Fraud Detection, Vesta Corporation

Fall 2023, AI Studio Project Write-Up[1]

Table of Contents

Business Focus

This project focuses on benchmarking machine learning models against a large-scale dataset derived from Vesta Corporation's real-world e-commerce transactions. The goal was to build a machine learning model that predicts the probability that an online transaction was fraudulent, as denoted by the binary target `isFraud`, and results in higher accuracy fraud detection to improve customer experience. Successful solutions will significantly improve fraudulent transaction alerts, benefiting millions globally, reduce fraud losses for businesses, and enhance the customer experience by preventing false positives.

Data Preparation and Validation

DATASET DESCRIPTION

We were given two data files by Vesta Corporation, `identity` and `transaction`, which are joined by `TransactionID`. Not all transactions have corresponding identity information. These datafiles included a mix of categorical transaction features, including `ProductCD`, `card1` - `card6`, `addr1`, `addr2`, `P_emaildomain`, `R_emaildomain` and `M1` - `M9`, as well as categorical identity features including `DeviceType`, `DeviceInfo` and `id_12` - `id_38`.

To pre-process the data, we joined both files. We then dropped ID columns, recoded categorical variables (like card type), and handled null values.

---

[1] *This sample project write-up is adapted from the Kaggle Solution Write-Up Documentation and Chris Deotte and Konstantin Yakovle's 1st Place Solution Write-Up for the IEEE-CIS Fraud Detection Competition. For this exercise, this case study includes fictional workplace settings and teams.*

EDA (EXPLORATORY DATA ANALYSIS)

Exploratory data analysis was daunting for our project because there were 430 total columns of data and their meanings were obscure.

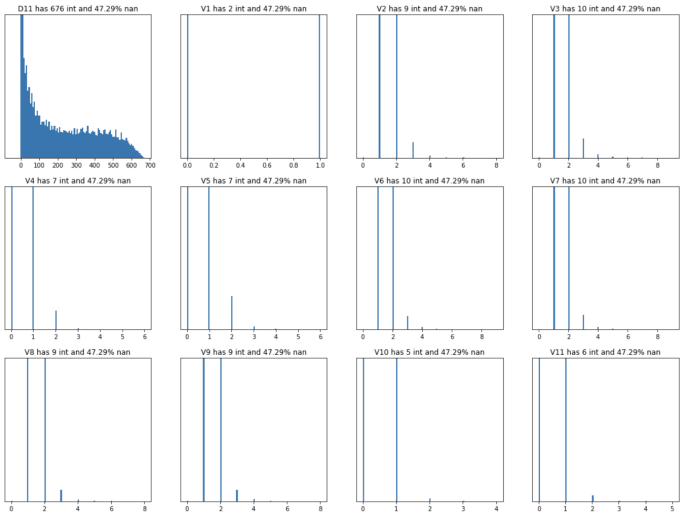
For the first 150 columns (the original 53 columns and first 95 V columns) we applied an idea used by Alijs. (Thanks, Alijs, for sharing your thoughts!) See our specific EDA in GitHub, here.[2]

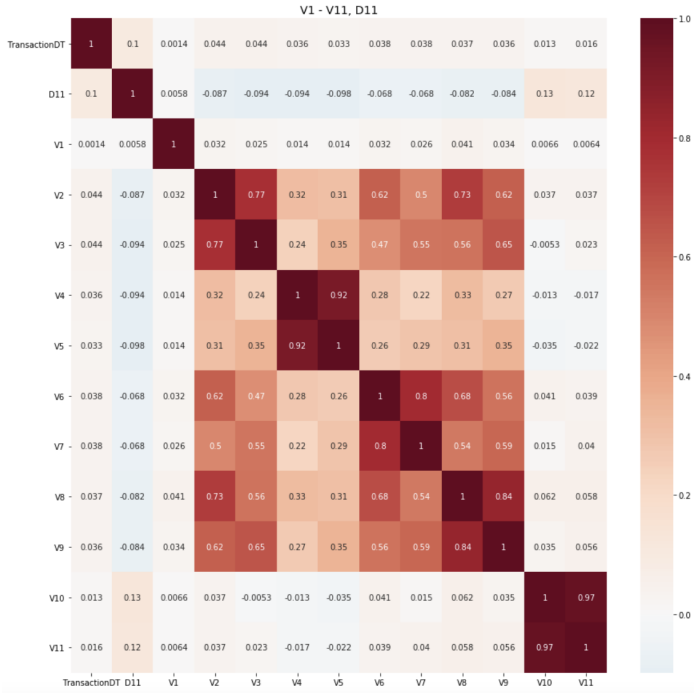
For the remaining 300 columns, we used a separate V and ID EDA analysis. We reduced the number of V columns with 3 tricks. First, groups of V columns were found that shared similar NAN structure, and then we used 1 of 3 methods:

- We applied PCA on each group individually
- We selected a maximum sized subset of uncorrelated columns from each group
- We replaced the entire group with all columns averaged

The V columns appear to be redundant and correlated. Therefore for each block of V columns with similar NAN structure, we could find subsets within the block that are correlated ($r > 0.75$). Then we can replace the entire block with one column from each subset.

For example in block `V1-V11`, we see that the subsets `[[1],[2,3],[4,5],[6,7],[8,9],[10,11]]` exist and we can choose `[1, 3, 4, 6, 8, 11]` to represent the `V1-V11` block without losing that much information. Alternatively you could apply PCA on each block, but this subset reduce method performed better.





[2] We recommend including links to your GitHub repo for any project resources that might impress a recruiter or hiring manager!

These reduced groups were further evaluated using feature selection techniques below. For example, the block `V322-V339` failed "time consistency" and was removed from our models.

FEATURE SELECTION

Because we had many columns and preferred to keep our models efficient, feature selection was particularly important. For example, my XGB model had 250 features and would train 6 folds in 10 minutes.

We used every trick we knew to select our features:

- Forward feature selection (using single or groups of features)
- Recursive feature elimination (using single or groups of features)
- Permutation importance
- Adversarial validation
- Correlation analysis
- Time consistency
- Client consistency
- Train/test distribution analysis

"Time consistency" is an interesting trick to train a single model using a single feature (or small group of features) on the first month of the train dataset, and predicts `isFraud` for the last month of train dataset. This evaluates whether a feature by itself is consistent over time. We found that 95% were consistent, but 5% of columns hurt our models. They had training AUC around 0.60 and validation AUC 0.40. In other words, some features found patterns in the present that did not exist in the future. Of course the possibility of interactions complicates things, but we double checked every test with other tests.

Approach

SELECTED MODELS

We selected 3 main candidate models to explore throughout our AI Studio project:

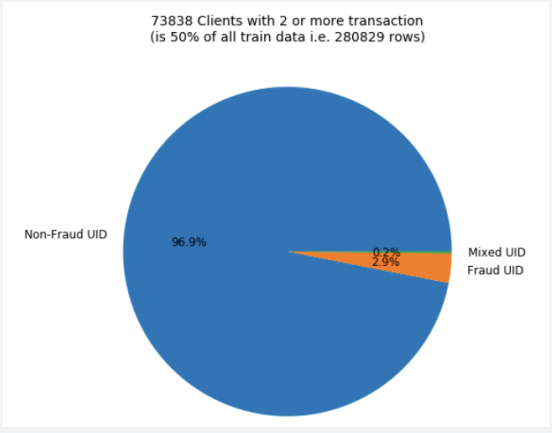
- **Catboost** – we selected this model because it can efficiently handle categorical features without the need for extensive preprocessing, and because it incorporates a robust regularization technique that helps prevent overfitting
- **LGBM** – Light Gradient Boosting Machine (LGBM) can handle large datasets and is built on the gradient boosting framework. The LGBM model can handle categorical features efficiently, without the need for one-hot encoding to simplify the preprocessing steps.
- **XGB** – XGBoost is more suitable for smaller to medium-sized datasets and requires one-hot encoding and more careful tuning of hyperparameters to achieve optimal performance.

MAGIC FEATURE

In our project we are not predicting fraudulent transactions. Once a client (credit card) has fraud, their entire account is converted to `isFraud`=1. Therefore we are predicting fraudulent clients (credit cards).

