 1} days divided by the number of days since last transaction for that same entity.	
Amount bin creation: Creates 5 bins divided by transaction amount into five essentially equal-spaced numbers of transactions.	1
Foreign transaction: Identify if the transaction was performed outside the US	1

Extra variables creation

Description	# of Variables created
Relative foreign transactions: Number of foreign transactions per cardnum over the last {0 1} days divided by the number of foreign transactions for the same card num over the past {1, 3, 7, 14, 30, 60} days.	12
Relative foreign transactions amount: Average amount of foreign transactions per cardnum over the last {0 1} days divided by the average amount of foreign transactions for the same card num over the past {1, 3, 7, 14, 30, 60} days.	12
Unique location transactions count: Number of transactions that occurred in a unique zip location for each cardnum over the last {0, 1} days divided by the number of transactions in unique zip location over the last {1, 3, 7, 14, 30, 60} days.	12

After the creation of each group of variables, some columns are going to be redundant and after deleting those, we are going to end up with the 2,634 new variables with predicting power to detect fraud.

Feature selection

Feature selection summary

Feature selection is a crucial step in building effective predictive models, especially in complex domains like fraud detection. It involves identifying and selecting the most relevant variables (features) from a larger dataset to improve model performance and interpretability. By focusing on the most informative features, feature selection reduces unnecessary noise and complexity, leading to more efficient and accurate modeling.

We performed feature selection to address the "curse of dimensionality," which suggests that as the number of features (dimensions) increases, it becomes harder to fit nonlinear models effectively. By reducing dimensionality, we can explore many potential variables without overwhelming the model, allowing us to focus on the most impactful ones. Additionally, feature selection produces a ranked list of variables, which guides our decisions on which features to add or remove in model experimentation. This approach accelerates the modeling process, enabling faster runs and facilitating optimization of model architecture and hyperparameters.

Feature selection steps

First step - Filter

The filter method in feature selection quickly evaluates each variable independently to determine its relevance to the target outcome. As a univariate approach, it assesses each candidate variable on its own, ignoring correlations with other variables. This method ranks variables by their individual predictive importance, providing a prioritized list to guide further modeling steps. Its main advantage is speed and scalability, as each variable is evaluated separately, making it efficient for handling large datasets and identifying promising features early in the process.

Second step - Wrapper

The wrapper method in feature selection uses a model wrapped around the process to evaluate combinations of variables rather than each variable individually. This multivariate approach considers groups of candidate variables together, allowing it to account for correlations and interactions, unlike the filter method. Common techniques include forward selection (adding variables), backward selection (removing variables), or a combination of both, known as stepwise selection. The wrapper method produces a ranked list of variables based on their combined importance, providing a refined selection tailored to the model's performance. However, it is generally slower than filter methods due to the need to assess multiple variable combinations, making it more suited for smaller datasets or after an initial filtering step.

Final step – Regularization

Regularization aids feature selection by identifying relevant features and reducing overfitting. L1 (lasso) regularization can shrink some coefficients to zero, effectively removing less important features. L2 (ridge) regularization reduces the influence of less important features by

shrinking their coefficients. These methods help create a more interpretable and stable model by balancing complexity and accuracy.

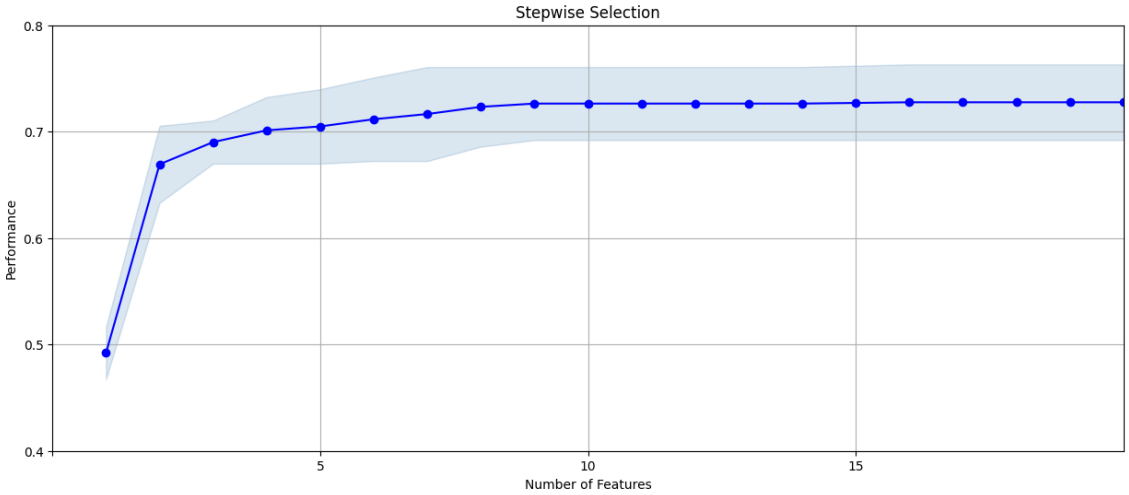
Feature selection - our process

For feature selection, we performed different combinations of filters and wrappers to asses for the best overall option.

Between them:

1- **num_filter = 200, num_wrapper=20, Random forest 5 trees, forward selection**
2- **num_filter = 200, num_wrapper=20, Random forest 5 trees, backward selection**
3- **num_filter = 200, num_wrapper=20, LGMB Classifier, forward selection**
4- **num_filter = 800, num_wrapper=20, LGMB Classifier, forward selection**

Based on the analysis, the optimal set of variables has been identified with the following parameters: num_filter = 800, num_wrapper = 20, LGBM Classifier, and forward selection. This selection was made due to its balanced mix of long and short windows, as well as the diverse entities that it has. The model's performance stabilizes around the 7th variable, thus the first 10 variables have been chosen as the optimal set for future predictions. Also, the overall performance was around 0.73 being the best one in comparison with the other tryals. The 20 selected variables are listed below



