1. Executive summary

Based on the analysis of card transaction data, we have developed a predictive model to identify and prevent fraud. Our model has been trained using advanced statistical techniques and has been tested for accuracy using out-of-time data. The model uses a combination of variables and algorithms to accurately identify fraudulent transactions. By rejecting only 3% of the applications, we can eliminate about 50% of the fraudulent activity, resulting in potential savings of up to 21 million. This model can be used by businesses to prevent fraud and minimize losses, making it an essential tool for protecting financial transactions.

2. Data Description

The data is about **Card Transaction data**. The data is about company card transactions for some US government organization. The dataset is from Mark Nigrini's website and made up 1000 fraud records. There are **10 fields** and **96,753 records** (data has 8 fields of unnamed which was not considered here).

(1). Numerical Table

Field Name	% Populated	Min	Max	Mean	Std	% Zero
Date	100.00	2010-01-01	2010-12-31	/	/	0.00
Amount	100.00	0.01	3,102,045.53	427.89	10,006.14	0.00

(2). Categorical Table

Field Name	% Populated	# Unique values	Most Common Value
Recnum	100.00	96,753	/
Cardnum	100.00	1,645	5142148452
Merchnum	96.51	13,091	930090121224
Merch description	100.00	13,126	GSA-FSS-ADV
Merch state	98.76	227	TN
Merch zip	95.19	4,567	38118
Transtype	100.00	4	P

Fraud	100.00	2	0

3. Data Cleaning

I first clean my data by remove one outlier and keep only P's transtype. There are three columns of missing fields: I use reasonable values to fill them first and then will fill with 'unknown'. For Merchnum and zip column, I used the index to fill it first. For state column, if they have valid zip code, I will fill this column with the state name of that zip code. There are sample codes for my cleaning part, I did not include all because they follow the same logic. And you could see that there are a lot of nulls in this dataset, it needs to be cleaned before using it.

```
In [4]: #remove the non-p, single outlier
        data = data.drop(data[(data['Transtype'] != 'P')].index)
        data = data.drop(data[(data['Amount'] >= 1000000)].index)
        data.shape
Out[4]: (96397, 10)
```

clean and impute merchnum

```
In [5]: data['Merchnum'].isnull().sum()
Out[5]: 3198

In [6]: merchdes_merchnum = {}
        for index, merchdes in data[data['Merch description'].notnull()]['Merch description'].items():
            if merchdes not in merchdes_merchnum :
                merchdes_merchnum[merchdes] = data.loc[index, 'Merchnum']

        # fill in by mapping with Merch description
        data['Merchnum'] = data['Merchnum'].fillna(data['Merch description'].map(merchdes_merchnum))

        # assign unknown for adjustments transactions
        data['Merchnum'] = data['Merchnum'].mask(data['Merch description'] == 'RETAIL CREDIT ADJUSTMENT', 'unknown')
        data['Merchnum'] = data['Merchnum'].mask(data['Merch description'] == 'RETAIL DEBIT ADJUSTMENT', 'unknown')

In [7]: data.Merchnum.isna().sum()
Out[7]: 2215

In [8]: data['Merchnum'] = data['Merchnum'].fillna('unknown')

In [9]: data['Merchnum'].isnull().sum()
Out[9]: 0
```

4. Variable Creation

After I got my cleaned dataset, I implemented day of week variable and make a risk table to get a brief idea the fraud proportion of each weekday. Then I build the Benford's law variables based on the formula for the merchnum and cardnum in order to get the unusual activities. I also make new entities based on current features: add cardnum and merchnum to other features. Based on the original features and new entities, I make new variables by count the day since of the entity and the average/max/med of amount over certain time of the entity. After I got the day since variable and the velocity variable, I can get the ratio of them and got the relative velocity variable. Those variables can flag out the unusual transactions which might lead to fraud. Like people with the larger amount of transaction and transaction frequency very low but used frequently recently. Also, to figure out some stolen card transaction and people who use other's information, we try to create a unique entity variable to get an insight of the uniqueness. Change of the amount is also a good sign to catch an infrequent recurring charge of same amount or unusual transaction amount.
After creating variables, we need to get rid of the categorial feature and check if there are replicated features, then we got all the candidate variables we created.

The total amount of variables I created after dedup is: 1392.
Before dedup is 1620.

Description of variables	#Variables created
Original fields from the dataset: 'Amount' field.	1
Date of week target encoded: (dow_risk) Average fraud percentage of that weekday	1
Benford's law variables: U* variable for cardnum and merchnum based on benford's law.	2
Day Since Variable: # days since an transaction with that entity has been seen.	10
Frequency Variable: # records with the same entity over the last {0,1,3,7,14,30,60}	70
Amount Variable: Average/max/median/total/ amounts of the same entity over the last {0,1,3,7,14,30,60} and the ratio of actual/(mean, max, median, total)	560
Velocity change variable: This set of Variables is the fraction of the number of applications with the same entity over the past 0 or 1 day out of the daily number of records with the same entity for 3,7,14,30 days	80
Velocity Day Since Ratio Variable: This set of variables is the ratio of relative velocity by day since for an entity	80
Unique Entity Number Variable: This set of variables is the amount of an entity that is unique with other entity.	540
Variability Variables: This set of variables include maximum, minimum, median amount change of the same entity over the last {0,1,3,7,14,30}	180
Amount bin Variable: Equally population into 5 bins	1
Acceleration Variable: The ratio of the entity over the past 0 or 1 day out of the daily squared number of records for 3,7,14,30 days	80

