Homework 5 - Project 1 Informal Report

Executive summary

In this fraud detection project, we analyzed a comprehensive dataset of credit card transactions from a US government organization, spanning a one-year period from January 1, 2010, to December 31, 2010. Our goal was to identify fraudulent transactions and reduce financial losses for the organization. By leveraging advanced data analytics techniques, we were able to achieve a Fraud Detection Rate (FDR@3%) of 56.98% for out-of-time (OOT) validation, which is a strong indicator of the model's effectiveness in identifying potential fraud. By implementing this fraud detection system, we estimate that the organization could potentially save up to $21,228,000, thus substantially mitigating the risk of financial losses due to fraudulent activities.

Data Description

1. Overview of data

The dataset is **Credit Card Transaction Data**, which contains real credit card transaction records from a US government organization for business purposes. The data came from the website of author Mark Nigrini over a **one-year period** from January 1, 2010, to December 31, 2010. There are **10 fields** and a total of **96,753 records**.

2. Statistics Tables

Numeric Fields Table

Field Name	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Most Common	Stdev
Date	96,753	100.00%	0	1/1/10	12/31/10	6/25/10	2/28/10	98 days 21:38
Amount	96,753	100.00%	0	0.01	3102045.53	427.88	3.62	10006.14

Categorical Fields Table

Field Name	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Recnum	96,753	100.00%	0	96,753	1
Cardnum	96,753	100.00%	0	1,645	5142148452
Merchnum	93,378	96.51%	0	13,091	930090121224
Merch description	96,753	100.00%	0	13,126	GSA-FSS-ADV
Merch state	95,558	98.76%	0	227	TN
Merch zip	92,097	95.19%	0	4,567	38118
Transtype	96,753	100.00%	0	4	P
Fraud	96,753	100.00%	95,694	2	0

Akansha Lalwani (A59019733)

Data Cleaning

Description of Exclusions

- We have a CSV format dataset that includes columns with missing values represented as 'NaN'. We will disregard these columns for our analysis.

- The date column's current data type is 'object', and we will transform it to the 'datetime' format.

- To adhere to the given requirements, we will filter the data to only include transactions with a transaction type of 'P' and remove any other transactions.

- Furthermore, there is an unusual transaction where the amount exceeds $3000000, which we will remove from the dataset as an outlier.

- Once we have cleaned the data, we will determine the number of rows that contain missing or 'NaN' values.

Imputation Logic

1. **Clean and impute merchnum:**

- First, it replaces any values of '0' in the 'Merchnum' column with NaN (not a number) using the replace() method increasing the null values from 3198 to 3251.

- Next, it creates a dictionary called 'merchdes_merchnum' to map 'Merch description' values to their corresponding 'Merchnum' values. This is done by iterating over the DataFrame rows where both 'Merch description' and 'Merchnum' are not null and adding each unique 'Merch description' to the dictionary along with its corresponding 'Merchnum' value.

- The code then fills in any remaining NaN values in the 'Merchnum' column by mapping each non-null 'Merch description' value to its corresponding 'Merchnum' value in the 'merchdes_merchnum' dictionary using the fillna() method.

- Any remaining NaN values in the 'Merchnum' column corresponding to adjustment transactions are assigned the value 'unknown' using the mask() method.

- Finally, any remaining NaN values in the 'Merchnum' column that do not have a corresponding 'Merch description' value are assigned a new unique 'Merchnum' value. This is done by creating a dictionary called 'merchnum_create' to map each unique 'Merch description' value to a new 'Merchnum' value, starting from the current maximum 'Merchnum' value in the DataFrame plus 1.

- It checks the number of null values for each column in the DataFrame to confirm that the 'Merchnum' column has no remaining null values after cleaning and imputation.

2. **Clean and impute State:**

- The code checks the number of missing values in the Merch state column and finds that there are 1020 missing values.

- It identifies the unique non-null zip codes for the transactions with missing Merch state values.

- It creates a dictionary zip_state that maps each of the zip codes to the corresponding state. For example, the zip code '00926' is mapped to 'PR' (Puerto Rico).

- It creates dictionaries merchnum_state and merchdes_state that map each non-null Merchnum and Merch description to the corresponding Merch state.

- It fills in the missing Merch state values by mapping each row's Merch zip, Merchnum, or Merch description to the corresponding Merch state using the dictionaries created in steps 3 and 4.

- It assigns 'unknown' to the Merch state values for the adjustment transactions.

- It changes the non-US states to 'foreign' and leaves missing values as they are.

- Finally, it checks the number of missing values in the Merch state column again and finds that there are 346 missing values.

3. **Clean and impute zip:**

- The missing values in zip are first identified using the isnull() function. The missing values in the ‘Merch zip’ column are 4300.

- Then, it creates two dictionaries merchnum_zip and merchdes_zip to map each unique value of ‘Merchnum’ and ‘Merch description' to its corresponding ‘Merch zip’. This is done by iterating through the non-null values of the ‘Merchnum’ and ‘Merch description’ columns and adding their respective ‘Merch zip’ values to the dictionaries if they are not already present in the dictionaries.

- Next, the missing values in the ‘Merch zip’ column are filled in by mapping the values in the ‘Merchnum’ and ‘Merch description’ columns to their corresponding "Merch zip" values using the dictionaries created earlier. This is done using the map() function.

- After this step, there are still 2658 missing values in the ‘Merch zip’ column. To handle this, it assigns the value ‘unknown’ to the missing values for transactions labeled as ‘RETAIL CREDIT ADJUSTMENT’ and ‘RETAIL DEBIT ADJUSTMENT’. This is done using the mask() function.

- Finally, it displays the first 50 rows where ‘Merch zip’ is missing to verify the imputation visually.

4. **New Cleaning/Imputation logic:**

- I have initialized a dictionary called 'state_zip' to hold non-null values of the 'Merch state' column as keys and the corresponding non-null values of one of the zip codes associated with that state.

- Using this dictionary, I have replaced any null values in the 'Merch zip' column with the appropriate zip code from the dictionary based on the 'Merch state' column's value.

Variable Creation

Description of how fraud occurs

In this fraud detection project, fraud occurs when unauthorized individuals gain access to credit card information and use it for illegitimate transactions. These fraudulent transactions typically exhibit abnormal patterns and behaviors when compared to genuine transactions. Some common indicators of fraud include unusual transaction amounts, sudden changes in transaction frequency, and transactions occurring in high-risk geographical locations.

To detect and prevent fraud, our project involves creating a variety of variables that capture these abnormal patterns in the dataset. By analyzing the relationships between these variables and the likelihood of fraud, our model can effectively identify potentially fraudulent transactions. Once flagged, these transactions can be further investigated or blocked to prevent financial losses for the organization.

The main objective of this project is to develop a robust fraud detection system that minimizes financial risks and protects the organization from fraudulent activities. To achieve this, we created multiple types of variables (184 in total) that cover different aspects of the transaction data:

1. **Frequency of transactions:** Represent the average fraud percentage for days of the week and months to capture any trends or seasonality in fraudulent activities.
2. **Geographic location of transactions:** Highlight the top 15 merchant states with the highest fraud percentages to account for regional risks.
3. **Time between transactions (Day Since, Velocity, Relative Velocity, Velocity Density):** Capture the frequency and patterns of transactions over different time periods to identify unusual activities.
4. **Cross-entity uniqueness:** Measure the number of unique combinations of entities in the dataset to uncover potential connections between different transaction elements.
5. **Entity amount variability:** Analyze the variability in transaction amounts for specific entities over time to detect abnormal spending patterns.
6. **Counts by Entity:** Track the number of unique entities within specific fields over various time periods to monitor any sudden changes in transaction patterns.
7. **Relative Velocity:** Evaluate the count of records and total amount with the same entities in relation to the square of the number of applications with those entities seen in recent time periods.

Table showing variable types and #

Description	# Variables Created
Date of week target encoded: average fraud percentage of that day	1
Month target encoded: average fraud percentage of that month	1
State Risk target encoded: Top 15 Fraud Merchant State with highest fraud percentage	1
Benford's Law Card number: count the # of first digits in card number	1
Benford's Law Merchant number: count the # of first digits in merchant number	1
Benford's Law Merchant Zip: count the # of first digits in merchant zip	1
Day Since: Number of days since application with that entity was seen	18
Velocity: Number of records with the same entity over the last {0,1,3,7,14,30,60} days	1134
Relative Velocity: Count of records and total amount with same entities seen in the past {0,1} day divided by the number of applications with those same entities seen in the last {3,7,14,30} days	288
Velocity Density: Count of records with same entities seen in the past {0,1} day divided by day since those same entities seen in the last {3,7,14,30} days	144
Cross Entity Uniqueness: Number of records with unique combinations of all the entities in the dataframe	306
Entity Amount Variability: Amount variability with the amount of same entity seen over the last {0,1,3,7,14,30,60} days	324
Counts by Entity: Number of unique entities for a particular fields over the last {0,1,3,7,14,30,60} days	1836
Relative Velocity (squared): Count of records and total amount with same entities seen in the past {0,1} day divided by square of the number of applications with those same entities seen in the last {3,7,14,30} days	144
Binning Amount: Creates 5 equal sized bins to divide amount column and assigns labels (1,5) to each bin	1

Feature selection

Description of what, why, how

Feature selection plays a crucial role in identifying the most relevant variables that contribute to detecting fraudulent transactions while reducing noise and improving model performance. The purpose of feature selection is to:

1. Simplify the model by removing irrelevant or redundant variables, thus making it easier to understand and interpret.
2. Reduce overfitting by minimizing the reliance on noisy or irrelevant features.
3. Improve computational efficiency by reducing the number of variables used in the model, speeding up the training and validation processes.

In this project, we employed multiple feature selection methods using Light Gradient Boosting Machine (LGBM) and Random Forest classifiers. These methods involved both forward and backward selection techniques, with different numbers of filters (num_filter) and wrappers (num_wrapper). By combining these methods, we aimed to identify the most important features for predicting fraudulent transactions.

