E-Commerce Transaction Fraud Detection

Saiakash Konidena
IIIT Bangalore
Bangalore, India
saiakash.konidena@iiitb.org

Ritik Gupta
IIIT Bangalore
Bangalore, India
ritik.gupta@iiitb.org

Tanmay Arora
IIIT Bangalore
Bangalore, India
tanmay.arora@iiitb.org

***Abstract*—We propose our approach where we have predicted the probability of an E-Commerce Transaction being fraud. We describe our work right from Preprocessing to Hyperparameter Tuning. We also describe some insights we gained during Exploratory Data Analysis.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

This document is a model and instructions for LaTeX. Please observe the conference page limits.

II. Problem Statement

The Problem to be solved is a simple binary classification. We have to predict the probability of an E-commerce Transaction being Fraud using data provided for each Transaction.

III. Previous Approaches

Fraud Detection in Financial Transactions has been a huge problem faced by Banks and Financial Institutions for decades. Early attempts at fraud detection were done by these individual institutions based on Data owned by them. Soon they realised a global system is necessary, with periodic sharing of such data which wasn't possible due to legal reasons. This resulted in companies which provided fraud protection services based on huge chunks of data collected with a lot of missing values, like in our case. Also, along with null values, we have unknown features from which one has to find patterns. Our current problem is very similar to this.

IV. Dataset

The dataset we use is one of Kaggle's Public Datasets (Link) The data is provided by one of the leading financial agencies, Vesta Corporation.

On basic analysis we find that the distribution of our Target Variable ie. isFraud is highly skewed. We have 97.5% Negatives and only 2.5% Positives. The Train Data is divided into 2 files, Transaction Data (tt) & Identity Data(ti) . Each Transaction is identified by a unique TransactionID. Not all Transactions have corresponding Identity Data but all Identity Features have a corresponding Transaction Data. This results in a lot of Null Values. We try a little unconventional approach (Two Model Approach) to deal with this huge amount of Null Values. Only 24% Transaction Data had corresponding Identity Data

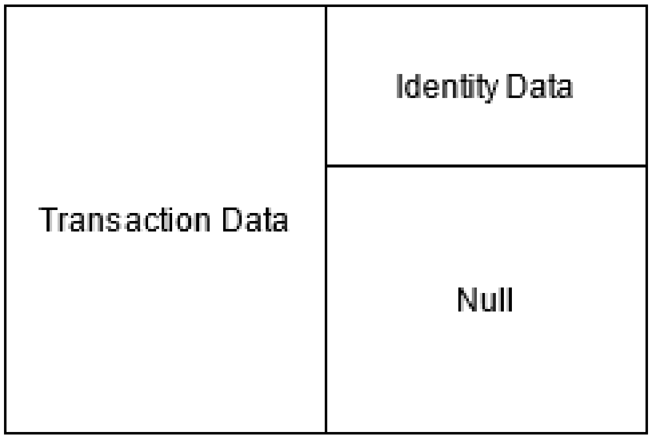


Fig. 1. Schema of our Dataset

V. Evaluation Metric

The issue with Highly Skewed Datasets like ours is that it is very easy to predict all observations to be of the majority class and we can easily get a good accuracy, 97.5% accuracy in our case, which sounds really amazing but all we did was predict all Transactions as not being Fraud and this model is not really useful for real life applications. In this case, AUC ROC is a better metric to measure the generalisability of our model. It has a range between [0, 1] with 1 being better than 0. By maximising the AUC ROC scores we try to maximize the True Positives Rate for all values of thresholds.

VI. Exploratory Data Analysis

Given the huge number of Null Values in our data, we couldn't get any valuable information without preprocessing the data, we have included our analysis as well in the next session.

VII. Preprocessing Feature Engineering

A. Identity Data

We had 41 columns corresponding to Identity Data. This data mostly described the devices used during the Transaction. We followed the following process while preprocessing:

- Dropped columns with high percentage Null Values (greater than 90%) and no correlation to the Target Variable.
- id_31
  This column contains the Browser Versions of the

devices, we had to clean up this data and get into a proper format. We tried 3 different formats

1) Browser Name + Version + SubVersion
2) Browser Name + Version
3) Browser Name

Out of these 1 seemed to give a better score and good amount of information.