5. Feature Selection

Then we do feature selection to get the most related variables for accuracy, which simply the model complexity and reduce overfitting. We first filter out about half of the variables using two filters KS and FDR, and then use forward selection using LGBM to select 20 variables. The variable with higher KS score will be the variable that separates the frauds from non-frauds better. Fraud detection rate tells us what % of all the fraud are caught at a particular examination cutoff location. A higher FDR indicates that the variable might do a better job in catching the frauds.

	wrapper order	variable	filter score
0	1	Card_Merchdesc_total_7	0.668332
1	2	Merchnum_desc_max_60	0.446950
2	3	Cardnum_max_3	0.553510
3	4	Merchnum_total_1	0.597729
4	5	merch_zip_total_7	0.571736
5	6	Merchnum_desc_total_7	0.568648
6	7	Merchnum_total_14	0.481583
7	8	merch_zip_avg_0	0.576255
8	9	Merchnum_total_7	0.577117
9	10	merch_zip_total_0	0.570794
10	11	Merchnum_desc_total_14	0.539233
11	12	merch_zip_total_14	0.476467
12	13	zip3_actual/max_14	0.422684
13	14	Merchnum_desc_total_0	0.573366
14	15	Merchnum_total_0	0.572679
15	16	amount_cat	0.541046
16	17	zip3_total_1	0.496322
17	18	Cardnum_avg_60	0.471712
18	19	Card_Merchdesc_med_0	0.558399
19	20	zip3_total_0	0.529874

6. Model Exploration

After I got my final variable list, I try different model and hyperparameters to find which one fits better by looking at the fdr @ 3% of its train, test, out of time. Then I built 5 machine learning models to classify the records. For each combination of hyperparameters, we run 5 times and average them to get a result, which helps to reduce the impact of randomness. By adjusting the hyperparameters, I find that some combination of hyperparameters got overfitting (those daker color in each model result). Thus, I could use the best model with the hyperparameters to get the result.

Model	Parameter						FDR @ 3%			
Logistic Regression	# vars	max_iter					trn	tst	oot	
1	10	20					0.638	0.608	0.434	
2	15	20					0.576	0.602	0.263	
3	20	20					0.61	0.588	0.287	
Single Decision Tree	# vars	max_depth	min_samples_split	min_samples_leaf						
1	10	5	50	30			0.671	0.638	0.444	
2	10	10	40	25			0.795	0.729	0.506	
3	15	15	30	20			0.899	0.724	0.471	overfit
4	15	20	20	15			0.944	0.723	0.372	overfit
5	20	30	10	10			0.995	0.729	0.42	overfit
Rondom Forest	# vars	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features				
1	10	5	5	50	30	3	0.701	0.697	0.502	
2	10	10	10	50	25	10	0.811	0.769	0.554	overfit
3	15	20	15	30	20	10	0.902	0.801	0.545	overfit
4	15	40	20	20	15	12	0.974	0.821	0.553	overfit
5	20	60	25	10	10	15	0.999	0.825	0.56	overfit
Boosted Tree(LGBM)	# vars	num_leaves	n_estimator							
1	10	2	5				0.588	0.561	0.328	
2	10	4	10				0.742	0.716	0.571	
3	15	6	20				0.791	0.756	0.521	
4	15	8	40				0.856	0.789	0.438	overfit
5	20	100	100				0.999	0.825	0.411	overfit
Catboost	# vars	max_depth	iterations							
1	10	2	5				0.662	0.649	0.476	
2	10	4	10				0.695	0.686	0.536	
3	15	6	20				0.765	0.735	0.549	
4	15	8	40				0.827	0.769	0.545	overfit
5	20	10	80				0.975	0.806	0.417	overfit
Neural Net	# vars	hidden_layer_sizes	activation	alpha	learning_rate	solver				
1	10	(2,)	logistic	0.1	constant	adam	0.628	0.618	0.344	
2	10	(10,10)	logistic	0.01	constant	adam	0.646	0.625	0.445	
3	15	(20,20,20)	logistic	0.001	constant	adam	0.722	0.727	0.533	
4	15	(2,)	relu	0.1	adaptive	lbfgs	0.637	0.625	0.375	
5	20	(20,20,20)	relu	0.001	adaptive	lbfgs	0.818	0.752	0.411	overfit

7. Final Model Performance

The final model I use is CatBoost, which has max_depth as 6 and iterations as 15. This combination of hyperparameters gives the most high oot. Based on the three results table, we could see that the number of goods/bads and the fraud rate for training, testing and out of time.