The forward selection technique starts with an empty set of features and iteratively adds the most relevant feature at each step, based on its contribution to the model's performance. Conversely, the backward selection technique begins with a full set of features and iteratively removes the least important feature until the desired number of features is reached.

As a result of these feature selection methods, we identified a list of 20 final variables with their respective filter scores. These selected features have shown to be the most informative for detecting fraudulent transactions, and by incorporating them into our model, we can achieve better prediction accuracy and overall performance in identifying and preventing credit card fraud.

Methods

1. **LGBM Classifier with Forward Selection**
   num_filter = 200, num_wrapper = 20
   num_filter = 300, num_wrapper = 20
   num_filter = 400, num_wrapper = 20

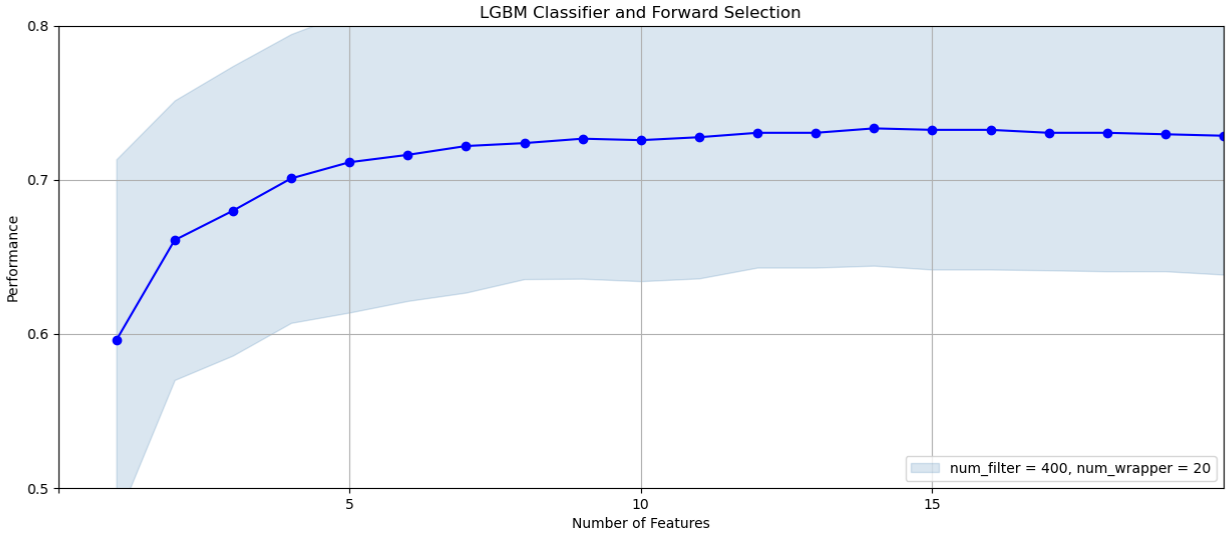
2. **LGBM Classifier with Backward Selection**
   num_filter = 100, num_wrapper = 20

3. **Random Forest Classifier with Forward Selection**
   num_filter = 200, num_wrapper = 20
   num_filter = 300, num_wrapper = 20
   num_filter = 400, num_wrapper = 20

List of final variables with filter score

wrapper order	variable	filter score
1	card_merch_total_14	0.630048056206397
2	card_zip3_max_14	0.6295145774877300
3	card_zip3_count_7	0.3878602480787210
4	Merchnum_desc_total_1	0.5284448838378500
5	Merchnum_desc_max_1	0.5236942252906410
6	Merchnum_desc_med_3	0.429393431315388
7	card_zip3_variability_max_3	0.3858683763465740
8	zip3_variability_avg_3	0.4050143980000460
9	merch_zip_total_14	0.44001853964965400
10	merch_zip_max_3	0.5144809550610280
11	Card_Merchnum_desc_total_60	0.5950188274379950
12	state_des_med_3	0.4255449473374220
13	Merchnum_desc_total_7	0.5171234063087220
14	merch_zip_max_1	0.522152980671316
15	card_merch_total_30	0.6154610858354430
16	Card_Merchnum_desc_total_30	0.6062795829714260
17	Card_Merchnum_Zip_total_30	0.6129313921901130
18	Card_Merchnum_Zip_total_14	0.6274209198898390
19	state_des_total_14	0.4908715461547040
20	Merchnum_desc_max_3	0.5168078939090770

Plot of performance vs # variables



Model Exploration

High-level description

In the Model Exploration phase, various machine learning models were tested to identify the one that performed best at detecting fraud. The models explored include:

1. **Logistic Regression**
2. **Decision Tree**
3. **Random Forest**
4. **Neural Network**
5. **Light Gradient Boosting Machine (LGBM) Classifier**
6. **XGBoost**

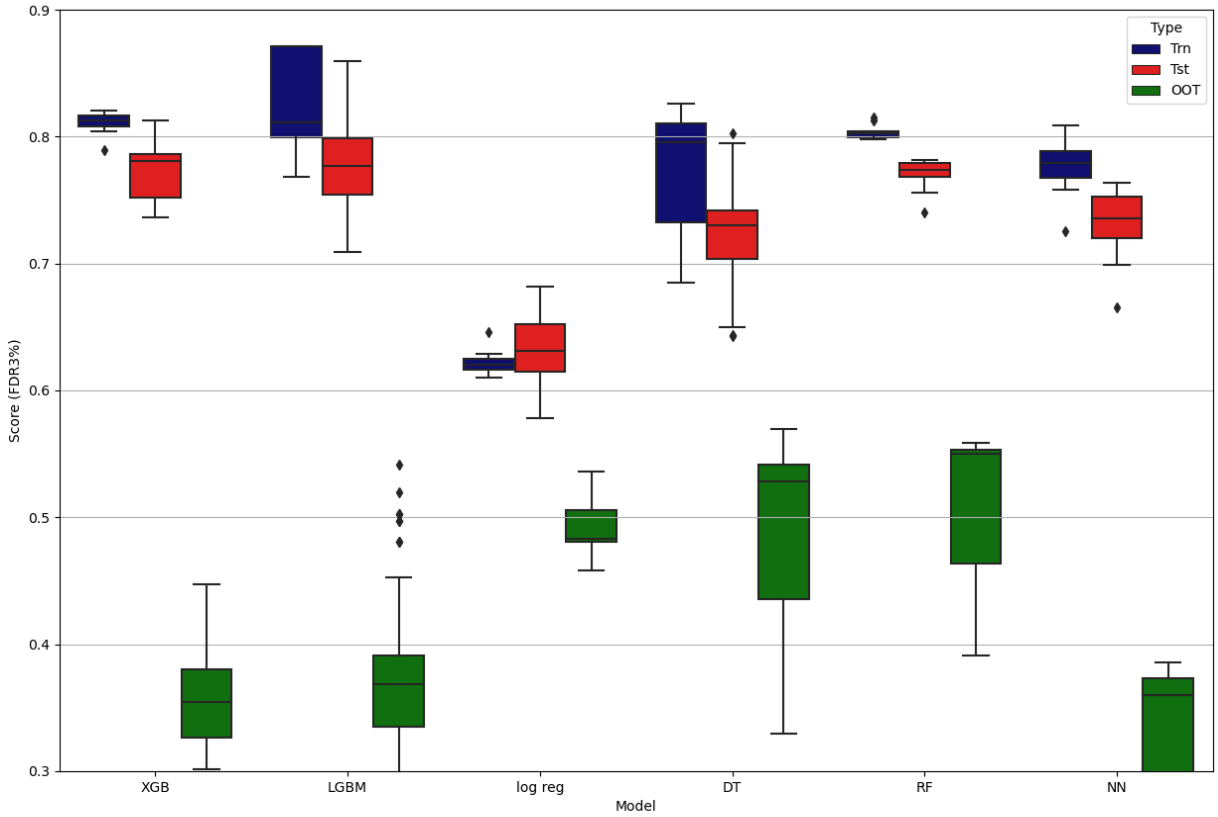
Each model was tested with different combinations of hyperparameters to optimize their performance. For each model, several iterations were conducted with different sets of parameters. The objective was to find the best combination of hyperparameters that would yield the highest average FDR at 3% for out dataset. The goal of this model exploration phase is to identify the model and the optimal set of parameters that provide the best performance in detecting fraudulent transactions, thus ensuring effective and efficient fraud prevention.

Table of tests

Model		Parameters						Average FDR at 3%		
Logistic Regression	Iteration	penalty	C	solver	l1_ratio		max_iter	Train	Test	OOT
	1	elasticnet	1	saga	0.5		1500	0.624867	0.615815	0.49162
	2	l1	1	liblinear	None		2500	0.617388	0.624968	0.475978
	3	l2	10	sag	None		2000	0.618721	0.61436	0.483799
	4	l2	10	newton-cg	None		2000	0.616405	0.621681	0.482123
	5	l2	0.1	newton-cg	None		2000	0.621024	0.617238	0.475978
	6	l1	0.1	saga	None		2500	0.625478	0.625934	0.493296
	7	l1	100	saga	None		2500	0.617217	0.619642	0.490503
	8	l2	10	lbfgs	None		4500	0.616221	0.628301	0.483799
Decision Tree	Iteration	criterion		splitter	max_depth	min_samples_split	min_samples_leaf	Train	Test	OOT
	1	gini		best	8	40	20	0.755479	0.717918	0.416201
	2	gini		best	10	100	40	0.797988	0.745372	0.493855
	3	entropy		best	10	100	50	0.830217	0.750007	0.434078
	4	gini		random	30	120	60	0.728195	0.690516	0.396648
	5	gini		best	10	100	50	0.800511	0.746124	0.439665
	6	entropy		random	10	100	50	0.705223	0.707514	0.270391
	7	gini		best	8	120	60	0.764513	0.71055	0.507263
	8	gini		best	8	90	50	0.768904	0.720124	0.507263
	9	gini		best	28	50	10	0.921006	0.724473	0.409497
Random Forest	Iteration	boostrap	criterion	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features	Train	Test	OOT
	1	TRUE	gini	100	8	120	60	sqrt	0.809151	0.769379	0.512291
	2	TRUE	entropy	100	8	120	60	sqrt	0.807182	0.765139	0.550279
	3	TRUE	gini	100	8	80	40	sqrt	0.813494	0.775923	0.479888
	4	TRUE	entropy	100	10	100	50	sqrt	0.837427	0.777048	0.558101
	5	TRUE	entropy	50	15	100	50	sqrt	0.855228	0.783621	0.52067
	6	TRUE	gini	50	None	100	50	log2	0.852952	0.793234	0.482682
	7	FALSE	gini	150	25	100	30	log2	0.870716	0.79027	0.412849
	8	FALSE	gini	150	35	120	20	sqrt	0.883037	0.803042	0.40838
	9	TRUE	gini	100	7	100	20	sqrt	0.803193	0.774611	0.537989
	10	TRUE	gini	200	27	100	1	sqrt	0.999836	0.812862	0.471508