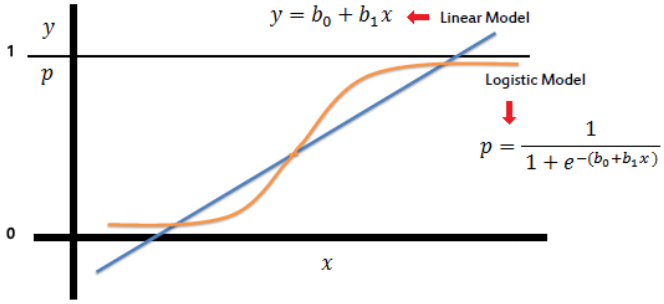
wrapper order	variable	filter score
1	Cardnum_unique_count_for_card_state_1	0,47607
2	Card_Merchdesc_State_total_7	0,32467
3	Card_Merchnum_desc_vdratio_1by14	0,26129
4	Cardnum_count_1_by_30	0,42823
5	Cardnum_max_14	0,31883
6	Card_dow_unique_count_for_merch_state_1	0,44736
7	Card_dow_vdratio_0by14	0,47909
8	Card_Merchnum_Zip_total_30	0,31441
9	merch_zip_max_30	0,25040
10	Cardnum_actual/toal_0	0,47955
11	Card_dow_vdratio_0by7	0,46796
12	Cardnum_vdratio_1by7	0,46677
13	Cardnum_unique_count_for_card_state_3	0,46641
14	Cardnum_actual/toal_1	0,45972
15	card_state_count_1_by_60_sq	0,30914
16	Cardnum_count_14	0,44544
17	Card_dow_count_7	0,48238
18	Card_dow_unique_count_for_merch_zip_1	0,44716
19	Cardnum_unique_count_for_card_state_7	0,44597
20	Cardnum_actual/max_0	0,44573

Preliminary model explores

Models used - description

Logistic regression

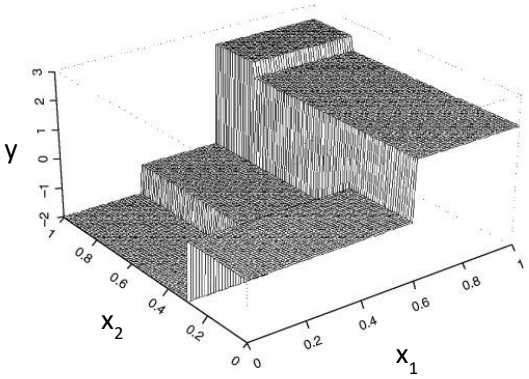
Logistic regression is a statistical method for binary classification that predicts the probability of a categorical outcome based on one or more independent variables. It is useful for modeling the likelihood of events like customer churn or purchase decisions. The model uses a logistic function to ensure probabilities range between 0 and 1. Generally speaking, this model is the base case from which we can compare non linear models.

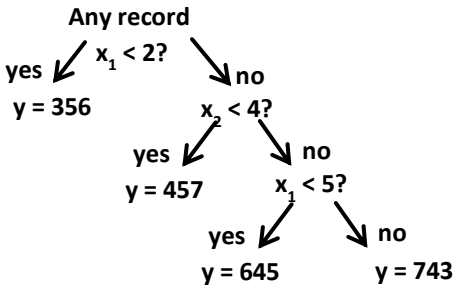


Decision tree

A decision tree is a versatile machine learning model used for both classification and regression tasks. It works by recursively splitting the data into subsets based on the most significant predictor variables, creating a tree-like structure of decisions. Each node represents a decision point based on a specific feature, leading to branches that represent the possible outcomes. This method is highly interpretable, as it visually maps out the decision-making process, making it easy to understand the influence of each variable.

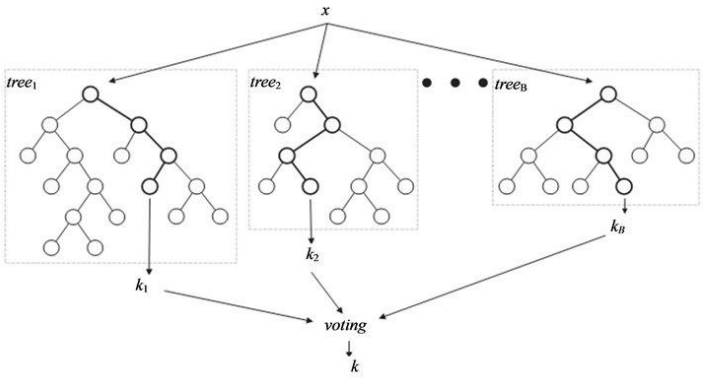
Decision tree with 2 independent variables





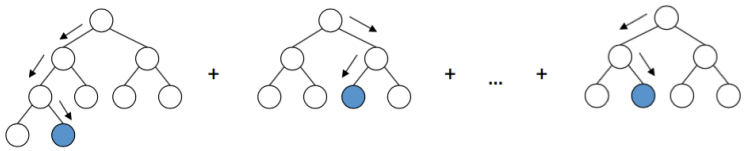
Random forest

Random forest is an advanced ensemble learning technique used for both classification and regression tasks. It operates by constructing multiple decision trees during training and merging their outputs to improve predictive accuracy and control overfitting. Each tree in the forest is built from a random subset of the data and features, ensuring diversity among the trees. Typically in Random Forest, we build a collection of many independent strong trees and average the result across them or voting (for classification). Each tree is built independently and is a strong, deep tree.



Boosted trees

Boosted trees are an ensemble learning technique that enhances the performance of decision trees by sequentially building a series of trees, where each tree corrects the errors of its predecessor. This method combines the strengths of multiple weak learners to create a strong predictive model. By focusing on the residuals of previous trees, boosted trees iteratively improve accuracy and reduce bias. We used LGBM and Cat Boost



LGBM (Light Gradient Boosting Machine) is a type of boosted tree algorithm. It is an implementation of gradient boosting that focuses on efficiency and scalability, making it particularly suitable for large datasets and high-dimensional data. LightGBM builds multiple decision trees sequentially, where each tree aims to correct the errors of the previous ones, thereby improving the overall model performance.

CatBoost (Categorical Boosting) is a powerful gradient boosting algorithm specifically designed to handle categorical features efficiently. Unlike traditional gradient boosting methods, CatBoost incorporates techniques to process categorical data directly, reducing the need for extensive preprocessing such as one-hot encoding. It uses ordered boosting to prevent overfitting and employs a combination of symmetric trees and oblivious trees to enhance prediction accuracy and speed.

