Business Case- Lana Fintech

Table of contents

Data Loading and initial look

Based on the outputs from head and info from *transactions_df*, at first glance there is no evidence that there are missing values for many columns, except for **Errors ?** column. Nevertheless, if after the exploratory analysis we find some missing values, we should impute them before feeding the data to the Machine Learning model.

	User	Card	Year	Month	Day	Time	Amount	Use Chip	Merchant Name	Merchant City	Merchant State	Zip	MCC	Errors?	Is Fraud?
0	0	0	2002	9	1	06:21	$134.09	Swipe Transaction	3527213246127876953	La Verne	CA	91750.0	5300	NaN	No
1	0	0	2002	9	1	06:42	$38.48	Swipe Transaction	-727612092139916043	Monterey Park	CA	91754.0	5411	NaN	No
2	0	0	2002	9	2	06:22	$120.34	Swipe Transaction	-727612092139916043	Monterey Park	CA	91754.0	5411	NaN	No
3	0	0	2002	9	2	17:45	$128.95	Swipe Transaction	3414527459579106770	Monterey Park	CA	91754.0	5651	NaN	No
4	0	0	2002	9	3	06:23	$104.71	Swipe Transaction	5817218446178736267	La Verne	CA	91750.0	5912	NaN	No

Exploratory Data Analysis

For the initial EDA, I would like to identify potential useful features to predict if a transaction is fraudulent or no. This "useful" features selection is based on:

- **Intuition**: aplication of common sense
- **Experience**: previous projects in which I have worked for classificatio problems
- **Facts**: whatever insight we can get from the EDA which can support / deny my intuition and experience.

And by "useful" initial features, they would classify as such if they:

- are numerical features;
- are categorical, but with not too many levels (for practical purposes);
- have potential to be good predictors, by correlation / discrimination properties.

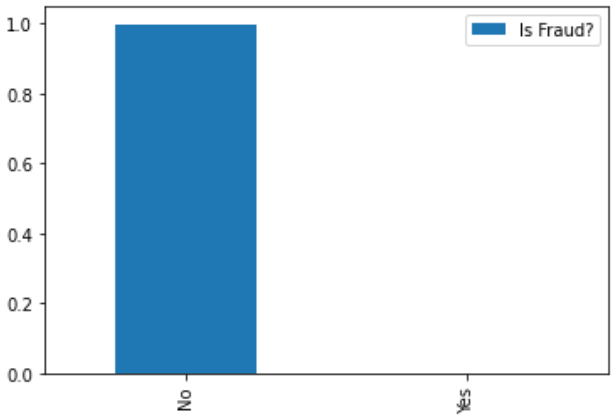
First, I'll explore the categorical features, and then we'll check the numerical features.

Is Fraud? (target)

First let's explore the target variable **Is Fraud?**. From the Distribution we can see there is an extreme imbalance in the classes: 99.9% (No) vs 0.1% (Yes). Usually when the positive class represents less than 10% of the distribution, the problem can be classified as *Anomaly detection*. It also makes sense for this case, as fraudulent transactions are usually anomalies, exceptions which should have some strange behavior. Ideally, we should apply some sampling technique to have a balanced distribution for the target before training the model. If we don't do that, the model will just try to predict that everything is not fraud to optimize the accuracy, which in baseline is 99.9%. Some options we have are:

- *Downsampling*: this would result into a big loss of information, as we'll only be working with 0.2% of the data.
- *Upsampling*: this would be the usual case, but in this case might introduce too many redundant instances.

For an initial balancing technique, we can use a Random Forest with weights for the classes, so that for each tree the samples are taken to balance the distribution. For a more advanced, approach, and based on my experience with Anomaly detection and Ensemble methods, I would like to apply *Blagging* technique.