More specifically, we defined reported chargeback on the card as fraud transaction (`isFraud`=1) and transactions posterior to it with either user account, email address or billing address directly linked to these attributes as fraud too. If none of the above is reported and found beyond 120 days, then we defined it as a legitimate transaction (`isFraud`=0).

You might think that after 120 days, a card becomes `isFraud`=0 . We rarely saw that in the training data. (Perhaps fraudulent credit cards get terminated instead of re-used). The pie chart below shows this. The training dataset has 73838 clients (credit cards) with 2 or more transactions. Of those, 71575 (96.9%) are always `isFraud`=0 and 2134 (2.9%) are always `isFraud`=1. Only 129 (0.2%) have a mixture of isFraud=0 and `isFraud`=1.



FINDING UIDs - UNIQUE IDENTIFICATION

Because our dataset had 430 columns, it was unclear how we could know which columns might help identify clients. We used the approach to find UIDs to identify individual clients (credit cards) because: (1) it allows models to find fraud better, (2) it allows us to validate how models perform on seen versus unseen clients, and (3) it allows us to post process predictions. We found UIDs in two different ways (specific details here):

- Wrote a script that finds UIDs
- Train our models to find UIDs

As a result, we found almost all 600,000 unique cards and respective clients. While our original approach without UIDs achieved local validation AUC = 0.9245, our approach that finds and uses UIDs achieved local validation AUC = 0.9377, using even less human assistance and beating all other methods. The purpose of producing UIDs by a script was for EDA, special validation tests, and post-process. We didn't add the script's UIDs to our models. Machine learning did better finding them on its own.

FRAUDULENT CLIENTS

Below is one example of 2,134 that we could show you of clients with `isFraud`=1. This is `client=2988694` . The key to identifying clients are the three columns `card1`, `addr1`, and `D1`. The column `D1` is "days since client (credit card) began". Therefore if we create D1n = TransactionDay minus D1, we get the day the card began (where `TransactionDay=TransactionDT/(24*60*60)`). In the example below, this credit card began on `day=-81` (which is approximately Sept 10, 2017 because the dataset is believed to begin on December 1, 2017). We also see that `D3n=TransactionDay minus D3` is a pointer to the day of each client's last transaction. (There are more columns that help us identify clients too).

	TransactionID	isFraud	TransactionAmt	card1	addr1	D1n	day	D3n	dist1	P_emaildomain	UID
1694	2988694	1	240.0	15775	251.0	-81.0	1.0	0.0	NaN	yahoo.com	2988694.0
10046	2997046	1	260.0	15775	251.0	-81.0	3.0	1.0	NaN	yahoo.com	2988694.0
34029	3021029	1	250.0	15775	251.0	-81.0	9.0	3.0	NaN	yahoo.com	2988694.0
36812	3023812	1	315.0	15775	251.0	-81.0	10.0	9.0	NaN	yahoo.com	2988694.0
40459	3027459	1	390.0	15775	251.0	-81.0	11.0	10.0	NaN	yahoo.com	2988694.0
43926	3030926	1	475.0	15775	251.0	-81.0	12.0	11.0	NaN	yahoo.com	2988694.0
43941	3030941	1	445.0	15775	251.0	-81.0	12.0	12.0	NaN	yahoo.com	2988694.0
44717	3031717	1	445.0	15775	251.0	-81.0	12.0	12.0	NaN	yahoo.com	2988694.0
44727	3031727	1	445.0	15775	251.0	-81.0	12.0	12.0	12.0	NaN	2988694.0
58485	3045485	1	295.0	15775	251.0	-81.0	15.0	12.0	NaN	yahoo.com	2988694.0

PREVENTING OVERFITTING

To prevent overfitting the train and overfitting the public test dataset, we did not directly use client UID or the dozens of columns in the dataset that help identify clients, such as D, V, and ID columns. (You know which columns identify clients from reviewing LGBM feature importance during train/test adversarial validation).

We could not add UID as a new column because 68.2% of clients in the private test dataset are not in the training dataset. Instead we created aggregated group features. For example, we took all the C, M columns and did this `new_features = df.groupby('uid')[CM_columns].agg(['mean'])`. Then we deleted the column `uid`. As a result, our model had the ability to classify clients that it had never seen before.