Neural network

Neural networks are sophisticated machine learning models inspired by the human brain's structure and function, designed to recognize patterns and make predictions. They consist of interconnected layers of nodes, or neurons, where each connection has an associated weight. Data is passed through these layers, with each neuron applying a transformation to the input and passing the result to the next layer. The network learns by adjusting the weights through a process called backpropagation, which minimizes the error between predicted and actual outcomes

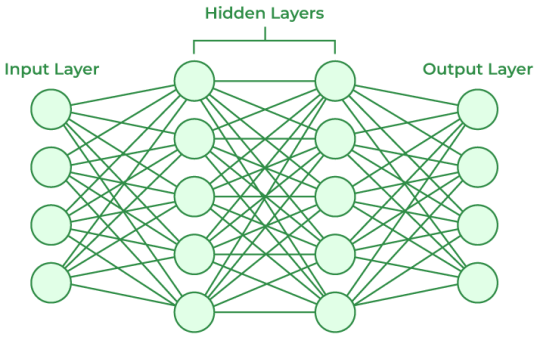
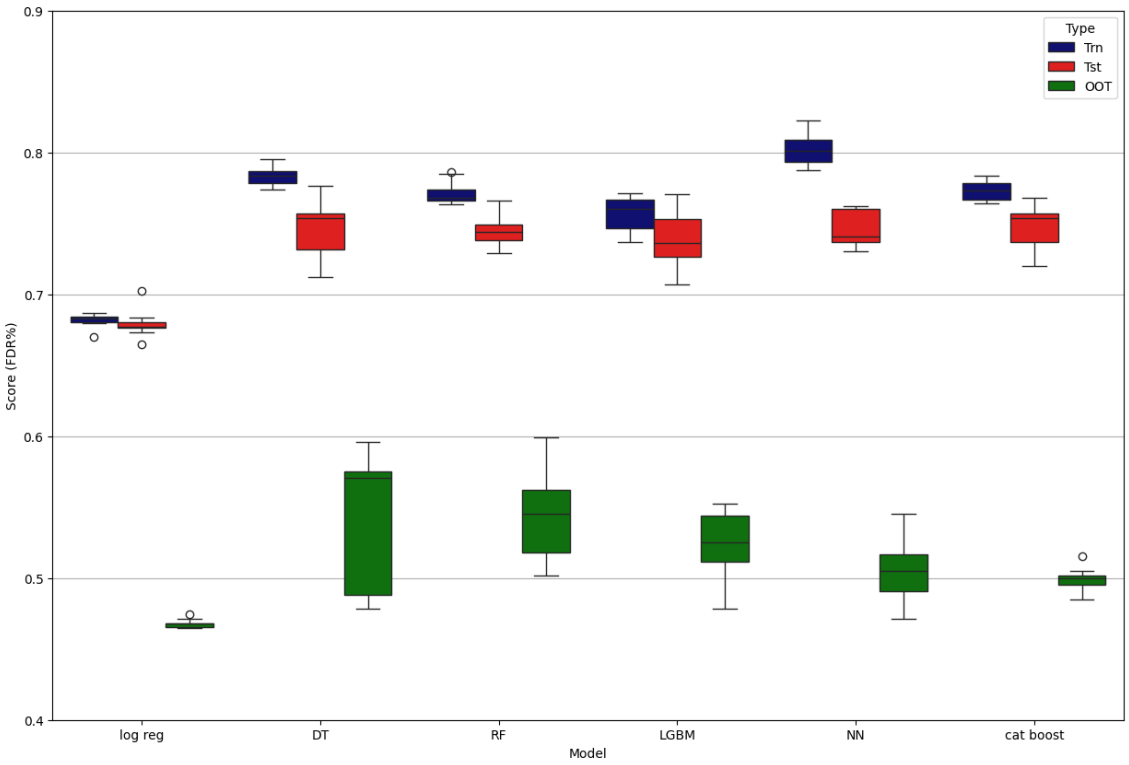


Table of tests

Model		Parameters					Average FDR at 3%		
Logistic Regression	Iteration	Penalty	C	Solver	max_iter		Trn	Tst	OOT
	1	l2	1	lbfgs	100		0,68	0,68	0,46
	2	l2	0,5	lbfgs	150		0,68	0,69	0,47
	3	l1	1	liblinear	100		0,68	0,69	0,47
	4	**l1**	**0,5**	**liblinear**	**150**		**0,68**	**0,68**	**0,47**
Decision Tree	Iteration	Splitter	Max_depth	min_sample_split	min_sample_leaf		Trn	Tst	OOT
	1	best	5	60	30		0,70	0,69	0,48
	2	**best**	**10**	**50**	**25**		**0,79**	**0,74**	**0,55**
	3	best	15	40	20		0,84	0,77	0,50
	4	random	5	60	30		0,63	0,63	0,45
	5	random	10	50	25		0,73	0,69	0,49
	6	random	15	40	20		0,73	0,71	0,51
Random Forest	Iteration	n_estimators	max_depth	min_sample_split	min_sample_leaf	max_features	Trn	Tst	OOT
	1	50	5	120	60	7	0,72	0,71	0,48
	2	50	10	80	40	13	0,77	0,73	0,55
	3	100	5	120	60	7	0,72	0,72	0,49
	4	**100**	**10**	**80**	**40**	**13**	**0,77**	**0,74**	**0,56**
	5	150	10	60	30	13	0,79	0,75	0,57
	6	150	15	40	20	13	0,85	0,78	0,55
Boosted Tree(LGBM)	Iteration	n_estimators	num_leaves	max_depth	min_child_samples		Trn	Tst	OOT
	1	30	5	3	60		0,75	0,75	0,54
	2	**30**	**7**	**4**	**40**		**0,77**	**0,76**	**0,54**
	3	50	10	5	30		0,81	0,77	0,52
	4	50	15	6	20		0,85	0,80	0,52
	5	70	10	5	30		0,82	0,78	0,54
	6	70	7	4	30		0,80	0,76	0,53
Neural Network	Iteration	n_layers	n_neurons per layer	learning_rate	momentum		Trn	Tst	OOT
	1	2	2	0,001	0,9		0,69	0,69	0,47
	2	2	10	0,001	0,9		0,75	0,72	0,50
	3	3	10	0,001	0,5		0,75	0,73	0,51
	4	4	10	0,01	0,9		0,75	0,73	0,52
	5	**4**	**10**	**0,005**	**0,9**		**0,76**	**0,75**	**0,53**
	6	5	15	0,005	0,9		0,80	0,75	0,51
Boosted Tree (Cat boost)	Iteration	verbose	learning_rate	l2_leaf_reg	depth	min_data_in_leaf	Trn	Tst	OOT
	1	**0**	**0,02**	**3**	**3**	**60**	**0,77**	**0,76**	**0,50**
	2	0	0,02	4	4	60	0,84	0,78	0,52
	3	0	0,03	5	4	80	0,83	0,79	0,52
	4	0	0,03	6	4	40	0,83	0,79	0,51
	5	0	0,03	7	4	30	0,83	0,79	0,51
	6	0	0,03	7	5	10	0,86	0,80	0,52

Performance plot



Final model performance

Light Gradient Boosting Machine (LGBM)

LGBMClassifier is a class provided by the LightGBM library, which implements the Light Gradient Boosting Machine algorithm for classification tasks. LightGBM is designed for high performance and efficiency, particularly with large datasets and high-dimensional data. It builds an ensemble of decision trees sequentially, where each tree corrects the errors of the previous ones, enhancing the overall model's predictive accuracy.