Training	#records	#goods	#bads	fraud rate
	58772	58155	617	0.010498

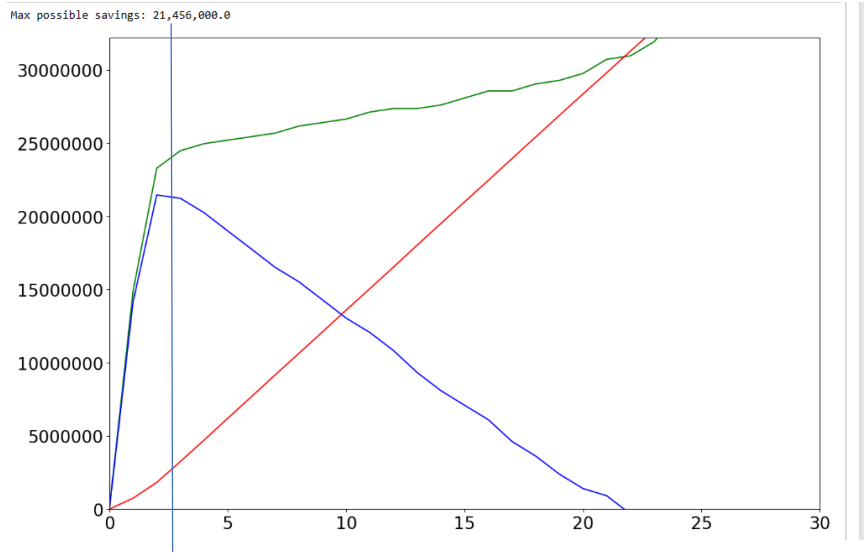
	Bin Statistics					Cumulative Statistics						
Population	#records	#goods	#bads	%goods	%bads	total #rec	cumulative	cumulative	%goods	FDR	KS	FPR
1	588	220	368	37.41497	62.58503	588	220	368	0.378299	59.64344	59.26514	0.597826
2	587	520	67	88.58603	11.41397	1175	740	435	1.272462	70.50243	69.22997	1.701149
3	588	551	37	93.70748	6.292517	1763	1291	472	2.219929	76.49919	74.27926	2.735169
4	588	574	14	97.61905	2.380952	2351	1865	486	3.206947	78.76823	75.56129	3.837449
5	588	579	9	98.46939	1.530612	2939	2444	495	4.202562	80.2269	76.02434	4.937374
6	587	575	12	97.95571	2.044293	3526	3019	507	5.191299	82.1718	76.9805	5.954635
7	588	577	11	98.12925	1.870748	4114	3596	518	6.183475	83.95462	77.77114	6.942085
8	588	586	2	99.65986	0.340136	4702	4182	520	7.191127	84.27877	77.08764	8.042308
9	587	583	4	99.31857	0.681431	5289	4765	524	8.19362	84.92707	76.73345	9.093511
10	588	577	11	98.12925	1.870748	5877	5342	535	9.185797	86.70989	77.52409	9.985047
11	588	587	1	99.82993	0.170068	6465	5929	536	10.19517	86.87196	76.67679	11.06157
12	588	587	1	99.82993	0.170068	7053	6516	537	11.20454	87.03404	75.8295	12.13408
13	587	585	2	99.65928	0.340716	7640	7101	539	12.21047	87.35818	75.14771	13.1744
14	588	584	4	99.31973	0.680272	8228	7685	543	13.21468	88.00648	74.7918	14.15285
15	588	578	10	98.29932	1.70068	8816	8263	553	14.20858	89.62723	75.41865	14.94213
16	588	584	4	99.31973	0.680272	9404	8847	557	15.21279	90.27553	75.06273	15.8833
17	587	584	3	99.48893	0.511073	9991	9431	560	16.21701	90.76175	74.54474	16.84107
18	588	585	3	99.4898	0.510204	10579	10016	563	17.22294	91.24797	74.02504	17.79041
19	588	587	1	99.82993	0.170068	11167	10603	564	18.23231	91.41005	73.17774	18.79965
20	587	585	2	99.65928	0.340716	11754	11188	566	19.23824	91.7342	72.49596	19.76678

Testing	#records	#goods	#bads	fraud rate
	25188	24925	263	0.010441

	Bin Statistics					Cumulative Statistics						
Population	#records	#goods	#bads	%goods	%bads	total #rec	cumulative	cumulative	%goods	FDR	KS	FPR
1	252	94	158	37.30159	62.69841	252	94	158	0.377131	60.07605	59.69891	0.594937
2	252	217	35	86.11111	13.88889	504	311	193	1.247743	73.38403	72.13629	1.611399
3	252	240	12	95.2381	4.761905	756	551	205	2.210632	77.94677	75.73614	2.687805
4	252	245	7	97.22222	2.777778	1008	796	212	3.193581	80.60837	77.41478	3.754717
5	251	246	5	98.00797	1.992032	1259	1042	217	4.180542	82.50951	78.32896	4.801843
6	252	248	4	98.4127	1.587302	1511	1290	221	5.175527	84.03042	78.85489	5.837104
7	252	250	2	99.20635	0.793651	1763	1540	223	6.178536	84.79087	78.61234	6.90583
8	252	249	3	98.80952	1.190476	2015	1789	226	7.177533	85.93156	78.75403	7.915929
9	252	251	1	99.60317	0.396825	2267	2040	227	8.184554	86.31179	78.12723	8.986784
10	252	252	0	100	0	2519	2292	227	9.195587	86.31179	77.1162	10.09692
11	252	251	1	99.60317	0.396825	2771	2543	228	10.20261	86.69202	76.48941	11.15351
12	252	251	1	99.60317	0.396825	3023	2794	229	11.20963	87.07224	75.86261	12.20087
13	251	251	0	100	0	3274	3045	229	12.21665	87.07224	74.85559	13.29694
14	252	252	0	100	0	3526	3297	229	13.22768	87.07224	73.84456	14.39738
15	252	248	4	98.4127	1.587302	3778	3545	233	14.22267	88.59316	74.37049	15.21459
16	252	250	2	99.20635	0.793651	4030	3795	235	15.22568	89.35361	74.12794	16.14894
17	252	250	2	99.20635	0.793651	4282	4045	237	16.22869	90.11407	73.88538	17.06751
18	252	252	0	100	0	4534	4297	237	17.23972	90.11407	72.87435	18.1308
19	252	251	1	99.60317	0.396825	4786	4548	238	18.24674	90.4943	72.24756	19.10924
20	252	252	0	100	0	5038	4800	238	19.25777	90.4943	71.23652	20.16807