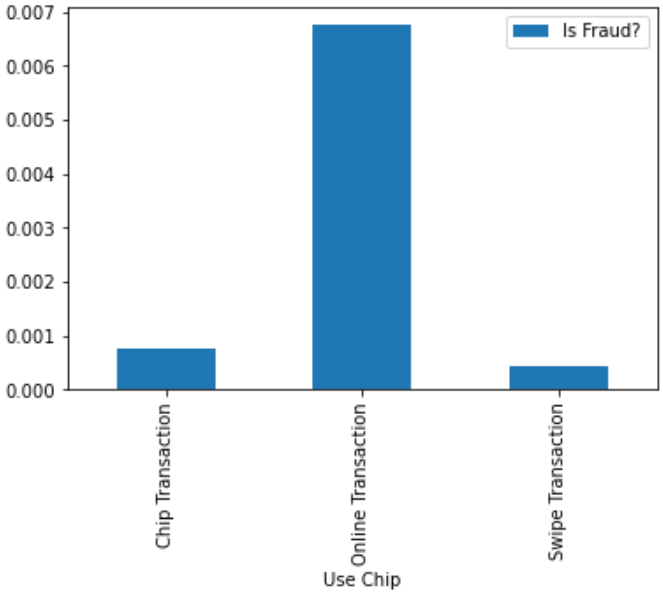


For practical purposes of the analysis and model building, we'll encode the target as a binary feature:

- "Yes" 1
- "No": 0

Use Chip

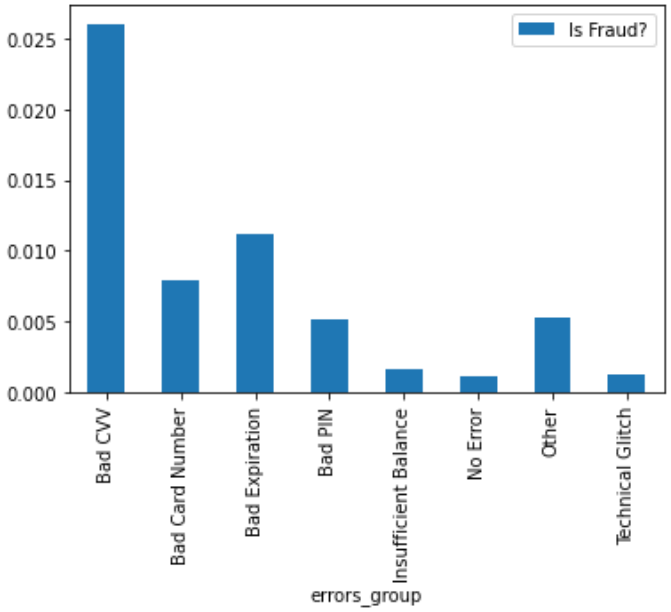
The **Use Chip** feature seems to have potential. It only has three levels: Chip, Online, and Swipe Transaction. In addition, it seems to be a discriminator by itself for the target. Notice how Online Transaction has a higher chance to be a fraudulent transaction, as around 85% of those were Online transactions, while fraudulent Chip Transactions it's relatively low around 10%. Fraudulent transactions with Swipe are around 5%. This was interesting to me, as I thought that the chip makes harder for fraud to occur, but it looks like there is higher chance for fraud (10% for chip vs 5% for swipe).



Errors?

I thought **Errors?** would have less levels, but as it is now with 23 levels seems too high to be included in a base model.

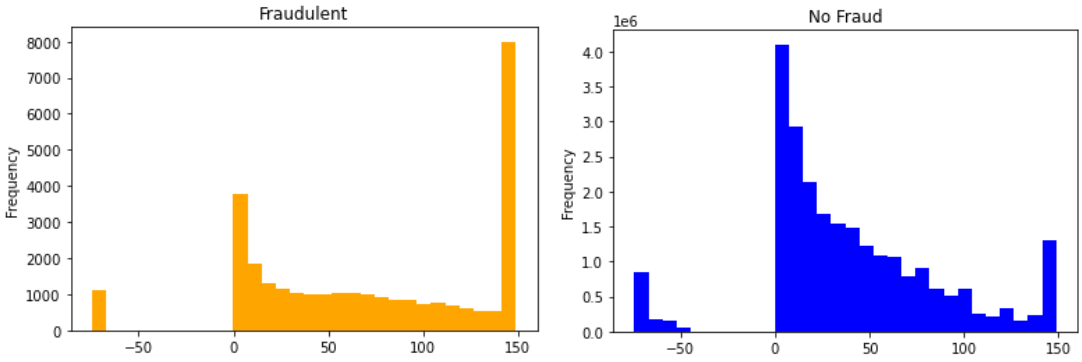
Nevertheless, if we see the cumulative percentage, notice how the top 6 errors cover around 99% of the cases. Therefore, we can consider these as the most important levels, and tag the rest as *Other* error. This would give us 7 levels instead of 23. This information will be encoded in a new feature "errors_group", which might be more useful than the original one.