Key features of LGBMClassifier include:

1. **Speed and Efficiency**: LightGBM uses histogram-based decision tree learning, which reduces memory usage and speeds up training.
2. **Leaf-wise Tree Growth**: Unlike level-wise growth used in other boosting algorithms, LightGBM grows trees leaf-wise, which can result in deeper trees and better accuracy.
3. **Scalability**: LightGBM supports parallel and GPU learning, making it scalable for large datasets.

Hyperparameters of choice:

1. **n_estimators**: This parameter specifies the number of boosting iterations, or trees, to be built. In this case, n_estimators=30 means the model will build 30 trees sequentially.
2. **num_leaves**: This parameter controls the maximum number of leaves in one tree. Setting num_leaves=7 limits each tree to have at most 7 leaves, which helps control the model's complexity and prevent overfitting.
3. **max_depth**: This parameter sets the maximum depth of each tree. With max_depth=4, each tree can have up to 4 levels, which also helps in controlling overfitting by limiting the tree's growth.
4. **min_child_samples**: This parameter specifies the minimum number of data samples that a leaf must have. Setting min_child_samples=60 ensures that each leaf has at least 60 samples, which helps in preventing overfitting by avoiding overly specific splits.

Summary Tables

Training	# Records	# Goods	# Bads	# Fraud Rate
	59684	58455	1229	0,020591783

	Bin Statistics					Cumulative Statistics						
Population bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Comulative Goods	Comulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	597	62	535	10,39%	89,61%	597	62	535	0,11%	43,53%	43,43	0,12
2	597	257	340	43,05%	56,95%	1194	319	875	0,55%	71,20%	70,65	0,36
3	597	514	83	86,10%	13,90%	1791	833	958	1,43%	77,95%	76,52	0,87
4	596	552	44	92,62%	7,38%	2387	1385	1002	2,37%	81,53%	79,16	1,38
5	597	561	36	93,97%	6,03%	2984	1946	1038	3,33%	84,46%	81,13	1,87
6	597	568	29	95,14%	4,86%	3581	2514	1067	4,30%	86,82%	82,52	2,36
7	597	581	16	97,32%	2,68%	4178	3095	1083	5,29%	88,12%	82,83	2,86
8	597	588	9	98,49%	1,51%	4775	3683	1092	6,30%	88,85%	82,55	3,37
9	597	586	11	98,16%	1,84%	5372	4269	1103	7,30%	89,75%	82,44	3,87
10	596	585	11	98,15%	1,85%	5968	4854	1114	8,30%	90,64%	82,34	4,36
11	597	589	8	98,66%	1,34%	6565	5443	1122	9,31%	91,29%	81,98	4,85
12	597	593	4	99,33%	0,67%	7162	6036	1126	10,33%	91,62%	81,29	5,36
13	597	591	6	98,99%	1,01%	7759	6627	1132	11,34%	92,11%	80,77	5,85
14	597	592	5	99,16%	0,84%	8356	7219	1137	12,35%	92,51%	80,16	6,35
15	597	593	4	99,33%	0,67%	8953	7812	1141	13,36%	92,84%	79,48	6,85
16	596	589	7	98,83%	1,17%	9549	8401	1148	14,37%	93,41%	79,04	7,32
17	597	590	7	98,83%	1,17%	10146	8991	1155	15,38%	93,98%	78,60	7,78
18	597	590	7	98,83%	1,17%	10743	9581	1162	16,39%	94,55%	78,16	8,25
19	597	593	4	99,33%	0,67%	11340	10174	1166	17,40%	94,87%	77,47	8,73
20	597	596	1	99,83%	0,17%	11937	10770	1167	18,42%	94,96%	76,53	9,23

Testing	# Records	# Goods	# Bads	# Fraud Rate
	25580	25059	521	0,020367475

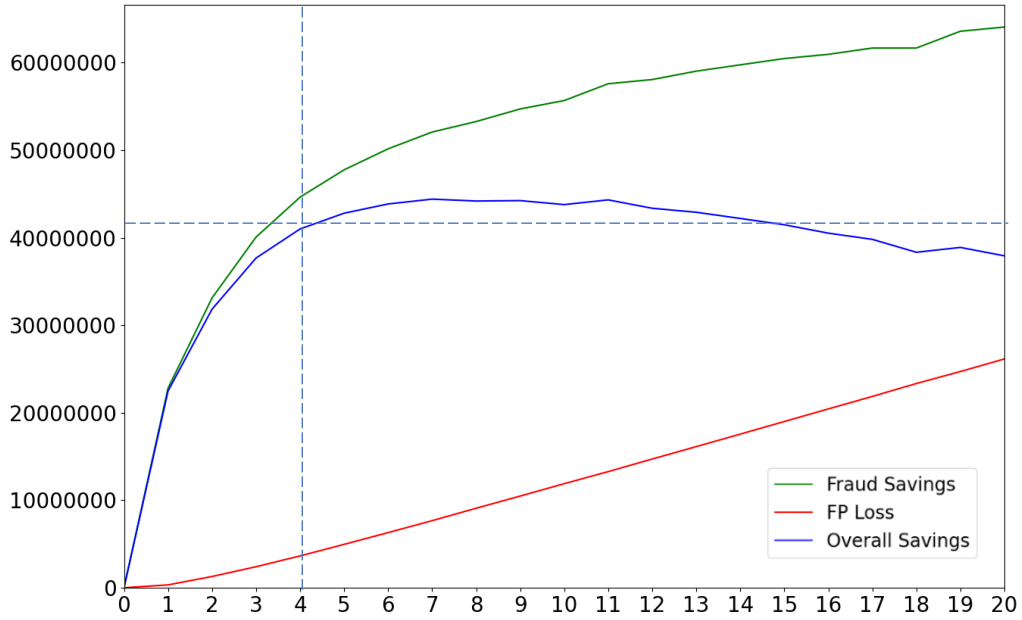
	Bin Statistics					Cumulative Statistics					
Population bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Comulative Goods	Comulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	256	34	222	13,28%	86,72%	256	34	222	0,14%	42,61%	42,47	0,15
2	256	133	123	51,95%	48,05%	512	167	345	0,67%	66,22%	65,55	0,48
3	255	210	45	82,35%	17,65%	767	377	390	1,50%	74,86%	73,35	0,97
4	256	232	24	90,63%	9,38%	1023	609	414	2,43%	79,46%	77,03	1,47
5	256	245	11	95,70%	4,30%	1279	854	425	3,41%	81,57%	78,17	2,01
6	256	243	13	94,92%	5,08%	1535	1097	438	4,38%	84,07%	79,69	2,50
7	256	247	9	96,48%	3,52%	1791	1344	447	5,36%	85,80%	80,43	3,01
8	255	248	7	97,25%	2,75%	2046	1592	454	6,35%	87,14%	80,79	3,51
9	256	249	7	97,27%	2,73%	2302	1841	461	7,35%	88,48%	81,14	3,99
10	256	248	8	96,88%	3,13%	2558	2089	469	8,34%	90,02%	81,68	4,45
11	256	251	5	98,05%	1,95%	2814	2340	474	9,34%	90,98%	81,64	4,94
12	256	253	3	98,83%	1,17%	3070	2593	477	10,35%	91,55%	81,21	5,44
13	255	253	2	99,22%	0,78%	3325	2846	479	11,36%	91,94%	80,58	5,94
14	256	255	1	99,61%	0,39%	3581	3101	480	12,37%	92,13%	79,76	6,46
15	256	254	2	99,22%	0,78%	3837	3355	482	13,39%	92,51%	79,13	6,96
16	256	254	2	99,22%	0,78%	4093	3609	484	14,40%	92,90%	78,50	7,46
17	256	256	0	100,00%	0,00%	4349	3865	484	15,42%	92,90%	77,47	7,99
18	255	253	2	99,22%	0,78%	4604	4118	486	16,43%	93,28%	76,85	8,47
19	256	256	0	100,00%	0,00%	4860	4374	486	17,45%	93,28%	75,83	9,00
20	256	254	2	99,22%	0,78%	5116	4628	488	18,47%	93,67%	75,20	9,48