OOT	#records		#goods		#bads		fraud rate						
	12473		12258		179		0.014351						
	Bin Statistics						Cumulative Statistics						
Population	#records	#goods	#bads	%goods	%bads		total #rec	cumulative	cumulative	%goods	FDR	KS	FPR
1	124	62	62	50	50		124	62	62	0.505792	34.63687	34.13108	1
2	125	90	35	72	28		249	152	97	1.240007	54.18994	52.94994	1.56701
3	124	119	5	95.96774	4.032258		373	271	102	2.210801	56.98324	54.77244	2.656863
4	124	122	2	98.3871	1.612903		497	393	104	3.20607	58.10056	54.89449	3.778846
5	125	124	1	99.2	0.8		622	517	105	4.217654	58.65922	54.44156	4.92381
6	124	123	1	99.19355	0.806452		746	640	106	5.22108	59.21788	53.9968	6.037736
7	125	124	1	99.2	0.8		871	764	107	6.232664	59.77654	53.54387	7.140187
8	124	122	2	98.3871	1.612903		995	886	109	7.227933	60.89385	53.66592	8.12844
9	124	123	1	99.19355	0.806452		1119	1009	110	8.231359	61.45251	53.22115	9.172727
10	125	124	1	99.2	0.8		1244	1133	111	9.242943	62.01117	52.76823	10.20721
11	124	122	2	98.3871	1.612903		1368	1255	113	10.23821	63.12849	52.89028	11.10619
12	124	123	1	99.19355	0.806452		1492	1378	114	11.24164	63.68715	52.44551	12.08772
13	125	125	0	100	0		1617	1503	114	12.26138	63.68715	51.42577	13.18421
14	124	123	1	99.19355	0.806452		1741	1626	115	13.26481	64.24581	50.981	14.13913
15	125	123	2	98.4	1.6		1866	1749	117	14.26823	65.36313	51.0949	14.94872
16	124	122	2	98.3871	1.612903		1990	1871	119	15.2635	66.48045	51.21695	15.72269
17	124	124	0	100	0		2114	1995	119	16.27509	66.48045	50.20536	16.76471
18	125	123	2	98.4	1.6		2239	2118	121	17.27851	67.59777	50.31925	17.50413
19	124	123	1	99.19355	0.806452		2363	2241	122	18.28194	68.15642	49.87449	18.36885
20	124	122	2	98.3871	1.612903		2487	2363	124	19.27721	69.27374	49.99654	19.05645

8. Financial curves and recommended cutoff



The green curve is how much fraud is caught, the blue curve is overall saving, and the red line is lost revenue.

Estimate saving: 21 million

Recommend cutoff point: 3%

9. Summary

We using the card transaction dataset and by first do a data exploration report, we notice that there are some invalid transtype, outliers and nulls. By removing invalid data and fill the null with reasonable value, the dataset is more useful to build a model. We use the cleaned dataset to create variables based on what we are targeting at like frequency, amount and count etc. Because we need to reduce the complexity and avoid overfitting, we do feature selection to choose most related variables to make the model more accuracy. We use filter and forward selection with LGBM to get the KS. Based on the KS score we choose some variables to build the model.

In order to get the best performance, we try different model with different parameters. By averaging the results of several runs for each trail, we reduce the impact of randomness. After choosing the best performance model, we can get three results table for training, testing and out of time. Based on what we already got, we can come up a table about the predicted fraud, lost revenues and overall saving. By looking at the table, we come a recommended cutoff about 3 %, giving us a estimated savings of 21 million.

The result we look at the oot fdr @ 3%, we can conclude that our model will eliminate about 50% of the fraud by rejecting only 3% of the applications and estimated saving amount is 21 million.