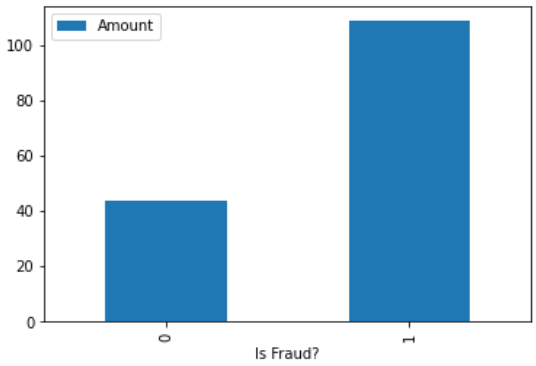
It's interesting to see how *Bad CVV* has the higher chance of being a fraudulent transaction (around 2.5%). Bad Expiration also has a smal chance of around 1%. And Bad Card Number and Bad PIN are the next two with almost 0.5% of probability. What is interesting to observe, is that there are some fraudulent cases in which at least one of the properties of the credit card is unknown.

Amount

For the **Amount** feature, first we'll do an appropriate parsing to float. In addition, we'll apply a capping by IQR to remove outliers (for plotting purposes).

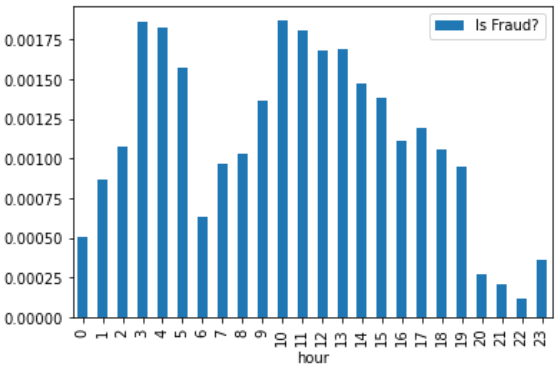


This feature seems to have some discrimination potential, as transaction which are fraudulent seem to be biased toward higher values than normal transactions. Maybe an indicator that the ones who commit fraud try to get as much as possible? Also, we can see that the average amount for fraudulent transactions is much higher than for normal ones: 109 vs 44 USD. By performing a Welch's t-test for the difference in means, we can see that there is enough evidence to reject the null hypothesis of equal population means (p-value < 0.05)



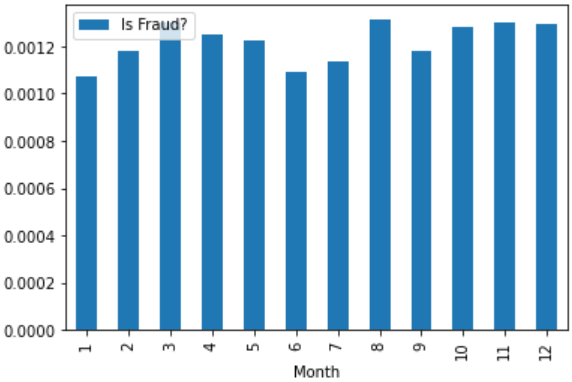
Time (hour)

For the **time** feature, first we'll do an appropriate parsing to float. We'll just extract the hour part from it and store it into a new feature **hour**. Notice how the transactions which are fraudulent have a varying patten across the hour of the day. With two peaks in the early morning (3am to 5am) and around the midday. After midday it begins to decrease, until it drops at 8pm until the midnight. This varying distribution allow us to think that the **hour** feature can be useful to identify a fraudulent transaction.