	Iteration	activation	solver	learning_rate	alpha	learning_rate_init	batch_size	hidden_layer_size	max_iter	Train	Test	OOT
Nueral Network	1	relu	sgd	constant	0.01	None	auto	(100,)	200	0.619258	0.614538	0.329609
	2	relu	sgd	constant	0.001	None	auto	(100,)	500	0.618472	0.622582	0.319553
	3	relu	sgd	constant	0.0001	0.001	auto	(100,)	200	0.782887	0.72274	0.348045
	4	relu	sgd	constant	0.0001	0.01	auto	(200,)	500	0.773537	0.716709	0.773537
	5	relu	sgd	adaptive	0.0001	0.001	auto	(100,)	200	0.775993	0.730024	0.327933
	6	relu	sgd	adaptive	1	0.03	auto	(200,)	300	0.537184	0.539292	0.260121
LGBM Classifier	Iteration	boosting_type	n_estimator	max_depth	learning_rate	num_leaves	subsample	colsample_bytree		Train	Test	OOT
	1	gbdt	200	23	0.05	30	0.5	1		0.994853	0.827198	0.34581
	2	gbdt	80	3	0.05	40	0.5	1		0.807898	0.772485	0.386592
	3	gbdt	100	-1	0.1	31	1	1		0.997568	0.793298	0.31676
	4	gbdt	500	7	0.05	50	0.8	0.8		1	0.817303	0.309497
	5	gbdt	90	3	0.05	30	0.5	1		0.808564	0.761725	0.377654
XGBoost	Iteration	booster	n_estimator	max_depth	min_child_weight	eta	tree_method	subsample	colsample_bytree	Train	Test	OOT
	1	gbtree	15	4	7	0.4	auto	1	0.5	0.810716	0.772675	0.357542
	2	gbtree	100	6	1	0.3	auto	1	1	1	0.816645	0.29162
	3	gbtree	15	4	7	0.4	approx	1	0.5	0.792273	0.764359	0.380447
	4	gbtree	15	8	5	0.05	auto	0.8	0.8	0.776215	0.758858	0.553631
	5	dart	20	10	5	0.08	auto	0.8	0.8	0.804949	0.751255	0.52514

“Box plot” of trn/tst/oot for several models



Final Model Performance

Description of final model hyperparameters

The final model chosen for this fraud detection project is the **XGBoost Classifier**, which is an ensemble learning method that builds gradient-boosted decision trees. XGBoost is known for its excellent performance and speed in classification problems such as fraud detection.

The hyperparameters of the final model are as follows:

1. **booster: 'gbtree'**, which indicates that the model uses gradient-boosted decision trees as base learners.
2. **n_estimators: 15**, the number of trees in the model. It represents the number of boosting rounds to be run, with each round adding a new tree to the ensemble.
3. **max_depth: 8**, the maximum depth of each tree. This controls the complexity of the trees and helps prevent overfitting.
4. **min_child_weight: 5**, the minimum sum of instance weight (hessian) needed in a child node. This parameter is used to control the minimum number of instances required to create a new node in the tree and helps prevent overfitting.
5. **colsample_bytree: 0.8**, the fraction of features to be randomly selected for each tree. This value helps in reducing overfitting by introducing randomness in the feature selection process.
6. **subsample: 0.8**, the fraction of the training data to be randomly selected for each boosting round. This value helps in reducing overfitting by introducing randomness in the data selection process.
7. **eta: 0.05**, the learning rate for the boosting process. This parameter controls the contribution of each tree to the final prediction, helping to prevent overfitting.
8. **tree_method: 'auto'**, which allows the model to automatically select the best tree construction algorithm based on the input data and the chosen hyperparameters.

The final model's performance metrics, as measured by the Average FDR at 3%, are:

Train: 76.40%

Test: 76.26%

OOT: 56.98%

These performance metrics indicate that the final XGBoost model has a good performance in detecting fraudulent transactions on both the training and testing datasets. The OOT is comparatively lower, which may suggest that the model is not as effective in capturing fraud patterns in unseen data.

Three summary tables

Training	# Records	# Goods	# Bads	Fraud Rate
	58779	58161	618	0.010513959

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	588	186	402	31.63%	68.37%	588	186	402	0.32%	65.05%	64.73	0.46
2	588	526	62	89.46%	10.54%	1176	712	464	1.22%	75.08%	73.86	1.53
3	587	568	19	96.76%	3.24%	1763	1280	483	2.20%	78.16%	75.95	2.65
4	588	573	15	97.45%	2.55%	2351	1853	498	3.19%	80.58%	77.40	3.72
5	588	587	1	99.83%	0.17%	2939	2440	499	4.20%	80.74%	76.55	4.89
6	588	579	9	98.47%	1.53%	3527	3019	508	5.19%	82.20%	77.01	5.94
7	588	583	5	99.15%	0.85%	4115	3602	513	6.19%	83.01%	76.82	7.02
8	587	583	4	99.32%	0.68%	4702	4185	517	7.20%	83.66%	76.46	8.09
9	588	582	6	98.98%	1.02%	5290	4767	523	8.20%	84.63%	76.43	9.11
10	588	581	7	98.81%	1.19%	5878	5348	530	9.20%	85.76%	76.57	10.09
11	588	586	2	99.66%	0.34%	6466	5934	532	10.20%	86.08%	75.88	11.15
12	587	585	2	99.66%	0.34%	7053	6519	534	11.21%	86.41%	75.20	12.21
13	588	585	3	99.49%	0.51%	7641	7104	537	12.21%	86.89%	74.68	13.23
14	588	586	2	99.66%	0.34%	8229	7690	539	13.22%	87.22%	73.99	14.27
15	588	585	3	99.49%	0.51%	8817	8275	542	14.23%	87.70%	73.47	15.27
16	588	585	3	99.49%	0.51%	9405	8860	545	15.23%	88.19%	72.95	16.26
17	587	580	7	98.81%	1.19%	9992	9440	552	16.23%	89.32%	73.09	17.10
18	588	584	4	99.32%	0.68%	10580	10024	556	17.23%	89.97%	72.73	18.03
19	588	584	4	99.32%	0.68%	11168	10608	560	18.24%	90.61%	72.38	18.94
20	588	586	2	99.66%	0.34%	11756	11194	562	19.25%	90.94%	71.69	19.92