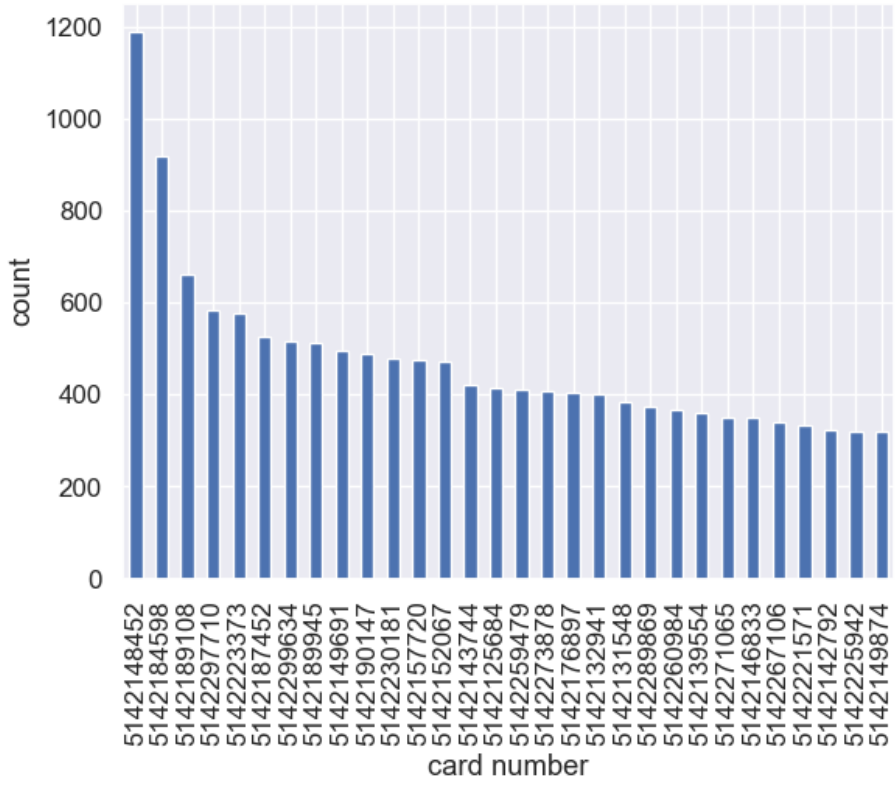
Appendix -- DQR

(1). Field Name: recnum

Description: Record Number: Ordinal unique positive integer for each record. From 1 to 96,753.