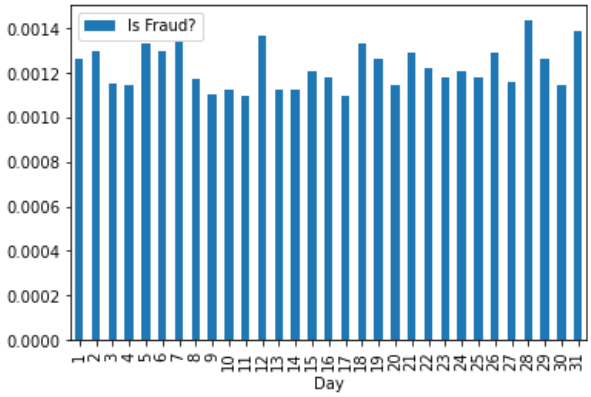
Month

Like **hour**, I thought that **Month** feature could also be an indicator of fraud, as some months have some special events like Easter, Christmas, Thanksgiving... which might make more attractive to have fraud. While there seems to be a pattern across the months, there is just a slight variation. With some interesting peaks at March, August, and the last three months.



Day

Day feature seems to show a similar pattern as with Month, just a slight variation but with some interesting peaks at some days, like 12 and close to 20. We see how on the first few days and the last days of the month there seem to be an increase in fraudulent transactions. Maybe, as these days are close to paydays, it might make more attractive to commit fraud.



Merchant-based features

Most **merchant-based** columns have too many unique values as to be included in a base model:

- **Merchant Name** has around 100K unique values
- **Merchant City** has around 13K unique values
- **Merchant State** has 223 unique values
- **Zip** has around 27K unique values
- **MCC** has 109 unique values

MCC is a feature which would be interesting to add, as it denotes the type of business providing a service or selling merchandise. In addition, MCCs are used by card issuers to categorize, track, or restrict certain types of purchases. Therefore, some MCCs might appear more often on fraudulent transactions. (CreditCards.com, 2021)

To include it as a feature, we'll consider the top 40 as relevant MCC codes, which represent around 96% of the cases. And classify the rest as others. We'll do something similar for the **Merchant State** feature, as also the top 40 represent around 96% of the cases.

For the states, it is of interest to identify the ones in which fraud is an issue, either because there is a high percent of fraudulent cases on that state, or because there is a high number of transactions in that state. To take both factors into consideration, we'll calculate a score as:

$$score = percent(fraudulentCases) * percent(totalCases)$$

Then we normalize the score column, and sort by it to identify the states with most fraudulent issues. We present the top 20 states, which make up around 88% of the total score. We can see three kinds of states:

- States with high percent of fraudulent cases: like Italy (54%), Algeria (96%), Turkey (54%), Nigeria (61%) ...
- States with high percent of total cases: like CA (12%), FL (7%), TX (8%), NY (7%) ...
- States with low percent of fraudulent cases but high percent of total cases (high score): like OH (8% score), Mexico (2%), IL (1%) ...

	Is Fraud?		percent	score
	mean	count		
Merchant State				
Italy	0.536312	8730	0.000403	0.410414
OH	0.000980	895970	0.041354	0.076964
CA	0.000290	2591830	0.119626	0.065831
Algeria	0.961774	654	0.000030	0.055137
Haiti	0.840807	446	0.000021	0.032872
FL	0.000215	1458699	0.067326	0.027525
TX	0.000170	1793298	0.082770	0.026736
Mexico	0.005769	47152	0.002176	0.023843
Turkey	0.544492	472	0.000022	0.022528
NY	0.000166	1446864	0.066780	0.021038
NJ	0.000271	630317	0.029092	0.014989
MI	0.000259	618407	0.028543	0.014025
NC	0.000185	779234	0.035966	0.012623
PA	0.000172	839647	0.038754	0.012623
Nigeria	0.613734	233	0.000011	0.012535
IL	0.000134	850074	0.039235	0.009993
TN	0.000216	504116	0.023268	0.009555
WA	0.000199	537762	0.024820	0.009379
VA	0.000247	425216	0.019626	0.009204
IN	0.000171	613432	0.028313	0.009204

For the MCCs, like we did with the states, it is of interest to identify the ones in which fraud is an issue.

We present the top 20 MCCs, which make up around 95% of the total score. We can see three kind of MCCs:

- MCCs with high percent of fraudulent cases: like Computers, Computer Peripheral Equipment, Soft... (10%); Electronic Sales (7%), Precious Stones and Metals; Furniture, Home Furnishings, and Equipment Sto... (6%); Watches and Jewelry (6%) ...
- MCCs with high percent of total cases: like Grocery Stores, Supermarkets (12%), Service Stations (11%), Eating places and Restaurants (7%), Wholesale Clubs (7%) ...
- MCCs with low percent of fraudulent cases but high percent of total cases (high score): like Discount Stores (2% score), Drug Discount Stores (1% score), Digital Goods: Media, Books, Movies, Music (1%) ...