Testing	# Records	# Goods	# Bads	Fraud Rate
	25191	24929	262	0.010400540

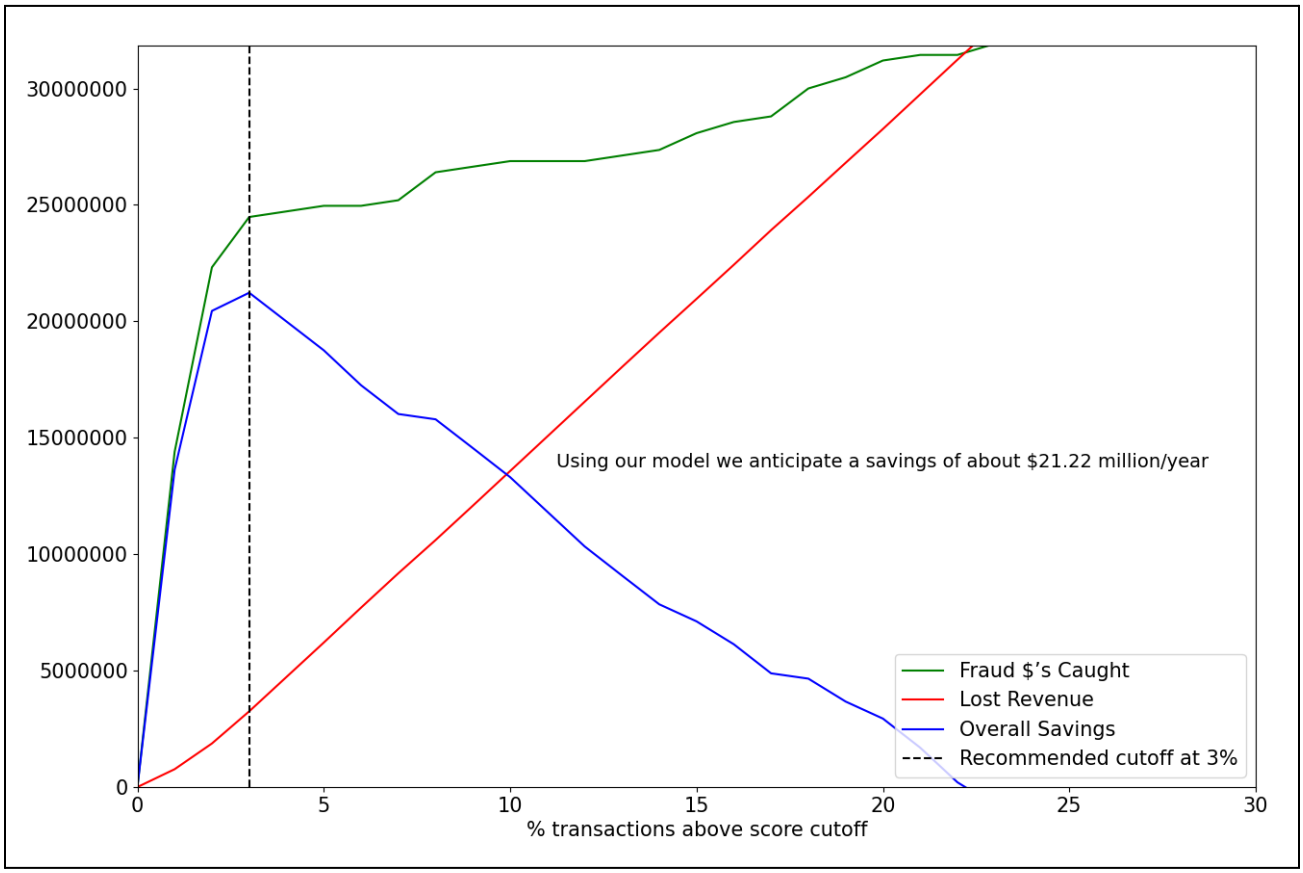
	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	252	94	158	37.30%	62.70%	252	94	158	0.38%	60.31%	59.93	0.59
2	252	230	22	91.27%	8.73%	504	324	180	1.30%	68.70%	67.40	1.80
3	252	239	13	94.84%	5.16%	756	563	193	2.26%	73.66%	71.41	2.92
4	252	245	7	97.22%	2.78%	1008	808	200	3.24%	76.34%	73.09	4.04
5	252	249	3	98.81%	1.19%	1260	1057	203	4.24%	77.48%	73.24	5.21
6	251	248	3	98.80%	1.20%	1511	1305	206	5.23%	78.63%	73.39	6.33
7	252	248	4	98.41%	1.59%	1763	1553	210	6.23%	80.15%	73.92	7.40
8	252	252	0	100.00%	0.00%	2015	1805	210	7.24%	80.15%	72.91	8.60
9	252	250	2	99.21%	0.79%	2267	2055	212	8.24%	80.92%	72.67	9.69
10	252	249	3	98.81%	1.19%	2519	2304	215	9.24%	82.06%	72.82	10.72
11	252	252	0	100.00%	0.00%	2771	2556	215	10.25%	82.06%	71.81	11.89
12	252	250	2	99.21%	0.79%	3023	2806	217	11.26%	82.82%	71.57	12.93
13	252	251	1	99.60%	0.40%	3275	3057	218	12.26%	83.21%	70.94	14.02
14	252	252	0	100.00%	0.00%	3527	3309	218	13.27%	83.21%	69.93	15.18
15	252	250	2	99.21%	0.79%	3779	3559	220	14.28%	83.97%	69.69	16.18
16	252	250	2	99.21%	0.79%	4031	3809	222	15.28%	84.73%	69.45	17.16
17	251	250	1	99.60%	0.40%	4282	4059	223	16.28%	85.11%	68.83	18.20
18	252	251	1	99.60%	0.40%	4534	4310	224	17.29%	85.50%	68.21	19.24
19	252	251	1	99.60%	0.40%	4786	4561	225	18.30%	85.88%	67.58	20.27
20	252	252	0	100.00%	0.00%	5038	4813	225	19.31%	85.88%	66.57	21.39

OOT	# Records	# Goods	# Bads	Fraud Rate
	12427	12248	179	0.014404120

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	124	65	59	52.42%	47.58%	124	65	59	0.53%	32.96%	32.43	1.10
2	125	93	32	74.40%	25.60%	249	158	91	1.29%	50.84%	49.55	1.74
3	124	115	9	92.74%	7.26%	373	273	100	2.23%	55.87%	53.64	2.73
4	124	121	3	97.58%	2.42%	497	394	103	3.22%	57.54%	54.33	3.83
5	124	122	2	98.39%	1.61%	621	516	105	4.21%	58.66%	54.45	4.91
6	125	125	0	100.00%	0.00%	746	641	105	5.23%	58.66%	53.43	6.10
7	124	124	0	100.00%	0.00%	870	765	105	6.25%	58.66%	52.41	7.29
8	124	120	4	96.77%	3.23%	994	885	109	7.23%	60.89%	53.67	8.12
9	124	119	5	95.97%	4.03%	1118	1004	114	8.20%	63.69%	55.49	8.81
10	125	123	2	98.40%	1.60%	1243	1127	116	9.20%	64.80%	55.60	9.72
11	124	124	0	100.00%	0.00%	1367	1251	116	10.21%	64.80%	54.59	10.78
12	124	122	2	98.39%	1.61%	1491	1373	118	11.21%	65.92%	54.71	11.64
13	125	125	0	100.00%	0.00%	1616	1498	118	12.23%	65.92%	53.69	12.69
14	124	123	1	99.19%	0.81%	1740	1621	119	13.23%	66.48%	53.25	13.62
15	124	121	3	97.58%	2.42%	1864	1742	122	14.22%	68.16%	53.93	14.28
16	124	123	1	99.19%	0.81%	1988	1865	123	15.23%	68.72%	53.49	15.16
17	125	122	3	97.60%	2.40%	2113	1987	126	16.22%	70.39%	54.17	15.77
18	124	124	0	100.00%	0.00%	2237	2111	126	17.24%	70.39%	53.16	16.75
19	124	124	0	100.00%	0.00%	2361	2235	126	18.25%	70.39%	52.14	17.74
20	124	120	4	96.77%	3.23%	2485	2355	130	19.23%	72.63%	53.40	18.12

Financial Curves and Recommended Cutoff

Plot with three curves



Recommendation for a cutoff and Description of the logic

We recommending score cutoff at 3%. The recommendation is based on the goal of maximizing overall savings while balancing the trade-off between fraud savings and false positive (FP) losses. By choosing a cutoff at the 3% level, the model aims to capture the highest proportion of fraud cases while minimizing the number of false positive transactions.

The performance of the final model is evaluated by calculating Fraud Detection Rate (FDR), which is the number of true positive frauds identified divided by the total number of frauds. The financial impact is then assessed by computing Fraud Savings (savings from identifying and preventing fraud), FP Loss (costs incurred due to false positives), and Overall Savings (difference between Fraud Savings and FP Loss).

When examining the 'Overall Savings' curve, the recommended cutoff at 3% is chosen because it represents the point where the overall savings are maximized. At this point, the model is able to

effectively identify and prevent a significant portion of fraudulent transactions, while minimizing the costs associated with false positives.

By choosing a score cutoff at 3%, the model achieves a balance between detecting fraud and reducing the number of false alarms, which ultimately leads to an anticipated savings of about $21.22 million per year.

Summary

In this project, we started with data cleaning and preprocessing to ensure that the dataset was ready for analysis. We addressed missing values, outliers, and data inconsistencies, as well as transformed categorical variables into numerical ones using techniques like one-hot encoding. We also excluded certain variables that were deemed irrelevant or had high multicollinearity.

Next, we focused on variable creation and feature engineering to generate new features that could potentially improve the performance of our models. To achieve this, we created multiple types of variables (184 in total) that cover different aspects of the transaction data like Date-related, Geographic, Time-based (Day Since, Velocity, Relative Velocity, Velocity Density), Cross-entity uniqueness, Entity amount variability, Counts by Entity and Relative Velocity variables.

Following this, we conducted a feature selection process to identify the most important variables for our model. We employed multiple feature selection methods using LGBM and Random Forest classifiers. These methods involved both forward and backward selection techniques, with different numbers of filters and wrappers. By combining these methods, we aimed to identify the most important features for predicting fraudulent transactions.

With the selected features, we explored various machine learning models, including Logistic Regression, Decision Trees, Random Forests, Neural Networks, LightGBM, and XGBoost. We experimented with different hyperparameters and configurations to find the best performing model. After comparing the models based on their performance metrics, we chose the XGBoost classifier as our final model.

We then evaluated the final model's performance on the train, test, and out-of-time (OOT) datasets. We measured the model's Fraud Detection Rate (FDR) at 3% and assessed its financial impact through the calculation of Fraud Savings, FP Loss, and Overall Savings.

We plotted the financial curves for the model, and based on the analysis, recommended a score cutoff at 3%. At this cutoff, the model achieved an FDR of 56.98% for the OOT dataset, resulting in an anticipated savings of about $21.22 million per year.

Although the XGBoost classifier has demonstrated promising results, there are additional steps that could be taken to potentially improve its performance and financial impact:

1. Advanced Feature engineering: Explore new features or transformations of existing variables to improve the model's predictive power.
2. Model stacking or ensembling: Combine multiple models or use different algorithms to capture complementary patterns in the data and improve overall performance.
3. Hyperparameter tuning: Perform more extensive hyperparameter search or use optimization algorithms like Bayesian Optimization to further refine the model.
4. Regular model updates: Retrain the model periodically with the most recent data to ensure it stays relevant and adapts to new patterns in fraudulent transactions.

Statement of the model performance

The final model is a XGBoost classifier, which demonstrated superior performance in terms of the Fraud Detection Rate (FDR) at 3% compared to other models explored.

The model's performance was evaluated using a train, test, and out-of-time (OOT) dataset. The primary performance metric was the Fraud Detection Rate (FDR) at 3%. The final XGBoost model achieved the following results:

Train: 76.40%

Test: 76.26%

OOT: 56.98%

In technical terms, the model showed a relatively consistent performance on both the training and testing datasets, indicating its ability to generalize well on unseen data. While the OOT dataset performance was lower compared to the training and testing datasets, the model still demonstrated an impressive FDR@3% of 56.98%. This suggests that the model can effectively detect fraudulent transactions in real-world scenarios, even with the inherent uncertainty in future data.