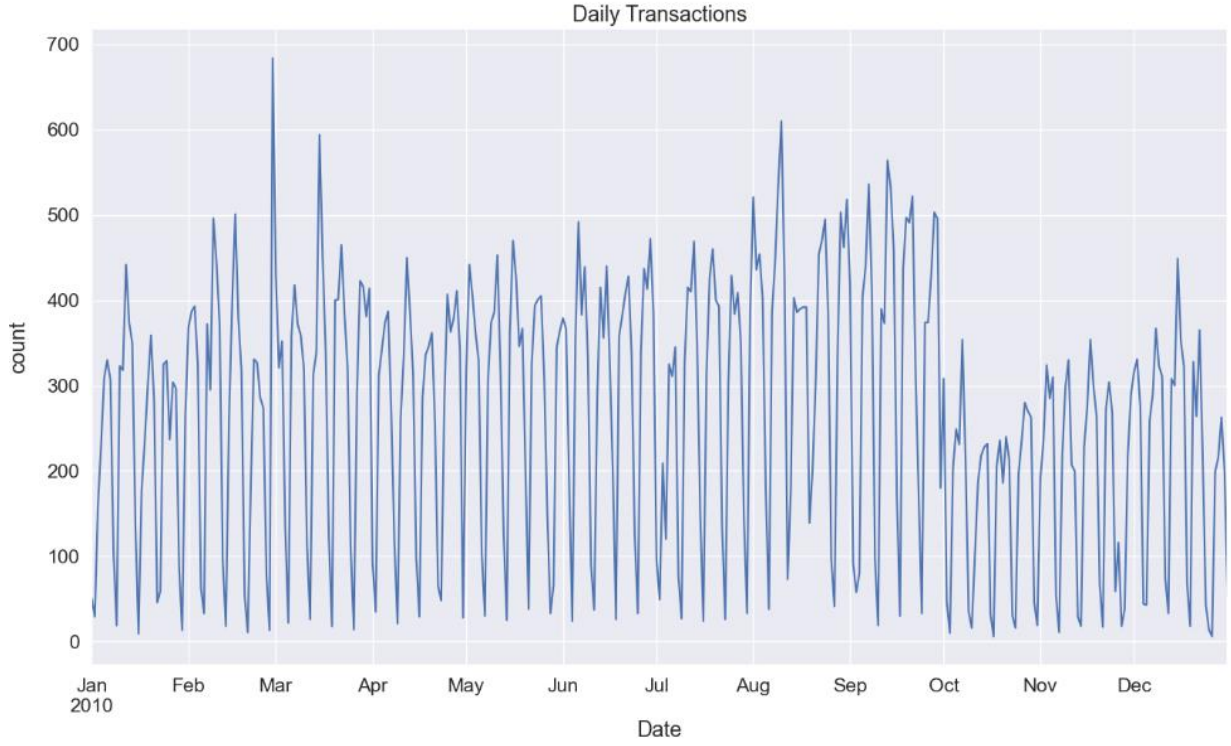
(2). Field Name: Cardnum

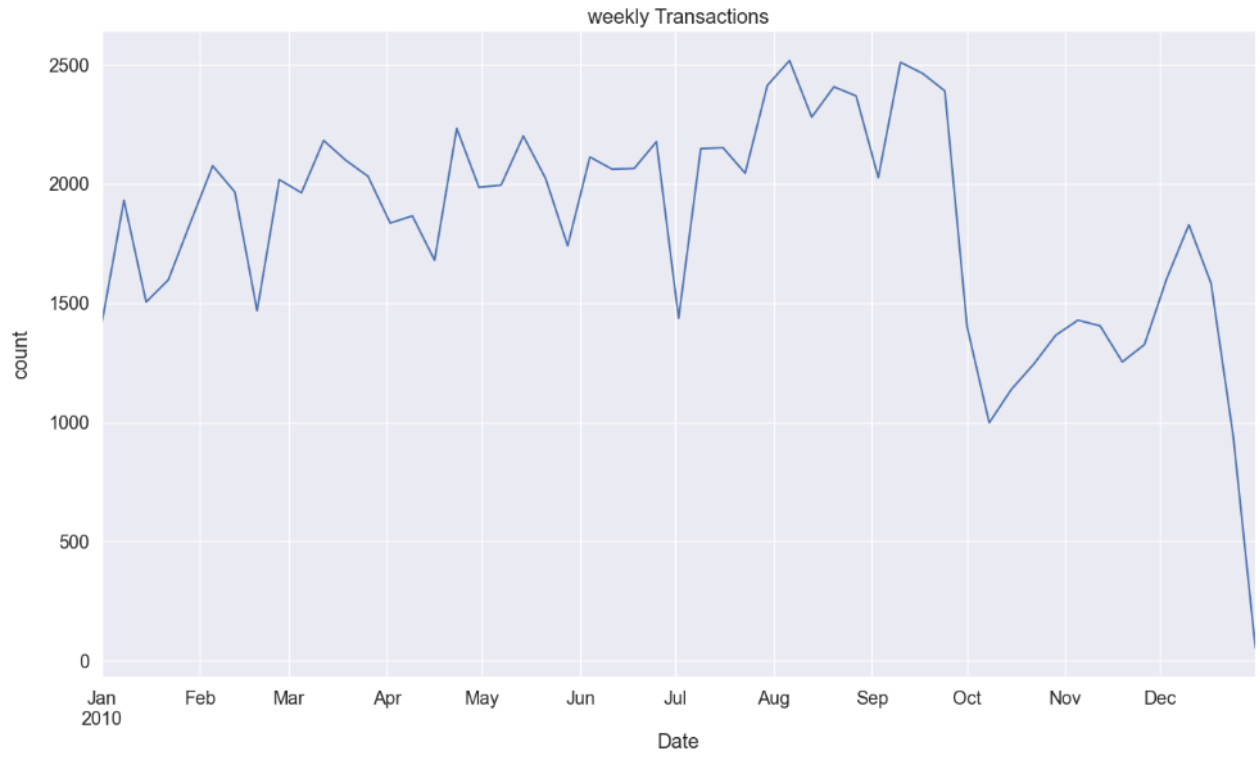
Description: Card Number field. The distribution is the top 30 field values of card number. The most common value of card number is 5142148452, the count of which is nearly 1192.

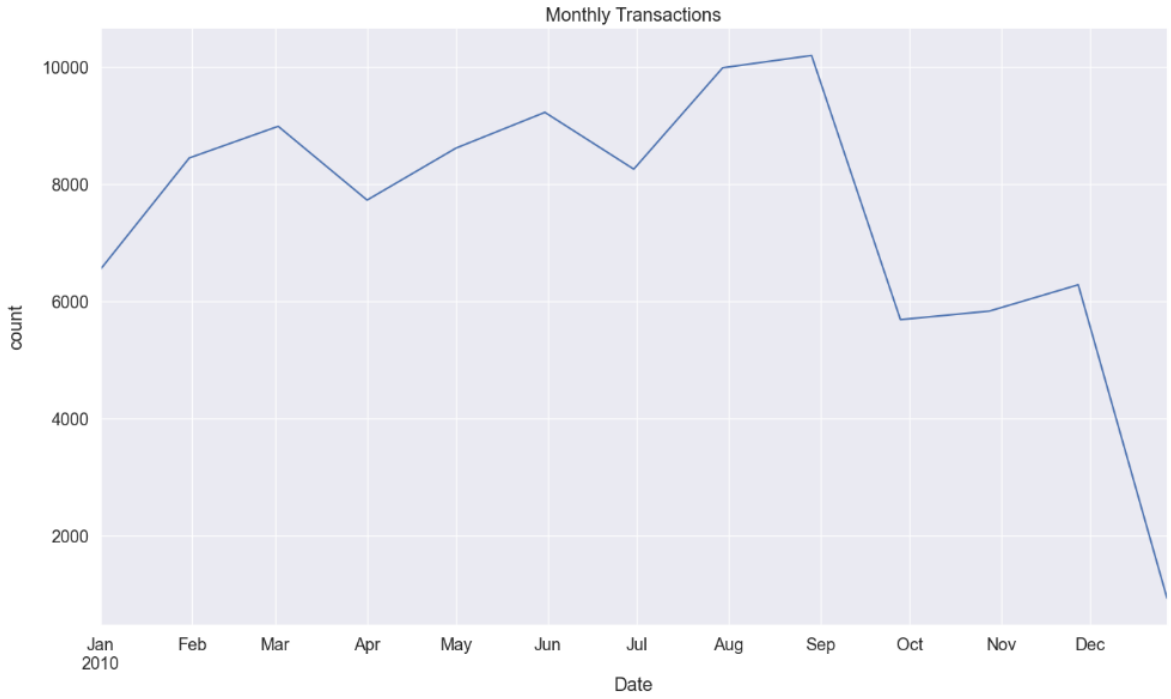


(3). Field Name: Date

Description: Date field. The distribution is the daily transactions, and I also attached the week and month distribution here.