To do a merge between our dataframe and the descriptions for the MCCs, we used this dataset from (Knaddison, 2021).

	(Is Fraud?, mean)	(Is Fraud?, count)	(mcc,)	(percent,)	(score,)	edited_description
0	0.005446	885720	5311	0.036319	0.162113	Department Stores
1	0.001960	1123037	5300	0.046051	0.073966	Wholesale Clubs
2	0.004739	454107	5310	0.018621	0.072319	Discount Stores
3	0.001423	1129061	4829	0.046298	0.054004	Money Orders – Wire Transfer
4	0.000751	1407636	5912	0.057721	0.035521	Drug Stores and Pharmacies
5	0.000330	2860738	5411	0.117306	0.031690	Grocery Stores, Supermarkets
6	0.007627	115247	5815	0.004726	0.029539	Digital Goods: Media, Books, Movies, Music
7	0.006175	137489	5651	0.005638	0.028531	Family Clothing Stores
8	0.066942	12593	5732	0.000516	0.028329	Electronic Sales
9	0.004474	160277	5719	0.006572	0.024095	Miscellaneous Home Furnishing Specialty Stores
10	0.060038	9444	5094	0.000387	0.019054	Precious Stones and Metals, Watches and Jewelry
11	0.000569	981523	4121	0.040248	0.018752	Taxicabs and Limousines
12	0.001469	323456	5211	0.013264	0.015963	Lumber and Building Materials Stores
13	0.093756	5013	5045	0.000206	0.015795	Computers, Computer Peripheral Equipment, Soft...
14	0.000518	900255	5814	0.036916	0.015660	Fast Food Restaurants
15	0.058470	7149	5712	0.000293	0.014047	Furniture, Home Furnishings, and Equipment Sto...
16	0.006933	58562	4722	0.002401	0.013644	Travel Agencies and Tour Operations
17	0.000984	407465	4814	0.016708	0.013476	Fax services, Telecommunication Services
18	0.000134	2638982	5541	0.108213	0.011896	Service Stations (with or without ancillary s...
19	0.004587	71501	7922	0.002932	0.011023	Theatrical Producers (Except Motion Pictures),...
20	0.000182	1797920	5812	0.073725	0.010989	Eating places and Restaurants

Set of features from transactions_df

Now we are ready to select the potential features from the *transactions_df*.

First, we'll get the dummy variables for the categorical features. There are no missing values, as we handled them on the feature engineering process.

	Use Chip	errors_group	Amount	hour	Month	Day	merchant_state_group	mcc_group
0	Swipe Transaction	No Error	134.09	6	9	1	CA	5300
1	Swipe Transaction	No Error	38.48	6	9	1	CA	5411
2	Swipe Transaction	No Error	120.34	6	9	2	CA	5411
3	Swipe Transaction	No Error	128.95	17	9	2	CA	5651
4	Swipe Transaction	No Error	104.71	6	9	3	CA	5912
...	...	...	...	...	...	...	...	...
24386895	Chip Transaction	No Error	-54.00	22	2	27	NH	5541
24386896	Chip Transaction	No Error	54.00	22	2	27	NH	5541
24386897	Chip Transaction	No Error	59.15	7	2	28	NH	4121
24386898	Chip Transaction	No Error	43.12	20	2	28	NH	4121
24386899	Chip Transaction	No Error	45.13	23	2	28	NH	5814

24386900 rows × 8 columns

Model Building

For the model, I decided to fit a Random Forest. This will allow us to get:

- Features importance: to determine the importance of each feature to predict the fraud.
- Out-of-bag Score: to get an idea of the actual performance of the model (to be compared to test metrics).

Ideally, the hyper-parameters of the model should be optimized with some technique, like K-fold cross-validation. But we'll choose some robuts & generic parameters for simplicity purposes. Some observations:

- using 500 estimators as a robuts number of decision trees.
- using max_depth, min_samples_split, min_samples_leaf, and max_features parameters to have weak learners.
- using max_samples = 30000 (for practical purposes)
- using class_weight='balanced' so that each sample is balanced on the target

The classifier had a **training time** of 7.6 min and **prediction time** of 1.7 min.

