IEEE Computational Intelligence Society

Kaggle Fraud Detection

Team: Luke Gray, Alyssa Wei, Jose Gonzalez, Fred(Lefan) Cheng and Rajesh Earlu

Table of Contents

Project Description

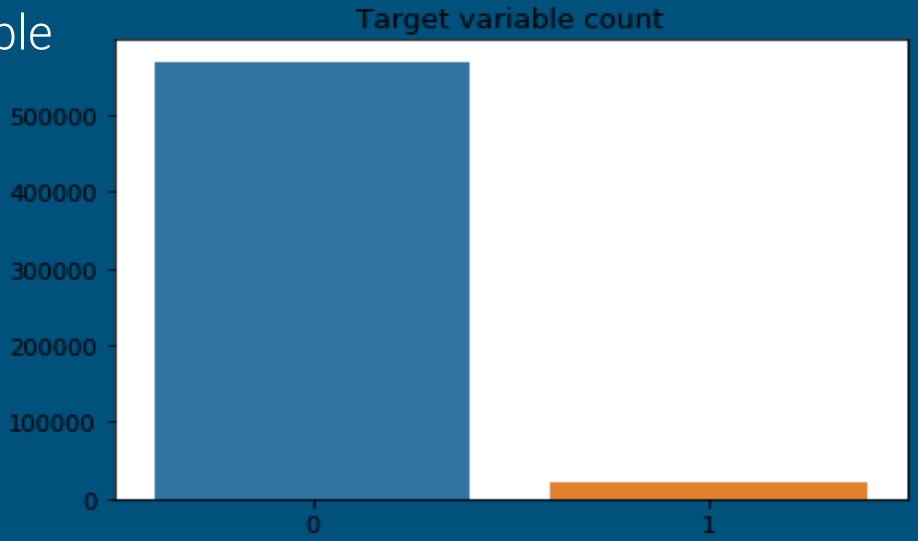
- We completed the IEEE-CIS (Institute of Electrical and Electronic Engineers) Fraud Detection competition on Kaggle
- The dataset of credit card transactions is provided by the Vesta Corporation, described as the world's leading payment service company
- The dataset includes identity and transaction CSV files for both test and train
- Train dataset: 590540 x 433; Fraud transactions: 20663
- Target variable ‘isFraud’

Important Features and Variables

- TransactionDT: timedelta from a given reference datetime (not a timestamp)
- TransactionAMT: transaction payment in USD
- ProductCD: product code, the product for each transaction
- card1-card6: payment card information, such as card type
- addr: address
- dist: distance
- P_ and (R_)emaildomain: purchaser and recipient email domain
- C1-C14: counting, addresses and other things, actual meaning is masked.
- D1-D15: timedelta, such as days between previous transaction, etc.
- M1-M9: match, such as names on card and address, etc.
- Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