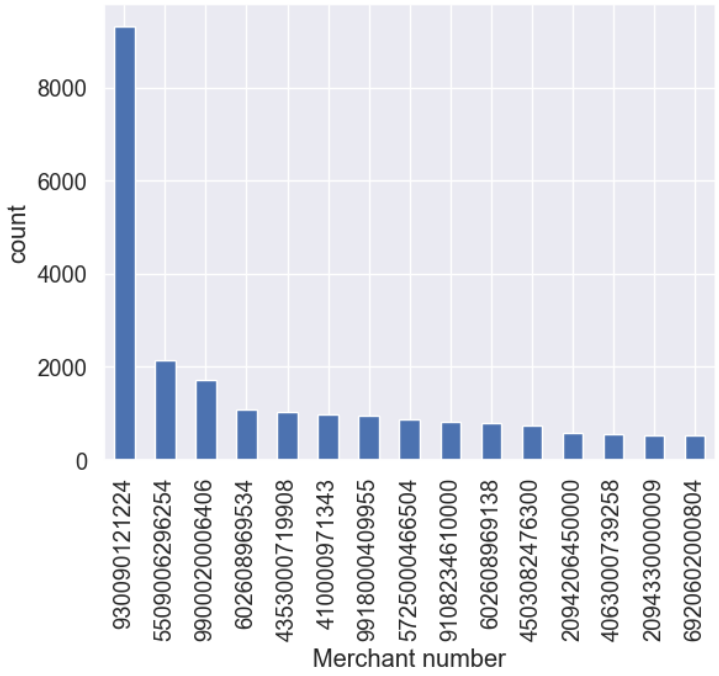

Daily Transactions
700
600
500
400
300
200
100
0
count
Jan
2010
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date


weekly Transactions
2500
2000
1500
1000
500
0
count
Jan
2010
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date



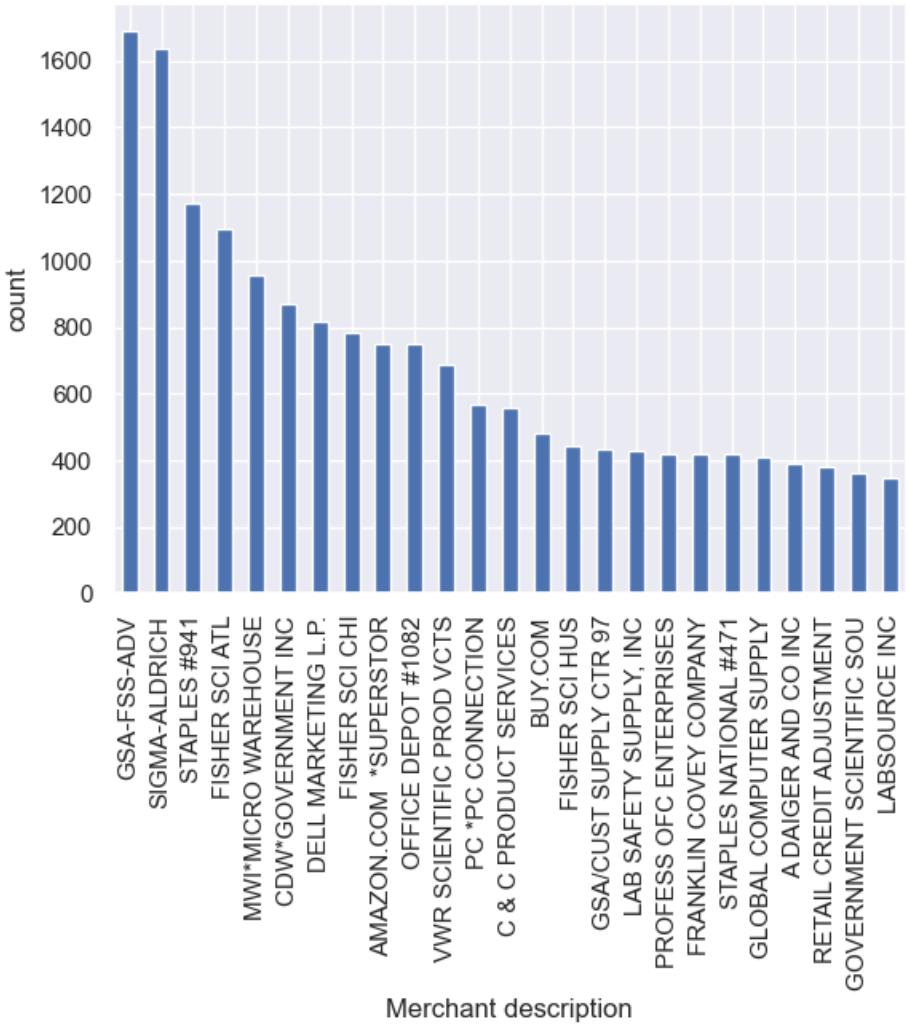
(4). Field Name: Merchnum

Description: Merchant number field. The distribution is the top 15 field values of merchant number. The most common value of merchant number is 930090121224, the count of which is 9310.