OOT	# Records	# Goods	# Bads	# Fraud Rate
	12232	11935	297	0,024280576

	Bin Statistics					Cumulative Statistics					
Population bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Comulative Goods	Comulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	122	27	95	22,13%	77,87%	122	27	95	0,23%	31,99%	31,76	0,28
2	123	80	43	65,04%	34,96%	245	107	138	0,90%	46,46%	45,57	0,78
3	122	93	29	76,23%	23,77%	367	200	167	1,68%	56,23%	54,55	1,20
4	122	103	19	84,43%	15,57%	489	303	186	2,54%	62,63%	60,09	1,63
5	123	110	13	89,43%	10,57%	612	413	199	3,46%	67,00%	63,54	2,08
6	122	112	10	91,80%	8,20%	734	525	209	4,40%	70,37%	65,97	2,51
7	122	114	8	93,44%	6,56%	856	639	217	5,35%	73,06%	67,71	2,94
8	123	118	5	95,93%	4,07%	979	757	222	6,34%	74,75%	68,40	3,41
9	122	116	6	95,08%	4,92%	1101	873	228	7,31%	76,77%	69,45	3,83
10	122	118	4	96,72%	3,28%	1223	991	232	8,30%	78,11%	69,81	4,27
11	123	115	8	93,50%	6,50%	1346	1106	240	9,27%	80,81%	71,54	4,61
12	122	120	2	98,36%	1,64%	1468	1226	242	10,27%	81,48%	71,21	5,07
13	122	118	4	96,72%	3,28%	1590	1344	246	11,26%	82,83%	71,57	5,46
14	122	119	3	97,54%	2,46%	1712	1463	249	12,26%	83,84%	71,58	5,88
15	123	120	3	97,56%	2,44%	1835	1583	252	13,26%	84,85%	71,58	6,28
16	122	120	2	98,36%	1,64%	1957	1703	254	14,27%	85,52%	71,25	6,70
17	122	119	3	97,54%	2,46%	2079	1822	257	15,27%	86,53%	71,27	7,09
18	123	123	0	100,00%	0,00%	2202	1945	257	16,30%	86,53%	70,24	7,57
19	122	114	8	93,44%	6,56%	2324	2059	265	17,25%	89,23%	71,97	7,77
20	122	120	2	98,36%	1,64%	2446	2179	267	18,26%	89,90%	71,64	8,16

Financial curves and recommended cutoff

Financial curves



Recommended cutoff

The objective is to establish an optimal score cutoff for a fraud detection system, with the primary objective of maximizing overall savings. The proposed methodology emphasizes finding a balance between two key outcomes: detecting fraudulent transactions and minimizing false positives that lead to revenue loss. By setting a threshold score, the system can distinguish between legitimate and potentially fraudulent transactions, thereby safeguarding revenue while enhancing fraud prevention efforts.

The chart visualizes three curves to guide decision-making on cutoff selection. The green curve, labeled “Fraud $ Savings,” represents cumulative savings from successfully flagged fraud cases as the score threshold increases. However, this curve plateaus beyond a certain point, indicating diminishing returns in fraud savings. Conversely, the red curve, “FP Loss” (lost in revenue) represents the financial impact of legitimate transactions incorrectly flagged as fraud (false positives), with losses escalating sharply at higher cutoffs. The blue curve, “Overall Savings,” captures the net result, showing the optimal threshold where fraud savings are maximized without significantly increasing revenue loss from false positives.

The recommended cutoff is strategically positioned around 4% of transactions, where overall savings are close to the maximum. This point reflects a balance where fraud detection remains high, but the growth in lost revenue due to false positives is kept in check. With this threshold, the model anticipates an annual savings of approximately $42 million. By integrating assumptions such as a $400 gain per detected fraud and a $20 loss per false positive, and scaling results based on a sample, this methodology provides a data-driven and financially sound foundation for the client to optimize their fraud prevention efforts.

Summary

This project aimed to design, train, and implement a fraud detection model to enhance the organization's ability to detect fraudulent transactions effectively while minimizing false positives. The process began with a clear design and solution approach, then extensive data collection and preparation (EDA), including cleaning, preprocessing, and feature engineering to ensure high-quality input data. A range of features was generated based on transaction history, user behavior, and transaction attributes, all designed to capture patterns indicative of fraud. Then we performed feature selection to reduce the amount of variables. The first step was a filter to stay with 800 variables and then we applied a wrapper with forward selection to find the most important one.