In business terms, the model's financial impact was analyzed to estimate the potential savings that could be achieved by implementing the XGBoost model with a recommended score cutoff at 3%. The analysis revealed an anticipated savings of about $21.22 million per year, showcasing the model's effectiveness in detecting and preventing fraudulent transactions. This substantial cost reduction for the company highlights the value of utilizing a well-performing machine learning model for fraud detection and prevention in the financial sector.

The performance of the model on the three datasets, as well as its financial impact, demonstrate the robustness and efficiency of the XGBoost classifier in addressing the problem of fraud detection. The model's strong results make it a suitable choice for deployment in a real-world setting, contributing to significant cost savings and improved fraud prevention measures for the organization.

Appendix

Data Quality Report

1. Data Description

The dataset is **Credit Card Transaction Data**, which contains real credit card transaction records from a US government organization for business purposes. The data came from the website of author Mark Nigrini over a **one-year period** from January 1, 2010, to December 31, 2010. There are **10 fields** and a total of **96,753 records**.

2. Summary Tables

Numeric Fields Table

Field Name	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Most Common	Stdev
Date	96,753	100.00%	0	1/1/10	12/31/10	6/25/10	2/28/10	98 days 21:38
Amount	96,753	100.00%	0	0.01	3102045.53	427.88	3.62	10006.14

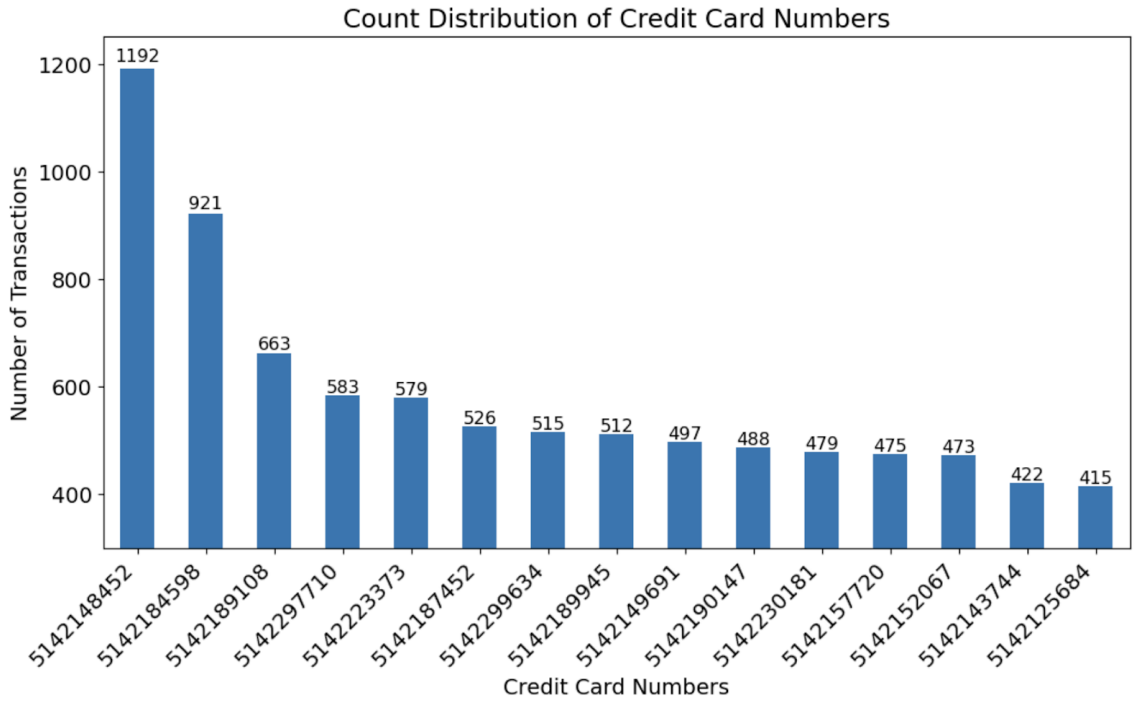
Categorical Fields Table

Field Name	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Recnum	96,753	100.00%	0	96,753	1
Cardnum	96,753	100.00%	0	1,645	5142148452
Merchnum	93,378	96.51%	0	13,091	930090121224
Merch description	96,753	100.00%	0	13,126	GSA-FSS-ADV
Merch state	95,558	98.76%	0	227	TN
Merch zip	92,097	95.19%	0	4,567	38118
Transtype	96,753	100.00%	0	4	P
Fraud	96,753	100.00%	95,694	2	0

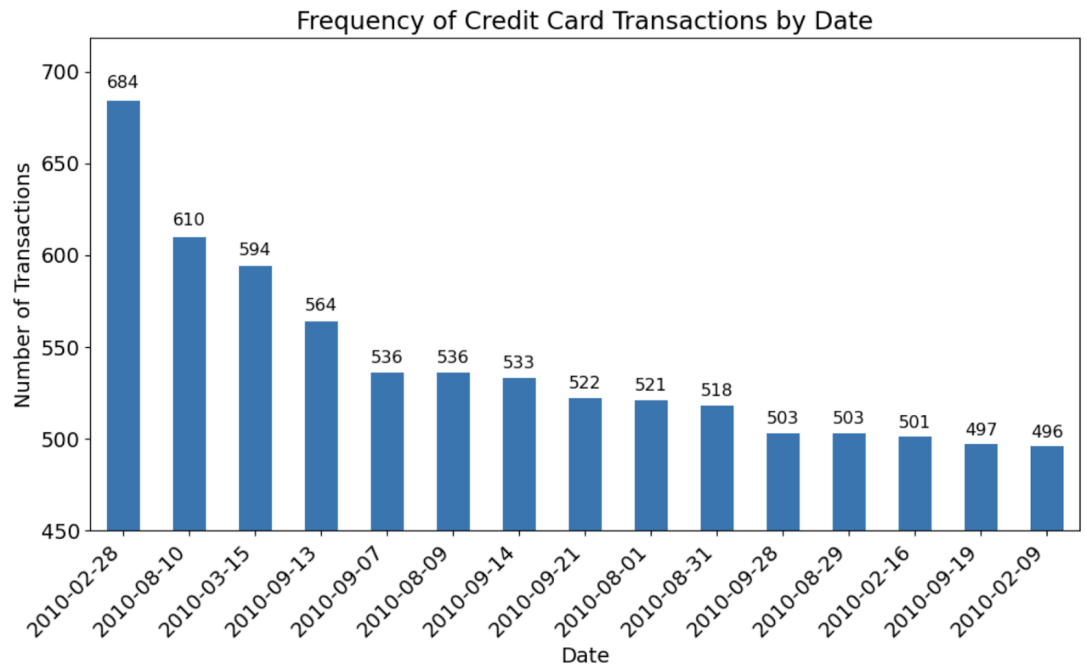
3. Visualization of Each Field

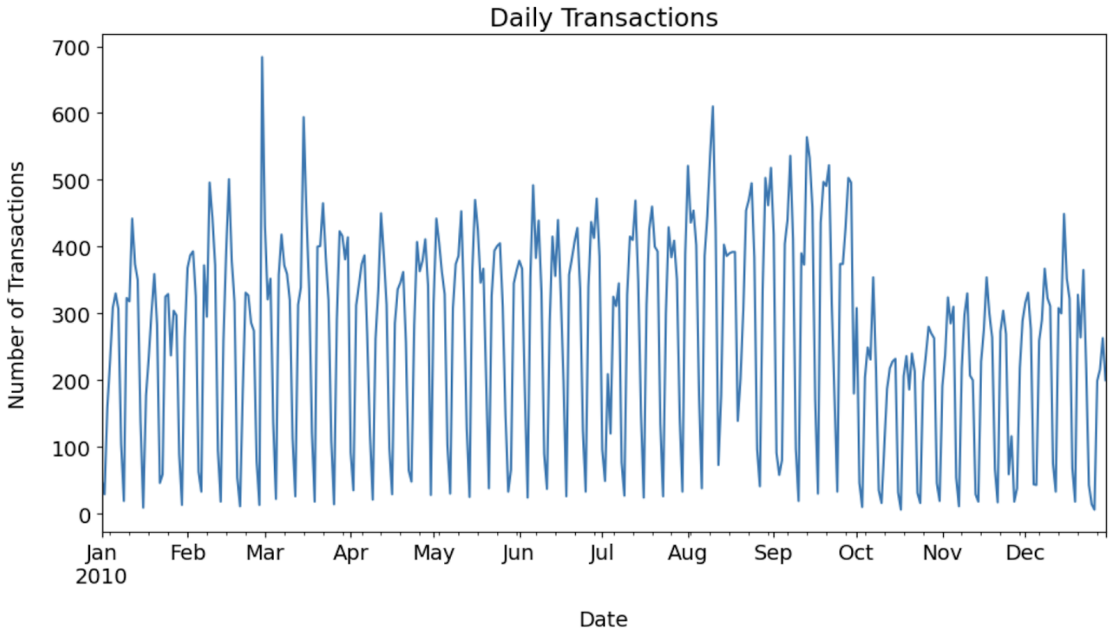
1) **Field Name: Recnum**
   Description: Record Number. A unique identifier for each transaction record, sequentially from 0 to 96,752.