- DeviceType
This was a binary value column with values 'Desktop' and 'Mobile'. We tried to use the id_31 column to impute a few missing values in this column based on the Browser Info. We filled the missing values with 'Desktop' which was the majority.
- DeviceInfo
This column gave us a lot of information about the Device used in the form of Device Models and User Agents. A lot of work went into cleaning this column into proper categories. We created 23 categories for this including 'Other'. We filled Null values with 'Other'.
- id_33
This column had the screen resolutions of the devices used. We binned the values of this columns into 6 generic categories like SD, HD, FHD, QHD, 4K, 5K and NA for Null values. This helped structure the data for Decision Tree Algorithms.
- id_34
Here the values were categorical with a string component, we just converted it into a proper integer.
- id_14
This column had values which were multiples of 60, and ranged from -600 to +720. 720 is the same as 12 hours. So we can safely assume this represents the Timezone of the Transaction or Timezone present in the Device Settings. We simply divide this column by 60. Addition of this column resulted in a better score for us.
- The remaining columns consisted of values like 0, 1, Found, Not Found, Unknown, etc. We just filled Null values with such values like Not Found, Unknown, 0, etc.

B. *Transaction Data*

The Transaction Data has lots of Null values and a huge number of unnamed columns. The Data can be seen as

- V Columns
These comprise a major chunk of the Transaction Data. We have 339 V columns. Trying to find correlation among all and reducing it to a smaller number of columns is a difficult task. One amazing approach to handle such columns was shown by a Kaggler Chris Deotte in this notebook [Link]
  - We begin on a basic assumption that, if some columns have the same number of Null values, then they will be correlated. We can understand this by a small example. A student if asked about his Employment Status, Job Salary, Position, etc will leave all these correlated columns empty. These empty values reflect as Null values in the data.
  - So based on this assumption we group the V columns based on their Null Values and plot the correlation matrix.
  - If these columns are highly correlated we find one representative column.
  - We find a representative column which gives maximum information, so we find one with the maximum different values present.

After repeating this process for the V, C, D columns we end up with a smaller number of columns which capture the maximum information possible.

- addr1 & addr2 These columns contain locations. Which means these are categorical columns. Given the huge number of values present we decided to take only the top addresses into consideration and group all others into 'Others'. Below graphs give a good idea

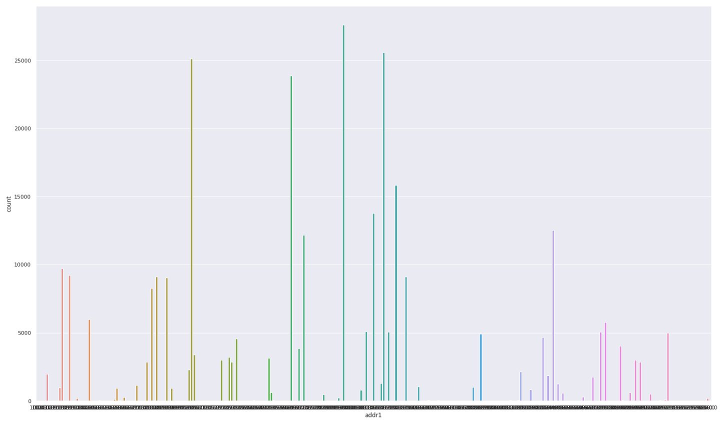


Fig. 2. Addr1 counts. Here we choose the cities with count above 5000 as categories and group remaining as 'Others'

- P_email domains and R_email domains We do something similar like we did for the addresses above. Here we look at the % of Fraud Transactions for the given Email Address and take the top emails IDs and group the rest into Others.
- Cards 1 - 6 & M1 - M9 We simply label encode all these columns.
- TransactionDT On running initial models on the data, this column had the highest importance on fitting the model. One interesting observation was that the first few observations were about 86400, which in seconds translates to a single day, i.e. 24 hours. So we decided to treat it as number of days. We tried different ways to deal with this data.
  1) Divide by 86400
  2) Divide and round off to nearest integer
  3) Split into 2 columns
     - Transaction Day = TransactionDT//86400
     - Transaction Time = (TransactionDT/86400 - TransactionDT//86400)

Out of all these 1 worked the best for our models.

VIII. Data Split

We tried 2 approaches before finalising our validation set

- A 75% 25% Split sorted on TransactionDT
  This approach seemed to give proper results but after a point they weren't consistent with the leaderboard results.
- A stratified Train Test Split with 25% Test Data.

The second approach gave us a better idea about our model performance.

IX. Hyperparameter Tuning

We used Hyperopt to tune the hyperparameters of our models, it uses Bayesian Optimisation Algorithms to arrive at hyperparameters which minimize our loss. We also tried manual tuning in few cases to see how sensitive our model was to certain changes.

X. Modelling

A. Various Classification Models

We tried basic classification models initially but didn't get good results due to the huge number of Null values present in the Data.