Evaluation Metrics

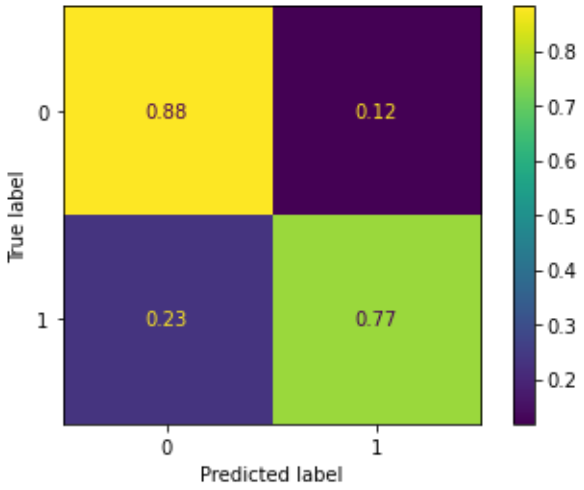
From the classification report, we can see how most of the metrics seem to be around 88% in average. The Out-of-bag score we get from the random forest classifier is also very close to 88%. This is a good indicator that the model did not overfit or underfit. In addition, the weighted avg for precision, recall, and thus F1-score are also close to 88%.

	precision	recall	f1-score	support
0	1.00	0.88	0.94	9742857
1	0.01	0.77	0.02	11903
accuracy			0.88	9754760
macro avg	0.50	0.82	0.48	9754760
weighted avg	1.00	0.88	0.94	9754760

Confusion Matrix

Something we can notice on the confusion Matrix, is that the model doesn't seem to have a specific bias to any of the classes. Nevertheless, the recall on the positive label could be increased to be also above 80%, by fine-tuning the model. This will increase number of True-Positive cases and reduce number of False-Positive cases (currently 12% of normal transactions are classified as fraudulent).

A good property we can observe, is that given the high recall (around 77% on positive class) it means that the model is able to capture most of the fraudulent transactions.



Features Importance

We have 97 features in total. From the random_forest, we can get the importances of the features based on information gain. As can be seen, the top 20 most important features are:

- merchant_state_group_Other
- Use Chip_Online Transaction
- Use Chip_Swipe Transaction
- Amount
- mcc_group_Other
- hour
- mcc_group_5311
- mcc_group_5541
- mcc_group_5499
- Day
- mcc_group_5411
- merchant_state_group_TX
- Month
- mcc_group_4784
- merchant_state_group_CA
- Use Chip_Chip Transaction
- mcc_group_5310
- merchant_state_group_FL
- merchant_state_group_NY
- mcc_group_5812

These 20 features by themselves make up around 92% of the cummulative importance. In addition, we could use these importances to remove unimportant features. Around 52 features have importance which is technically zero (less than 0.1%). By removing these, and refitting the model, we might compare how the performance is impacted.

It is interesting to see the Merchant State which seem important. *Other* being the most important feature might be an indicator that when transactions are not in the most common states, that it is a fraudulent one (anomaly):

- Other
- TX
- CA
- FL
- NY

Another interesting aspect from features importance, is to see the descriptions for this important mccs:

- 5311: Department Stores
- 5541: Service Stations
- 5499: Misc. Food Stores – Convenience Stores and Specialty Markets
- 5411: Grocery Stores, Supermarkets
- 4784: Toll and Bridge Fees
- 5310: Discount Stores
- 5812: Eating places and Restaurants

Most of these were already identified from our previous analysis. However, these two were not part of the top 20, but ended up being important:

- 5499: Misc. Food Stores – Convenience Stores and Specialty Markets
- 4784: Toll and Bridge Fees

References

CreditCards.com. (9 de August de 2021). *Credit Card Glossary: Terms and Definitions*. Obtenido de creditcards.com: https://www.creditcards.com/credit-card-news/glossary/term-merchant-category-code-mcc/

Knaddison, G. (9 de August de 2021). *mcc_codes.csv.* Obtenido de mcc-codes: https://github.com/greggles/mcc-codes/blob/main/mcc_codes.csv