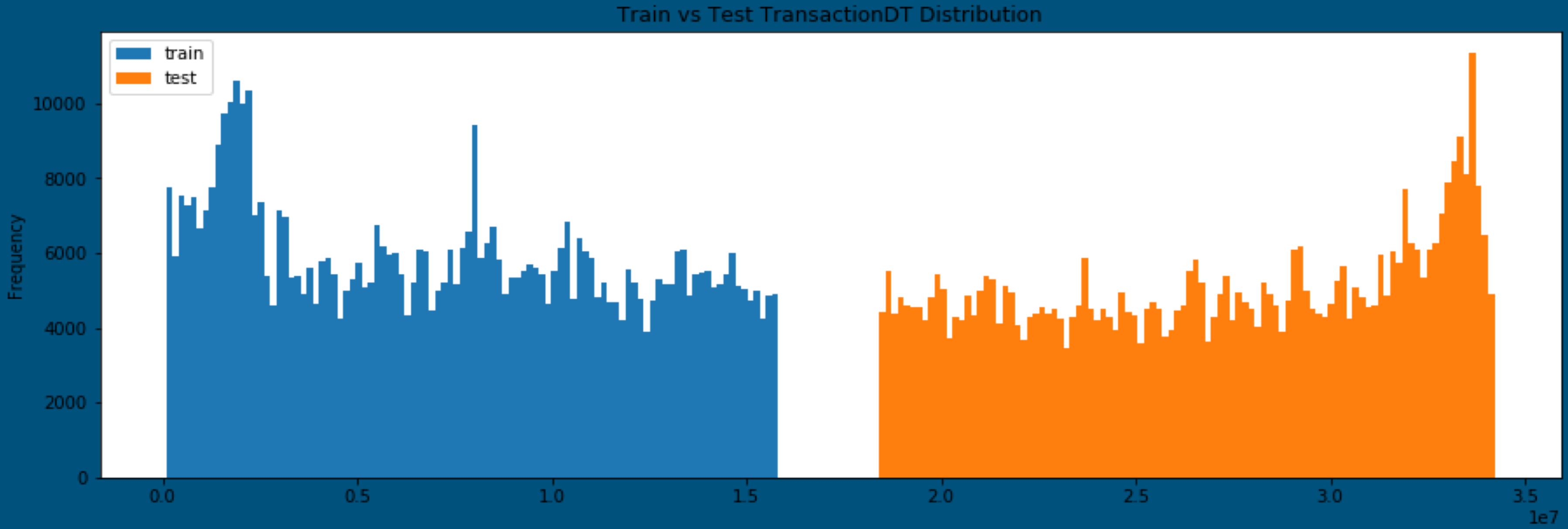
Exploratory Data Analysis (EDA)

- One of the first things we noticed when conducting our EDA was the sparsity of the dataset
- Only ~3.5% of the total transactions were positively classified as part of the 'isFraud' variable