1) Logistic Regression
2) SVM

B. 2 Model Setup

Given the nature of our data and the absence of Identity Data for a huge number of Transactions we decided to try out a 2 model setup. We split the dataset and Trained 2 different models on 2 different subsets. This is done to deal with the huge number of null values on combining the Transaction & Identity Data.

1) Model 1
   We trained this completely on the Transaction Data. With the same Preprocessing as before. We used XGBoost as our model in this case. The size of this dataset was 354324 Observations.
2) Model 2
   We trained this model an the subset of Transactions which had corresponding Identity Data. We used XGBoost here as well to train our model. The size of this dataset was 89000 observations.
   We tried 3 different approaches to combine this data in the following ways
   a) Add the Probabilities.
   b) Average the Probabilities.
   c) Take the lower values of the probabilities.

   Plots of all the predictions made by both models shows us how the huge difference in the amount of Data for both the models affects the predictions of these models.

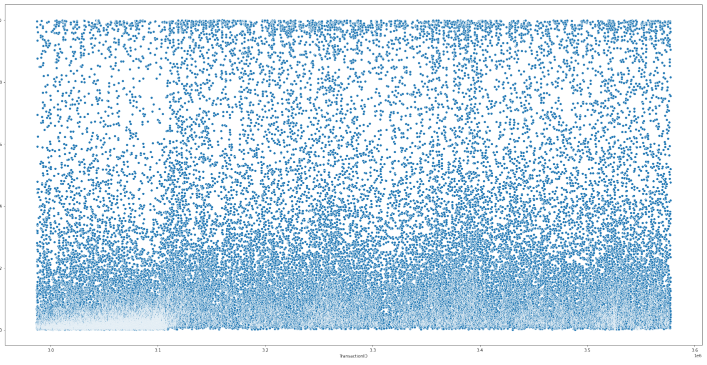


Fig. 3. Predictions made by Model 2

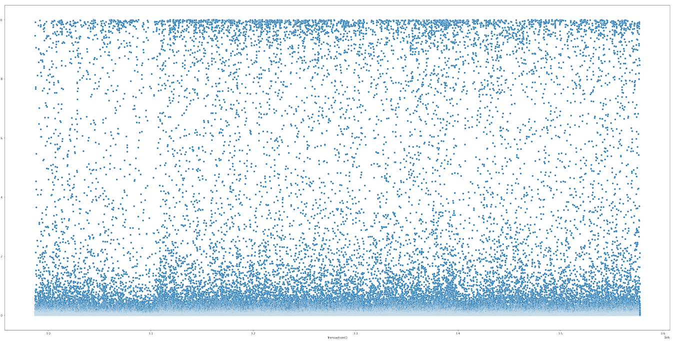


Fig. 4. Predictions made by Model 1. Huge amounts of Data helped the model. We can see that there arent many closer to the 0.5 probability. The higher values are higher and lower values are lower.

C. Boosting Models

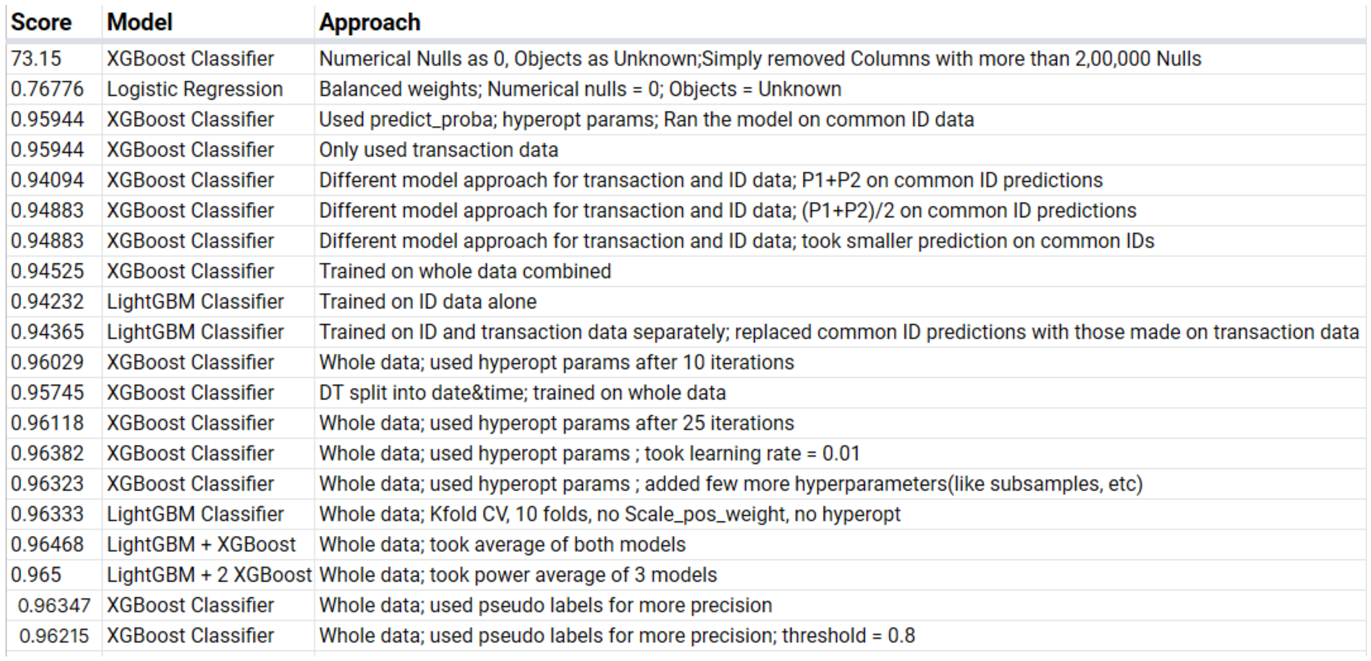
We tried Boosting models like XGBoost (Extreme Gradient Boosting) & LightGBM (Light Gradient Boosting Model) using which we got very good results on our data. In the end we averaged the predictions of these models to get our final submission scores.