Once we had all the data prepared, The model training phase involved testing multiple machine learning algorithms and assessing each one's effectiveness in identifying fraud within the dataset. As a baseline model, we performed a logistic regression. After tuning and optimization of the nonlinear models, the final model was selected based on its accuracy, precision, recall, and overall performance in identifying fraudulent transactions while limiting false positives. We selected an LGBM model with 30 sequential trees, no more than 7 leaves, 4 levels deep as maximum, and 40 points per leave as minimum. To evaluate the model's robustness, it was tested on an out-of-time (OOT) dataset, achieving a Fraud Detection Rate (FDR) of 4%. This FDR threshold was strategically chosen to strike a balance between fraud detection and business cost, with an estimated savings of approximately $42 million annually based on the model's predictive capabilities.

In addition to the main analysis, we assessed various threshold settings to understand the impact of different FDR levels on financial outcomes, such as saved revenue and lost opportunities due to false positives. The project concludes that, while the model is effective, there are potential areas for improvement. Future enhancements could include incorporating additional data sources, and periodically retraining the model on new data to adapt to evolving fraud tactics. Moreover, exploring alternative threshold settings and refining cost calculations could further optimize the model's business impact, ensuring sustained efficiency in fraud detection.

Appendix

Data Quality Report

1. Data Description

The dataset is **Card Transactions**, which contains **Transactions** of credit cards that occurred in 2010. The data came from a **US government Organization** in Tennessee. There are **10 fields 97,852 records**.

2. Summary Tables

Numeric Fields Table

Field Name	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Stdev	Most Common
Amount	97,852	100.00%	0	0.01	310,2045.53	425.47	9,949.8	3.62

Categorical Fields Table

Field Name	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Date	97,852	100.0%	0	365	2/28/10
Merchnum	94,455	96.5%	0	13,091	930090121224
Merch description	97,852	100.0%	0	13,126	GSA-FSS-ADV
Merch state	96,649	98.8%	0	227	TN
Transtype	97,852	100.0%	0	4	P
Recnum	97,852	100.0%	0	97,852	1
Fraud	97,852	100.0%	95,805	2	0
Cardnum	97,852	100.0%	0	1,645	5142148452
Merch zip	93,149	95.2%	0	4,567	38118

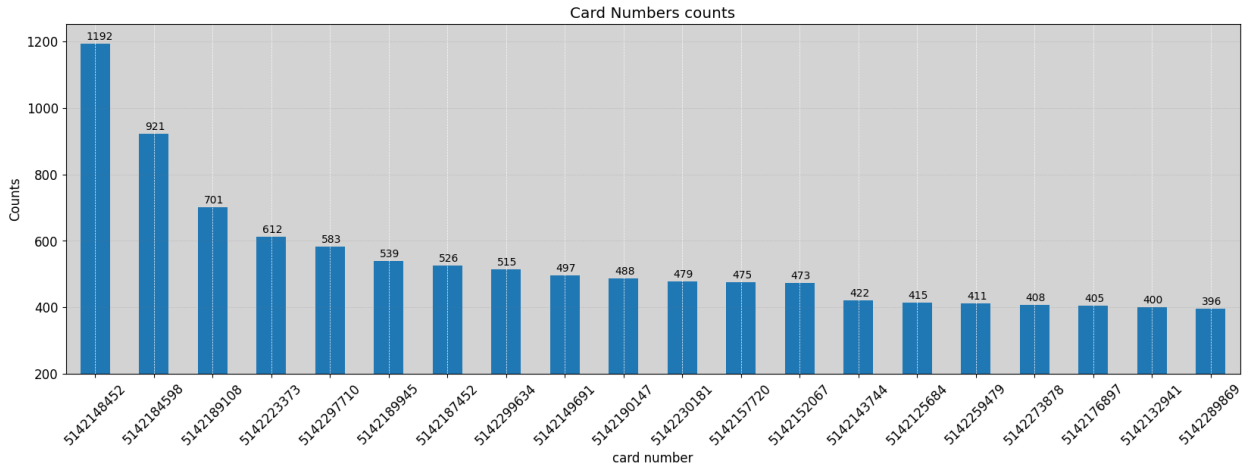
3. Visualization of Each Field

1) Field Name: Recnum

Description: Ordinal unique positive integer for each application record, from 1 to 1000,000.