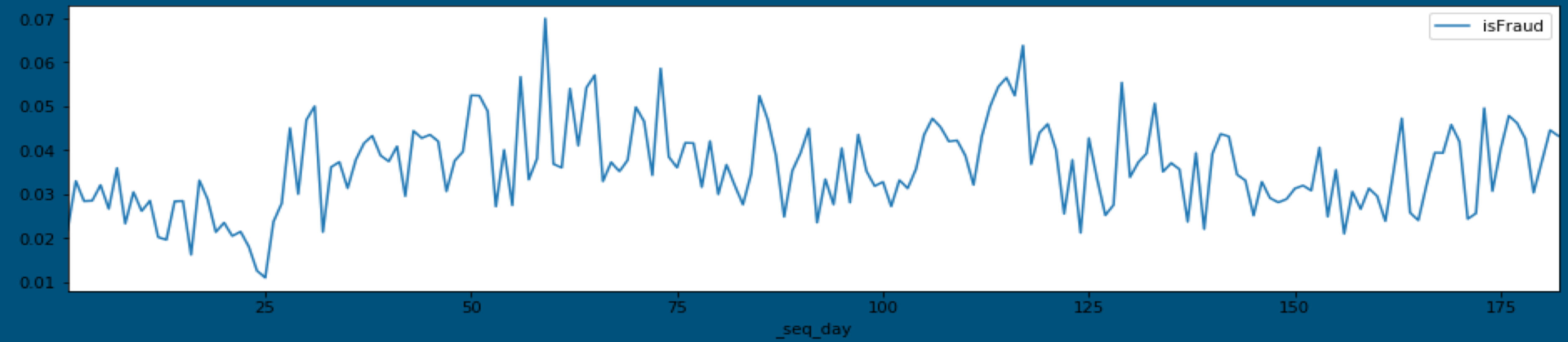


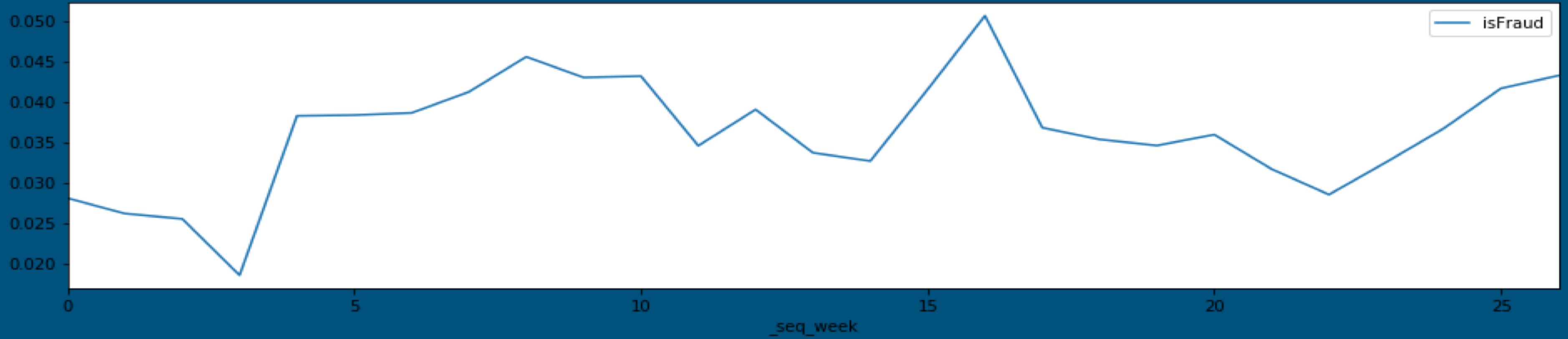
EDA

- Another observation was immediately apparent is imbalanced nature of the data. This shows that 'TransactionDT' is a timedelta gap, not a timestamp