D. Pseudo Labels

Pseudo Labels is a very interesting concept which we read about online. We train our data on a given dataset and predict the labels for our testset. We separate the test observations for which we have high confidence. We take these and add to our training dataset and train our model again on this to get a better score. We tried this out and did get marginal improvements over our original results. We tried different thresholds as well for confidence in our initial predictions. All these have been summarised in the table.

Acknowledgment

We, team RuSTy, would like to thank Prof. G Srinivasaraghavan & Prof. Neelam Sinha for teaching this course. I would also like to thank Saurabh Jain, Tejas Kotha, Tanmay Jain for providing proper guidance and helping us when stuck. Also, thanks to all the Teaching Assistants for this course for the amazing tutorial sessions and assignments.



Score	Model	Approach
73.15	XGBoost Classifier	Numerical Nulls as 0, Objects as Unknown;Simply removed Columns with more than 2,00,000 Nulls
0.76776	Logistic Regression	Balanced weights; Numerical nulls = 0; Objects = Unknown
0.95944	XGBoost Classifier	Used predict_proba; hyperopt params; Ran the model on common ID data
0.95944	XGBoost Classifier	Only used transaction data
0.94094	XGBoost Classifier	Different model approach for transaction and ID data; P1+P2 on common ID predictions
0.94883	XGBoost Classifier	Different model approach for transaction and ID data; (P1+P2)/2 on common ID predictions
0.94883	XGBoost Classifier	Different model approach for transaction and ID data; took smaller prediction on common IDs
0.94525	XGBoost Classifier	Trained on whole data combined
0.94232	LightGBM Classifier	Trained on ID data alone
0.94365	LightGBM Classifier	Trained on ID and transaction data separately; replaced common ID predictions with those made on transaction data
0.96029	XGBoost Classifier	Whole data; used hyperopt params after 10 iterations
0.95745	XGBoost Classifier	DT split into date&time; trained on whole data
0.96118	XGBoost Classifier	Whole data; used hyperopt params after 25 iterations
0.96382	XGBoost Classifier	Whole data; used hyperopt params ; took learning rate = 0.01
0.96323	XGBoost Classifier	Whole data; used hyperopt params ; added few more hyperparameters(like subsamples, etc)
0.96333	LightGBM Classifier	Whole data; Kfold CV, 10 folds, no Scale_pos_weight, no hyperopt
0.96468	LightGBM + XGBoost	Whole data; took average of both models
0.965	LightGBM + 2 XGBoost	Whole data; took power average of 3 models
0.96347	XGBoost Classifier	Whole data; used pseudo labels for more precision
0.96215	XGBoost Classifier	Whole data; used pseudo labels for more precision; threshold = 0.8

Fig. 5. Final Results

REFERENCES

[1] XGBoost Documentation
[2] Sci-kit Learn Documentation
[3] Hyperopt Documentation
[4] Handling Class Imbalance using XGBoost - Machine Learning Mastery
[5] Credit Card Fraud Detection Using Meta-Learning:Issues and Initial Results