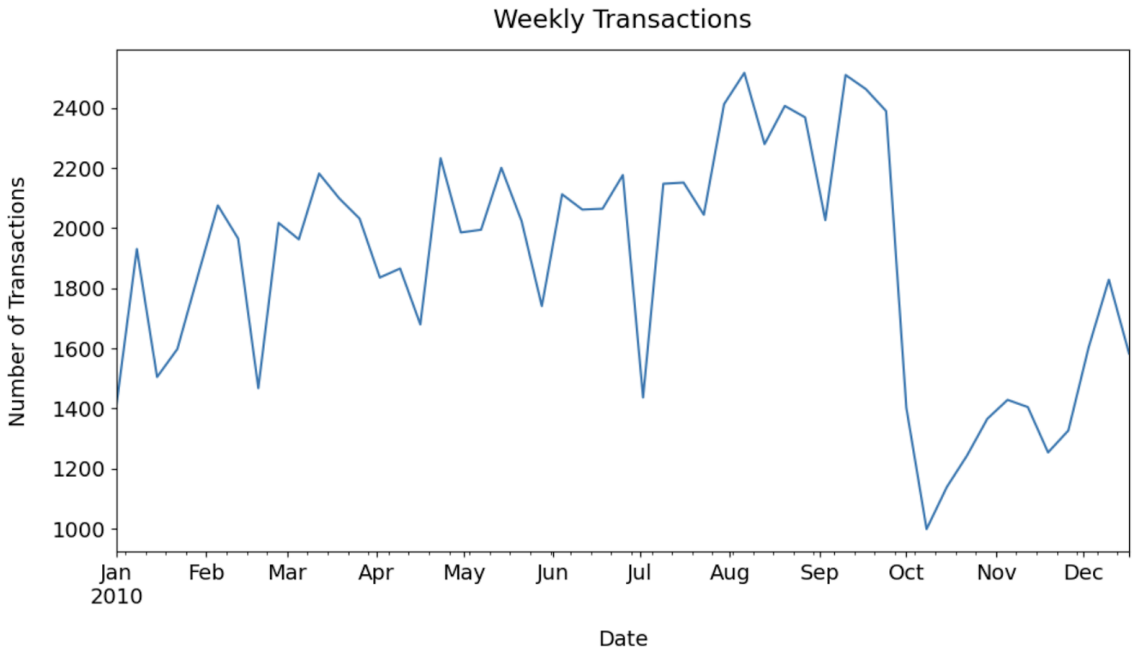
2) **Field Name: Cardnum**
   Description: Credit Card Number. 1,645 distinct values. The distribution shows the top 15 field values of Cardnum. The most common card number is 5142148452, with a total count of 1,192.

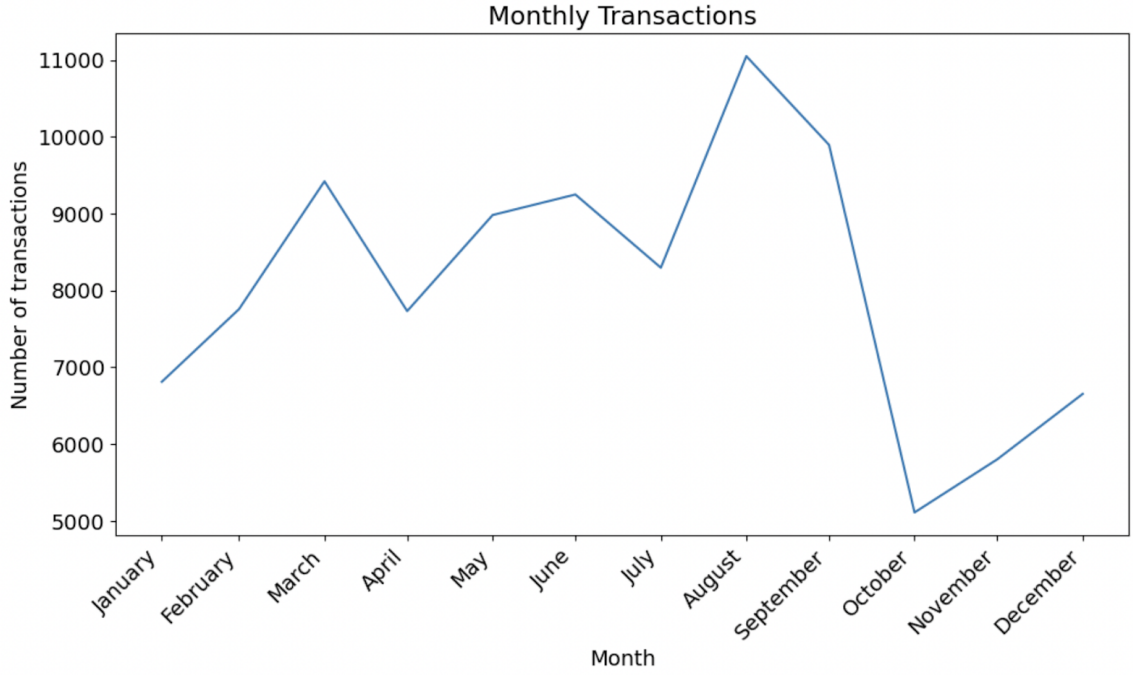


3) **Field Name: Date**
   Description: Date on which transaction took place. The first distribution shows the maximum number of transactions (684) took place on February 28, 2010. The subsequent distributions show the trend of daily transactions, weekly transactions and monthly transactions across time.


Frequency of Credit Card Transactions by Date
Number of Transactions
Date
700
650
600
550
500
450
684
610
594
564
536
536
533
522
521
518
503
503
501
497
496
2010-02-28
2010-08-10
2010-03-15
2010-09-13
2010-09-07
2010-08-09
2010-09-14
2010-09-21
2010-08-01
2010-08-31
2010-09-28
2010-08-29
2010-02-16
2010-09-19
2010-02-09

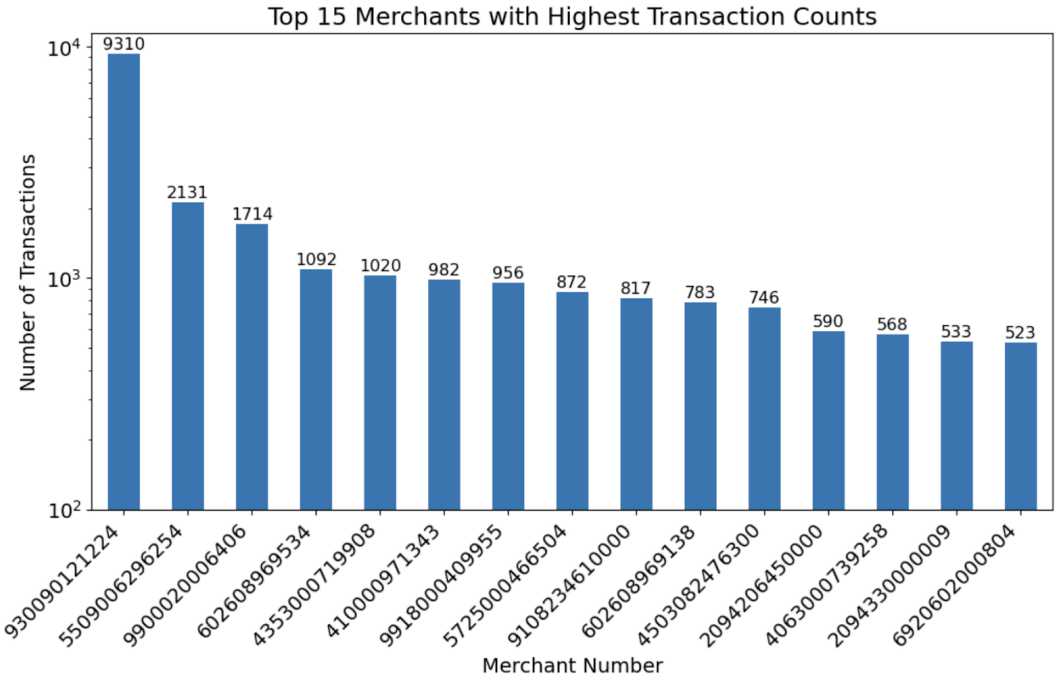

Daily Transactions
Number of Transactions
Date
700
600
500
400
300
200
100
0
Jan
2010
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec





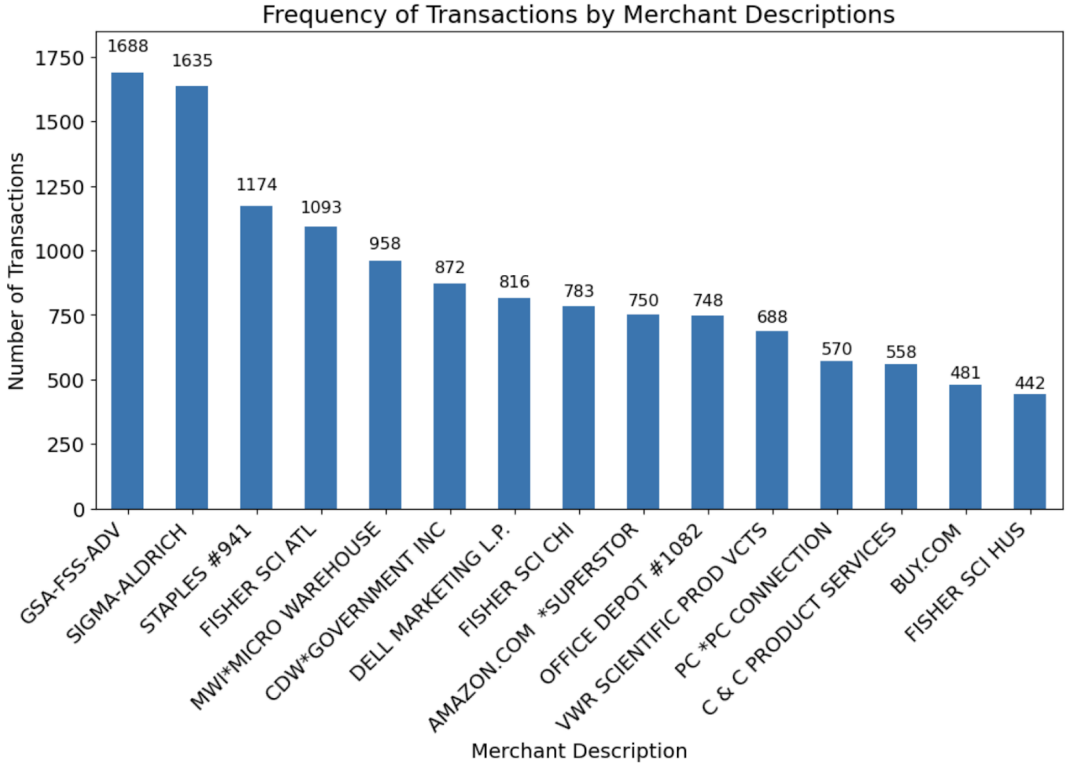
4) Field Name: Merchnum

Description: Merchant Number. The distribution shows the top 15 field values of Merchnum with highest transaction count. The most common merchant number is 930090121224, with a total count of 9,310.