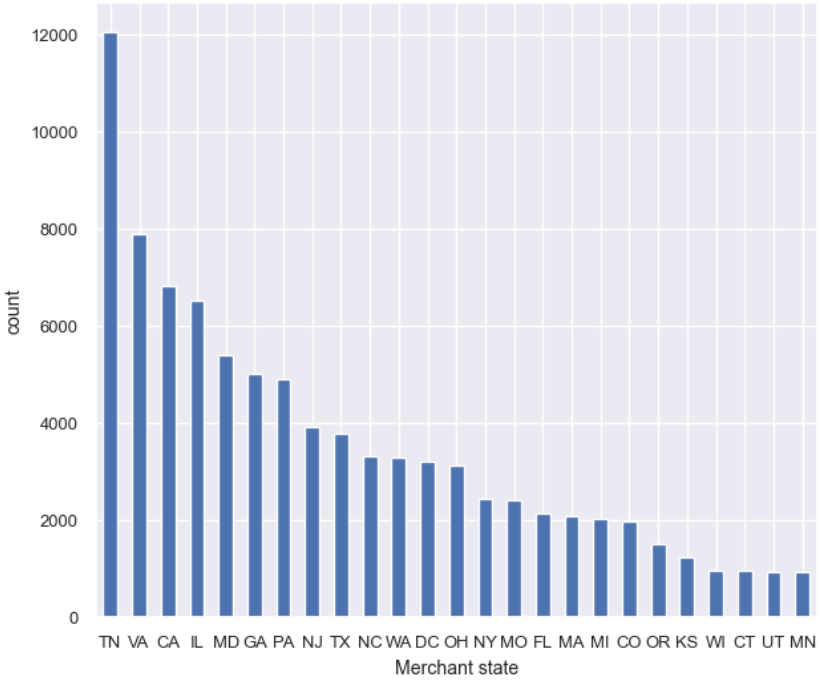
(5). Field Name: Merch description

Description: Merchant description. The distribution is the top 25 field values of merchant descriptions. The most common value of merchant description is GSA-FSS-ADV, the count of which is 1688.



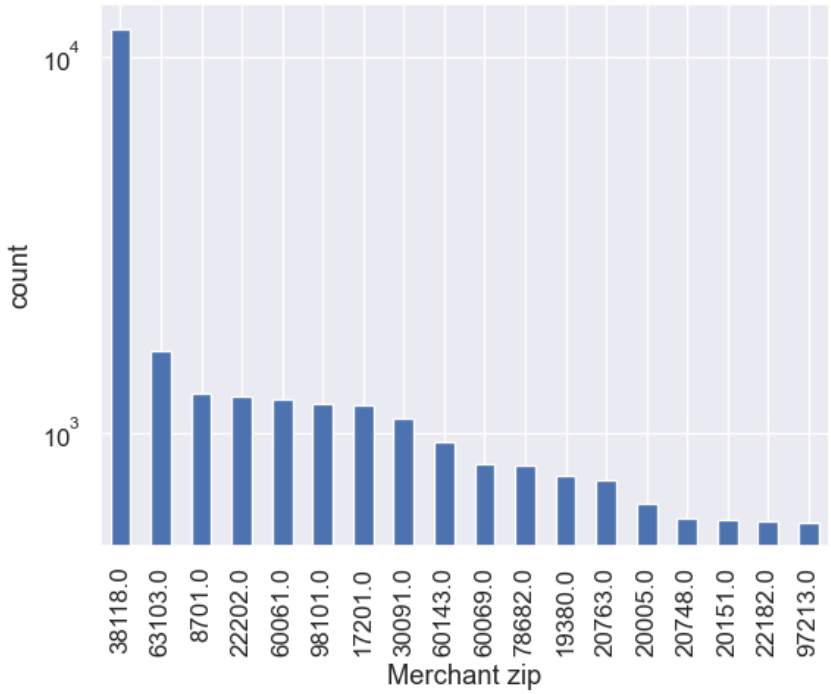
(6). Field Name: Merch state

Description: Merchant state field. The distribution is the top 25 field values of state. The most common value of address is TN, the count of which is 12035.



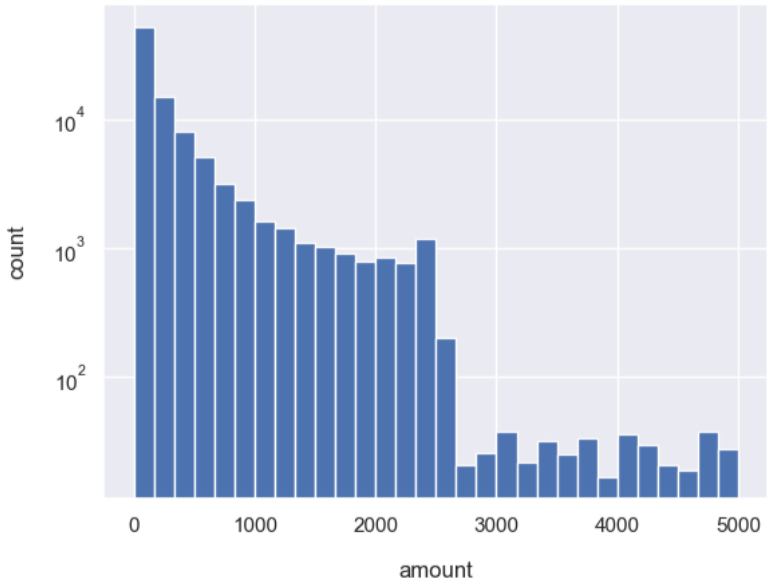
(7). Field Name: Merch zip

Description: Merchant zip code field. The distribution is the top 18 field values of zip code. The most common value of zip code is 38118, the count of which is 11868.



(8). Field Name: Amount

Description: the transaction amount field. This is the distribution plot of the transaction amount. The most common amount is 3.62 and the count of which is 4283. We could notice that most of the transaction amount is below 2500.



(9). Field Name: Transtype

Description: transaction type field. The distribution is the top 20 field values of transaction type. The most common value of transaction type is P, the count of which is 96398.

(10). Field Name: fraud_label

Description: fraud label field. Fraud = 0 (no fraud label), Fraud = 1 (fraud label).

The count of fraud_label = 0 (95694), fraud_label = 1 (1059).