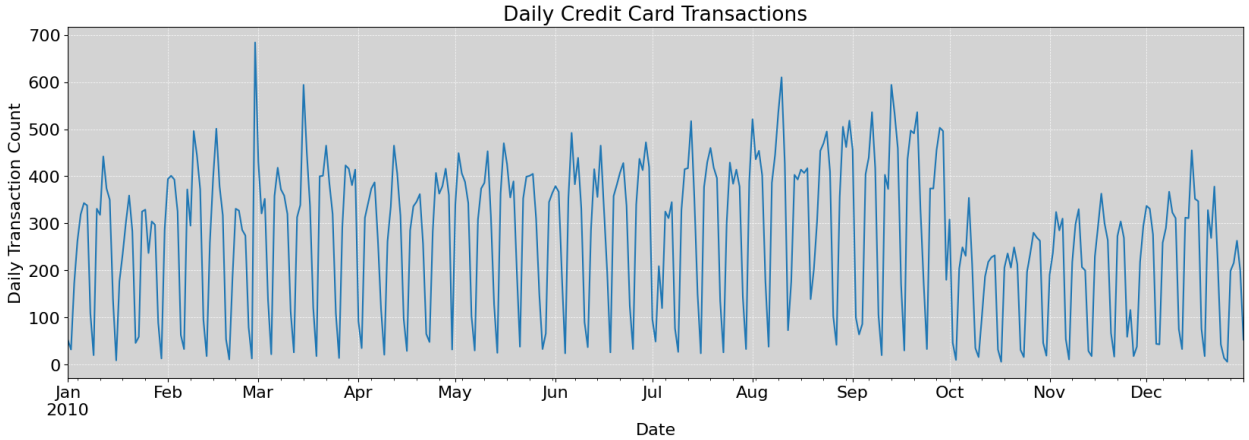
2) Field Name: Cardnum

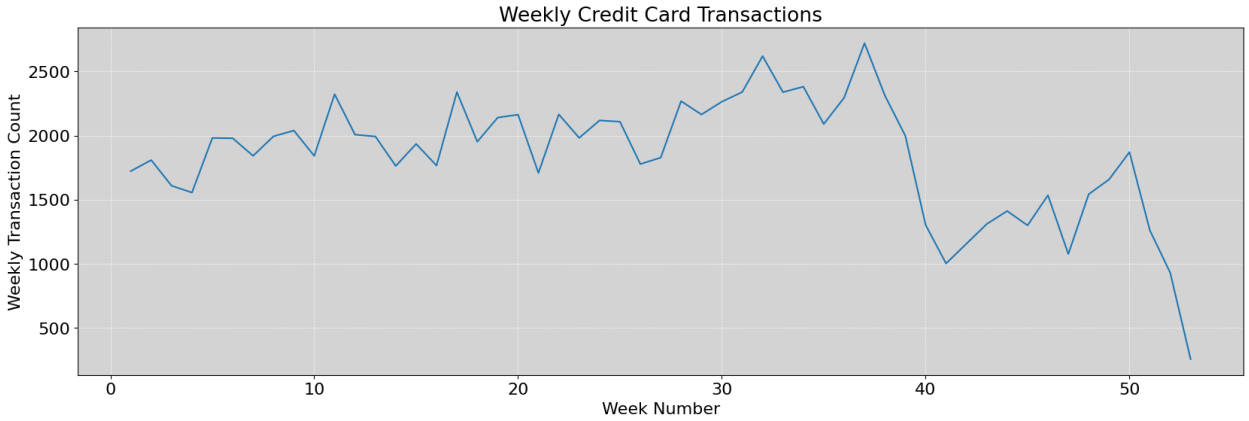
Description: Transaction's Card number. The distribution shows the top 20 field values of Cardnumb. The most common Cardnum is "5142148452" with a total count of 1,192.



3) Field Name: date

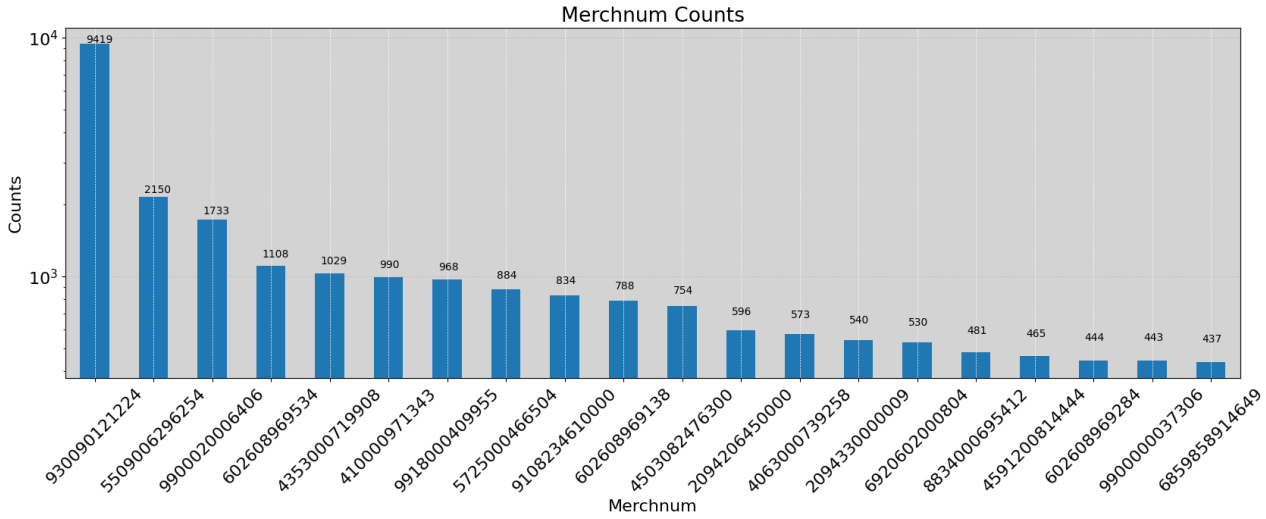
Description: Transaction date. The first distribution shows the number of daily transactions across time. The second distribution shows the number of weekly transactions across time.





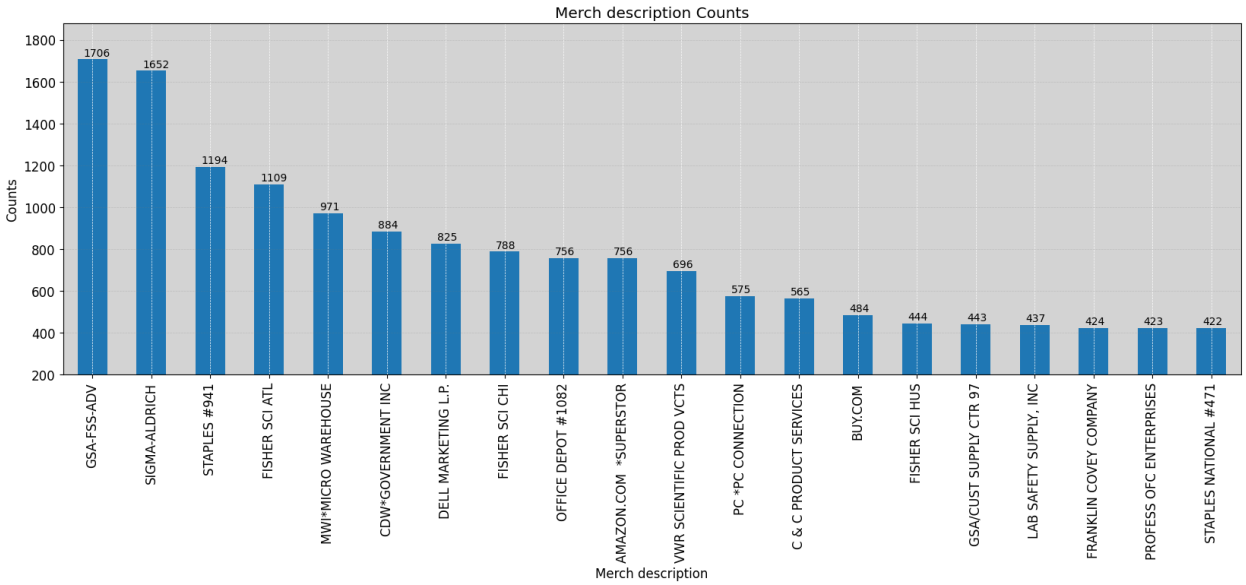
4) Field Name: Merchnum

Description: Transactions' Merchnum. The distribution shows the top 20 field values of Merchnum counts. The most common Merchnum is "930090121224 ", with a total count of 9,419.