VALIDATION STRATEGY

We never trusted a single validation strategy so we used lots of validation strategies. Train on first 4 months of train, skip a month, predict last month. We also did train 2, skip 2, predict 2. We did train 1 skip 4 predict 1. We reviewed LB scores (which is just train 6, skip 1, predict 1 and no less valid than other holdouts). We did a CV GroupKFold using month as the group. We also analyzed models by how well they classified known versus unknown clients using our script's UIDs.

For example when training on the first 5 months and predicting the last month, we found that our

- XGB model did best predicting known UIDs with AUC = 0.99723
- LGBM model did best predicting unknown UIDs with AUC = 0.92117
- CAT model did best predicting questionable UIDs with AUC = 0.98834

Questionable UIDs are transactions that our script could not confidently link to other transactions. When we ensembled and/or stacked our models we found that the resultant model excelled in all three categories. It could predict known, unknown, and questionable UIDs forward in time with great accuracy !!

Key Findings And Insights

KEY RESULTS

Our final model was a combination of 3 high scoring single models. CatBoost (Public/Private LB of 0.9639/0.9408), LGBM (0.9617/0.9384), and XGB (0.9602/0.9324). These models were diversified because Katerina built the CAT and LGB, while I built the XGB and NN. We engineered features independently. (In the end we didn't use the NN, which had LB 0.9432.) The XGB notebook is posted here.

One final submission was a stack where LGBM was trained on top of the predictions of CAT and XGB and the other final submission was an ensemble with equal weights. Both submissions were post-processed by taking all predictions from a single client (credit card) and replacing them with that client's average prediction. This PP increased LB by 0.001.

INSIGHTS

1. **Time is not most important**. Our team initially presumed that this would be a timeseries project, however this project isn't actually about time: the nature of fraud changes does not radically change over time (and so adversarial validation has AUC=1), but rather *because the clients in the dataset change radically over time*. Once we realized this, we knew that the challenge for this project was building a model that can predict unseen clients (not unseen time).
2. **More is better (when it comes to validation strategies).** We did not trust a single validation strategy, and instead used lots of validation strategies. For example, we did train on the first 4 months of the train, skipped a month, and predicted the last month. We also did train 2, skip 2, predict 2. We did train 1, skip 4, predict 1. We reviewed LB scores (which is just train 6, skip 1, predict 1, which was no less valid than other holdouts). We did a CV GroupKFold using month as the group. We also analyzed models by how well they classified known versus unknown clients using our script's UIDs. For example, when training on the first 5 months and predicting the last month, we found that our:XGB model did best predicting known UIDs with AUC = 0.99723
    - LGBM model did best predicting unknown UIDs with AUC = 0.92117

- CAT model did best predicting questionable UIDs with AUC = 0.98834

Questionable UIDs are transactions that our script could not confidently link to other transactions. When we ensembled and/or stacked our models, we found that the resultant model excelled in all three categories. It could predict known, unknown, and questionable UIDs forward in time with great accuracy!

3. **Working in teams is powerful, but not always easy.** This was one of my first experiences working on a team to tackle a technical project of such large scope. The most rewarding lesson has been the power of collective problem-solving. Tackling complex technical issues led to innovative solutions that no individual could have conceived alone. This collective problem-solving approach not only enhanced the quality of the final product but also fostered a sense of camaraderie among team members. That said, our team benefited most from setting, clear and open communication and norms. Regular team meetings, status updates, and openly sharing when we got stuck helped to ensure that everyone was on the same page. I am eager to bring these lessons learned to my internship this summer!.

Acknowledgements

Wow! Our project launch date is here! I've learned so much in these last 10 weeks, and am grateful to everyone at Vesta Corporation for all that you've shared. To my Challenge Advisor, Chiara, and teammates Alex, Alijs, Brooke, and Katerina, thanks for your support, and opportunity to learn alongside you! I can't wait to see what the Spring AI Studio brings next!

Additionally, thank you to Break Through Tech, the Cornell Tech AI Program team, and especially Erika and Abby for an amazing program experience so far! We can't wait to spread the word to all of our friends to participate next year!