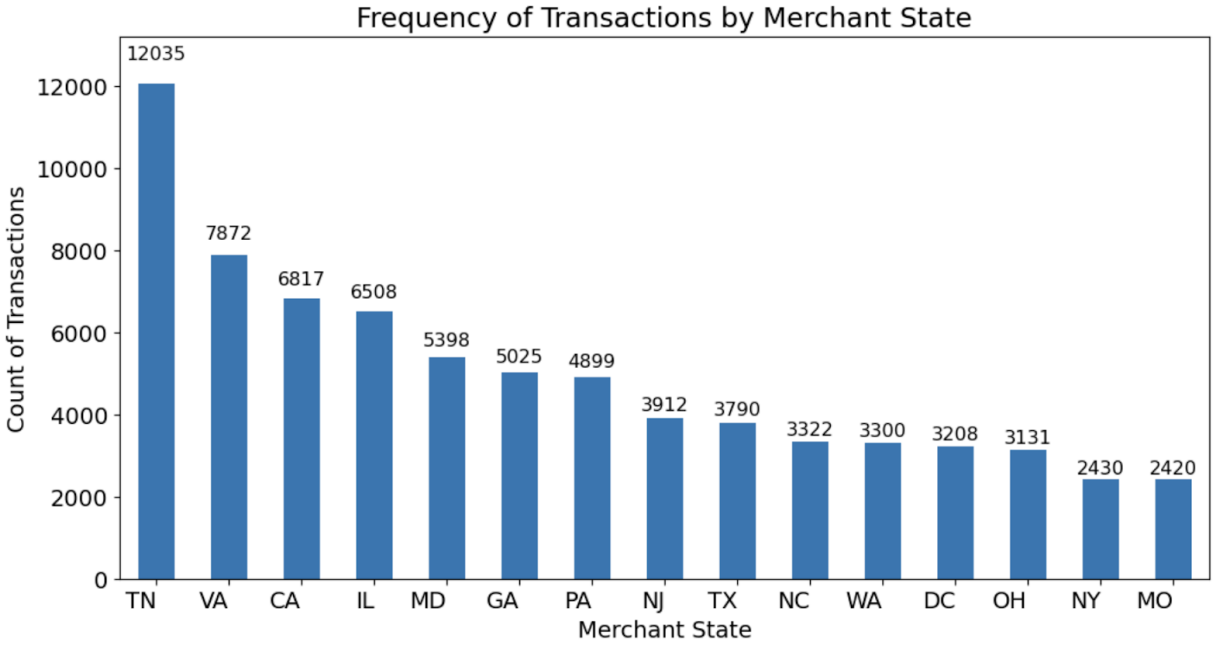
5) Field Name: Merch description

Description: Merchant's Description. The distribution shows the top 15 field values of Merchant's Description. The most common last name is 'GSA-FSS-ADV', with a total count of 1,688.



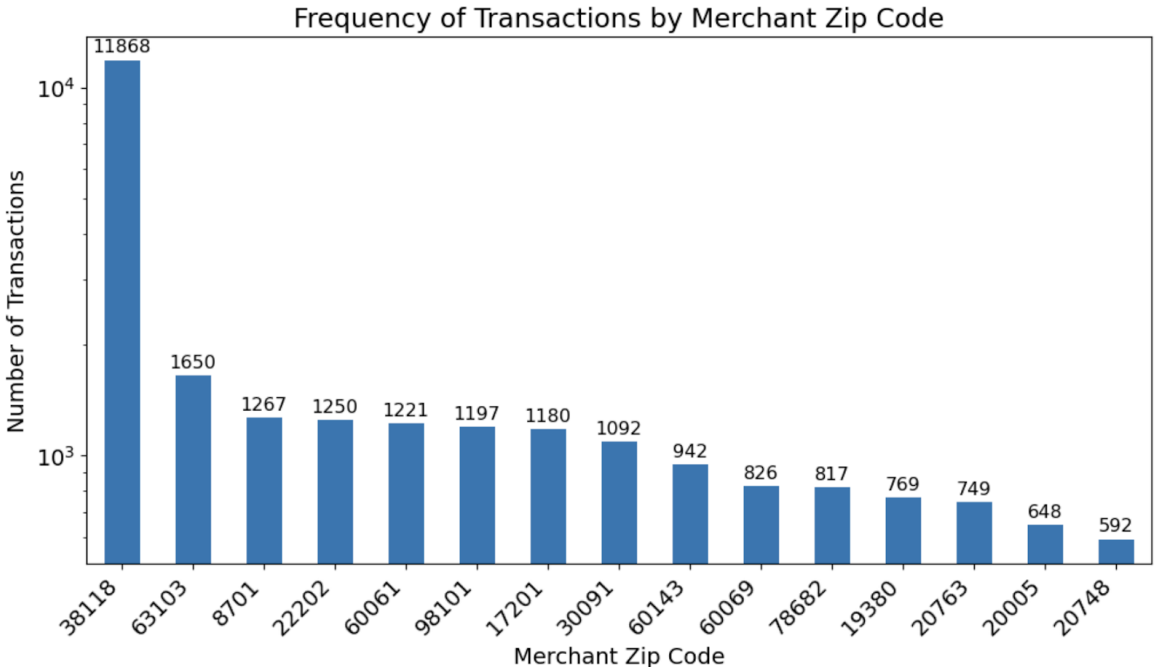
6) **Field Name: Merch state**

Description: Merchant's State. The distribution shows the top 15 field values of Merchant's State. The most common address is 'TN', with a total count of 12,035.



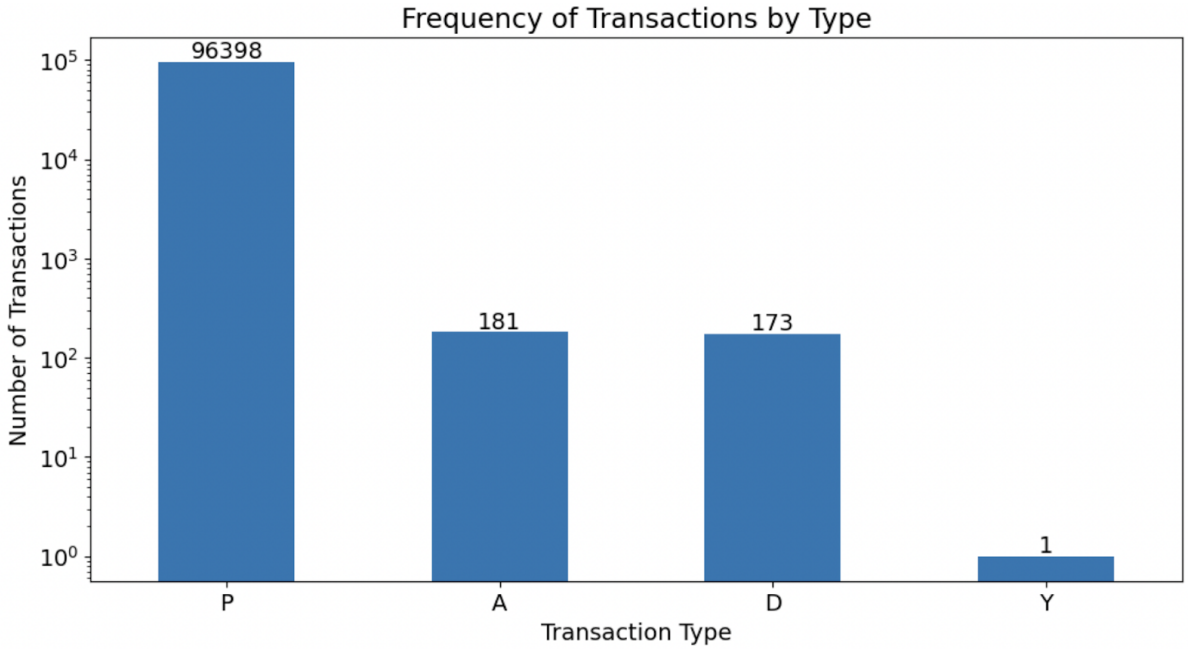
7) **Field Name: Merch zip**

Description: Merchant's zip code. The distribution shows the top 15 field values of a merchant's zip code. The most common zip code is 38118, with a total count of 11,868.