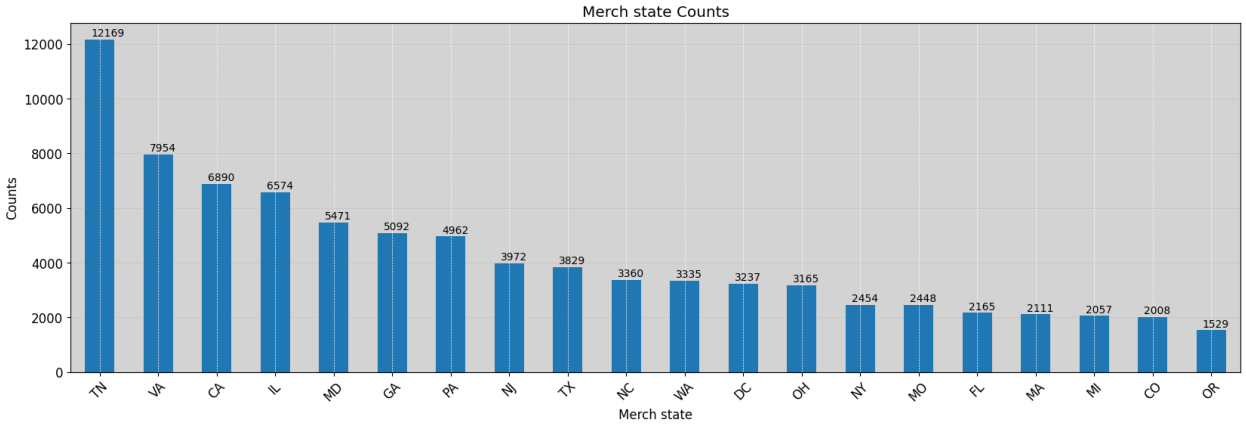
5) Field Name: Merch description

Description: Applicant's first name. The distribution shows the top 20 field values of transactions' Merch description. The most common Merch description is 'GSA-FSS-ADV, with a total count of 1,706.



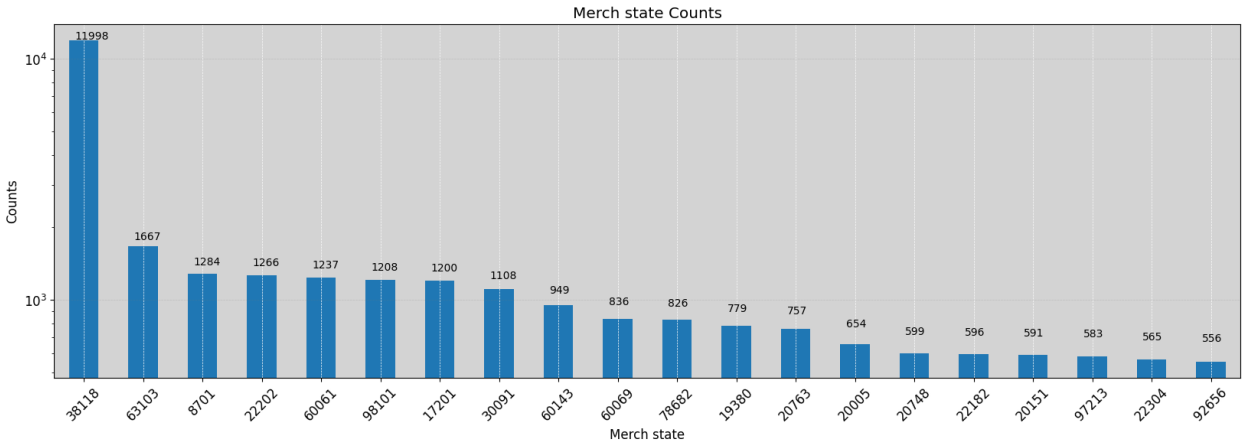
6) Field Name: Merch state

Description: Transactions' Merch state. The distribution shows the top 20 field values of Transactions' Merch state. The most common Merch state is 'TN', with a total count of 12,169.



7) Field Name: Merch zip

Description: Transactions' Merch zip. The distribution shows the top 20 field values of Transactions' Merch zip. The most common Merch zip is '38118, with a total count of 11,998.

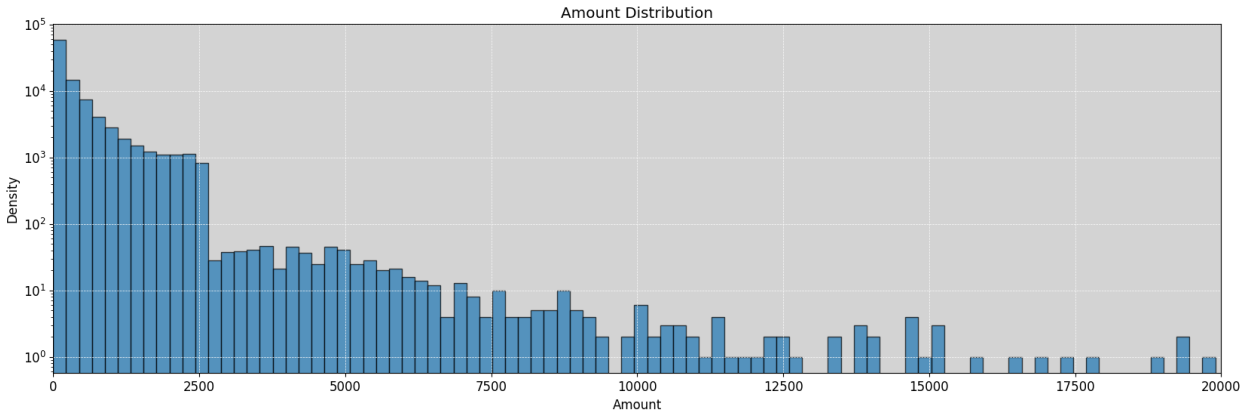


8) Field Name: Transtype

Description: Transactions' Transtype. The distribution shows the top 4 field values of Transactions' Transtype. The most Transtype code is "P", with a total count of 97,497.

9) Field Name: Amount

Description: Transactions' Amount. The distribution shows the histogram of Transactions' Amount. There is a discontinuity in the count of transactions that are greater in the range of 2,500 to 3,000.



10) Field Name: Fraud

Description: Fraud identification label. Fraud = 0 (Not fraudulent), Fraud =1 (Fraud identified). The total count of fraud_label = 0 is 95,805. The total count of fraud_label = 1 is 2,047.