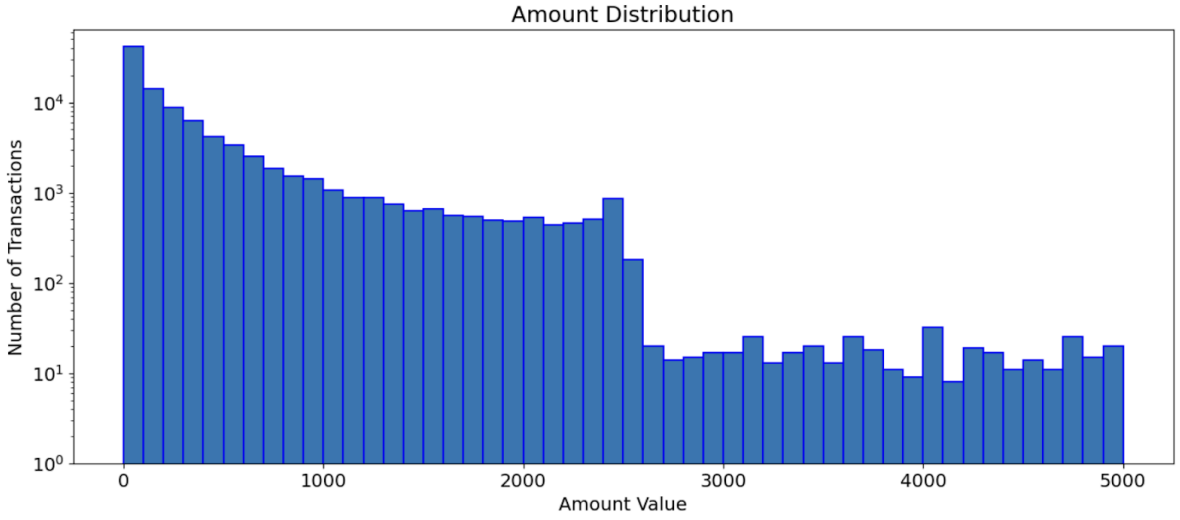
8) **Field Name: Transtype**

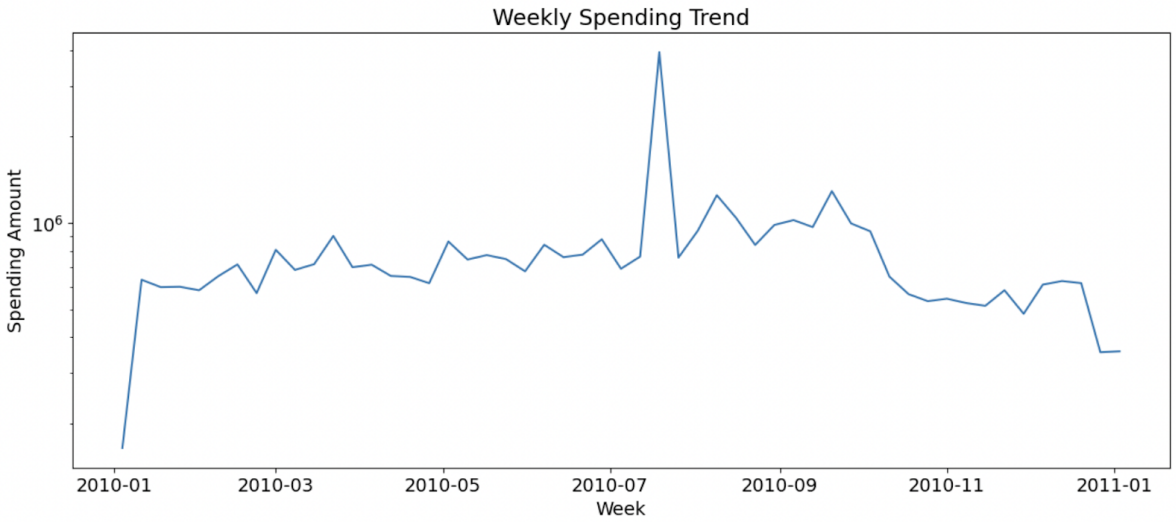
Description: Transaction Type. The distribution shows the top 15 field values of transaction types. The most common transaction type is P, with a total count of 96,398.



9) **Field Name: Amount**

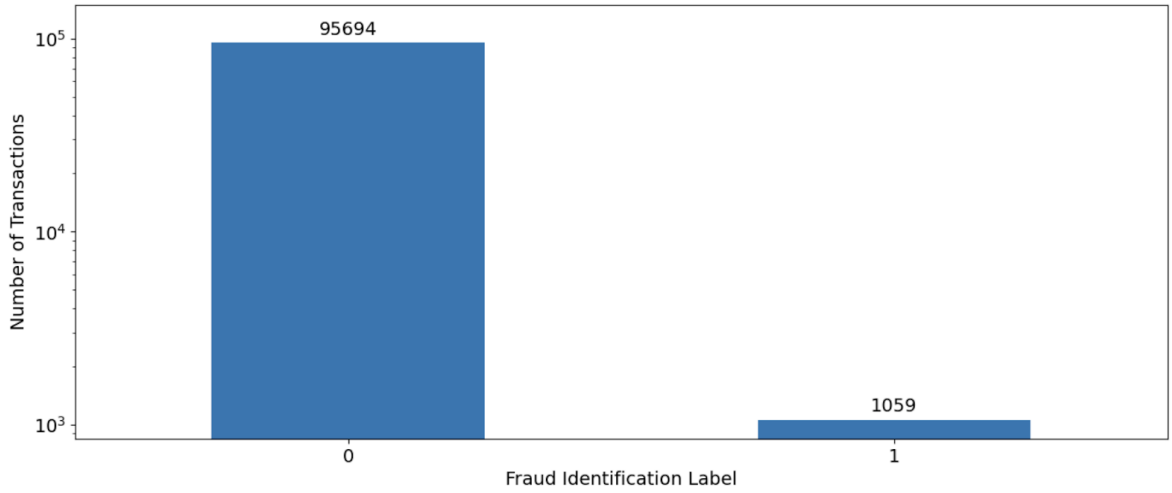
Description: Amount of Transaction Value in dollars. The first distribution shows the number of transactions vs amount. The second distribution shows the weekly spending trend.

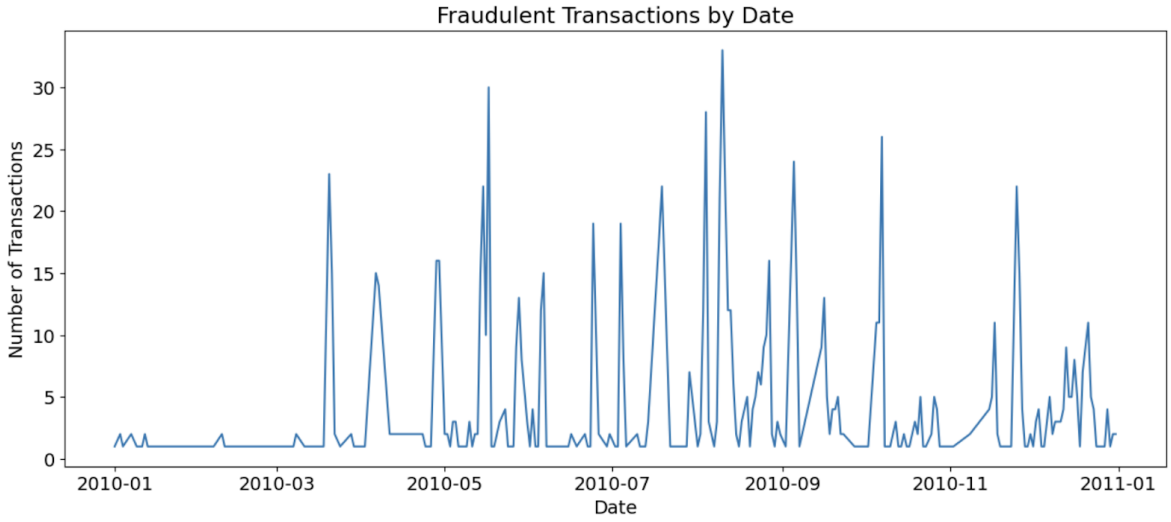


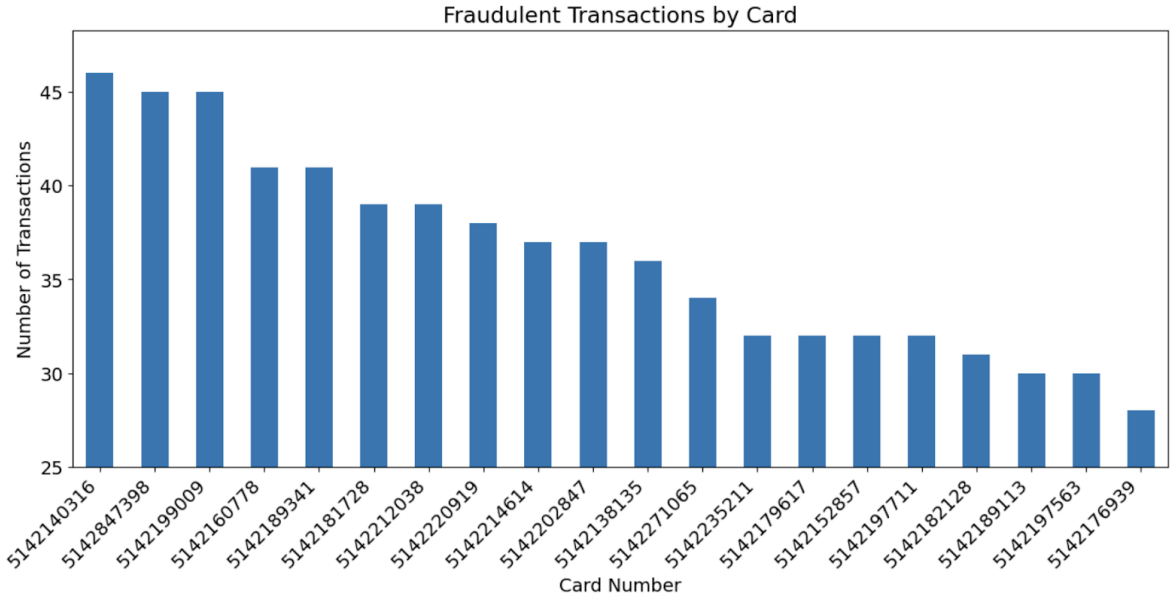


10) Field Name: Fraud

Description: Fraud identification label. Fraud = 0 (Not fraudulent), Fraud =1 (Fraud identified). The total count fraud = 0 is 95,694. The total count of fraud = 1 is 1059. The other two distributions show fraudulent transactions trend across days and card numbers.




Fraudulent Transactions by Date
Number of Transactions
0
5
10
15
20
25
30
2010-01
2010-03
2010-05
2010-07
2010-09
2010-11
2011-01
Date


Fraudulent Transactions by Card
Number of Transactions
25
30
35
40
45
5142140316
5142847398
5142199009
5142160778
5142189341
5142181728
5142212038
5142220919
5142214614
5142202847
5142138135
5142271065
5142235211
5142179617
5142152857
5142197711
5142182128
5142189113
5142197563
5142176939
Card Number