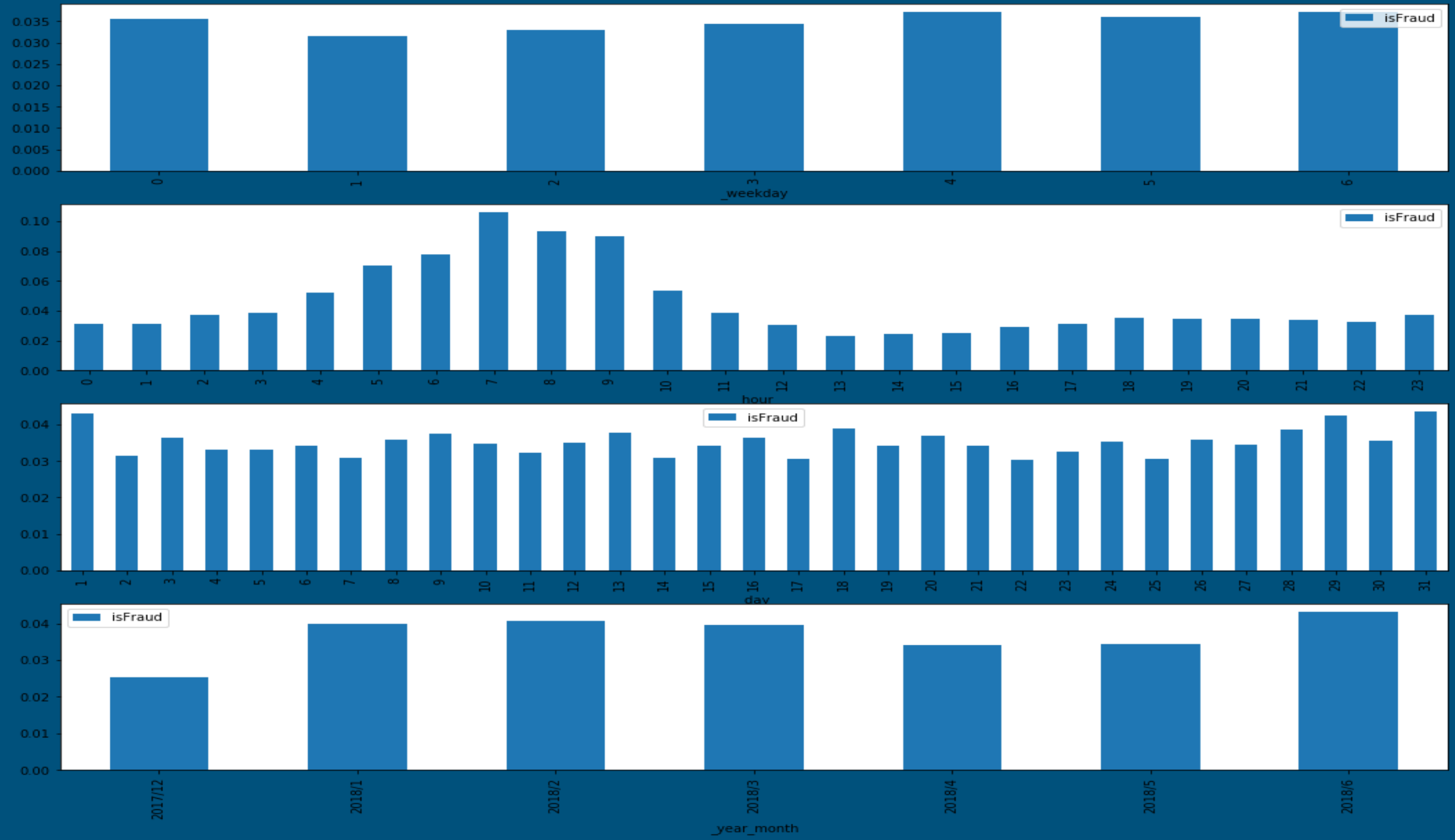


EDA Timedelta



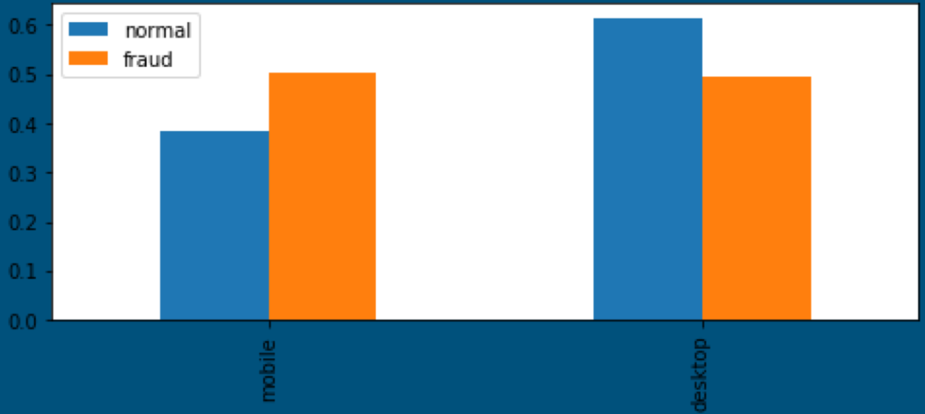


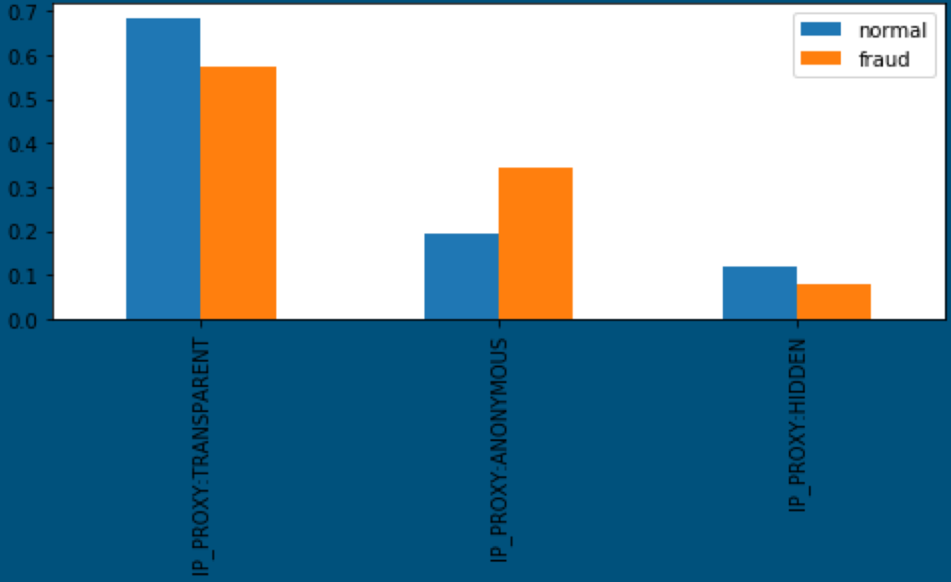
EDA Timedelta



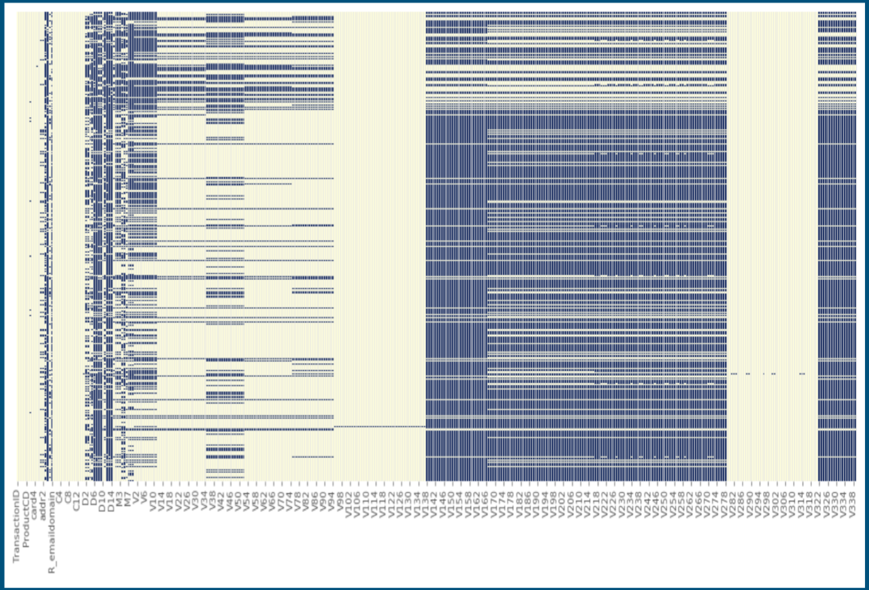
EDA

- Target variable ‘isFraud’ is more prevalent in the mobile ‘DeviceType’ as well as more prevalent in the ‘IP_PROXY:ANONYMOUS’ based on ‘id_31’
- Visualizations of other interesting findings below:





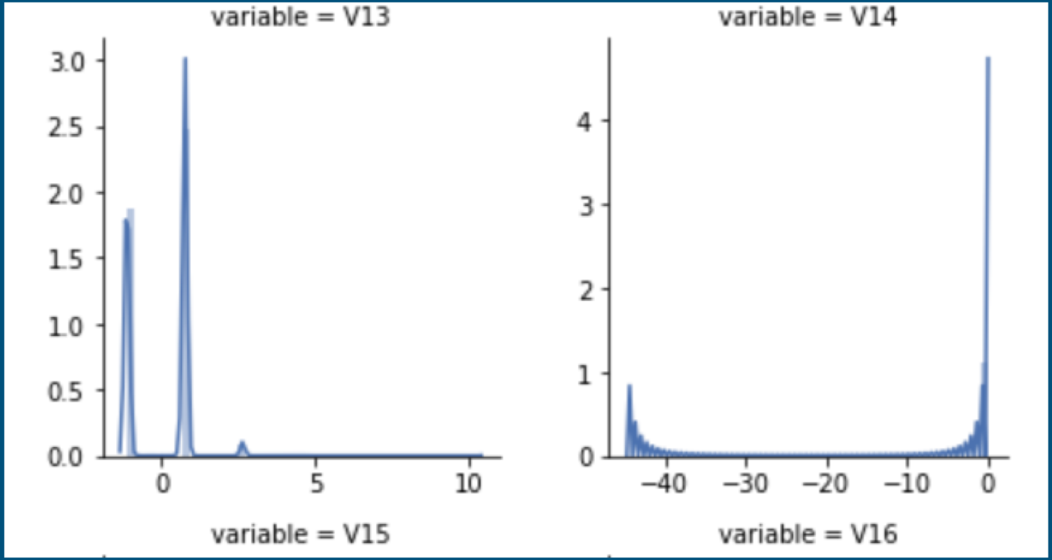
Missingness and Imputation



Dataset has very high percentage of missing values, especially the V columns.

Missingness and Imputation

Anonymized columns not only had a high mount of missing data, but their distributions weren't normal.



Missingness and Imputation

Plan A:

1. Drop columns with over 80% of missing value.
2. Impute columns with less than 20% missing values by the mean of each row's product ID.
3. Use machine learning model with the columns without missing value as input variables to predict the remaining missing values.
4. Precise but time consuming.
5. Hard to impute for anonymized data.

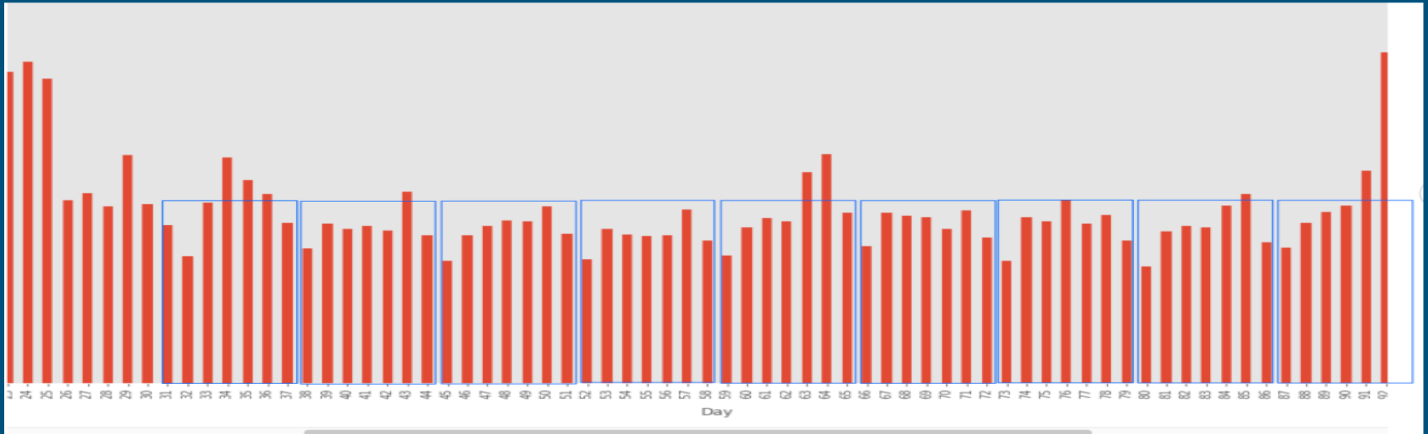
Plan B:

1. Impute all the missing values with -999 first which is very fast and model can still find some pattern instead of losing information by dropping them.
2. Do more complex imputation afterwards if we have extra time.

Feature Engineering

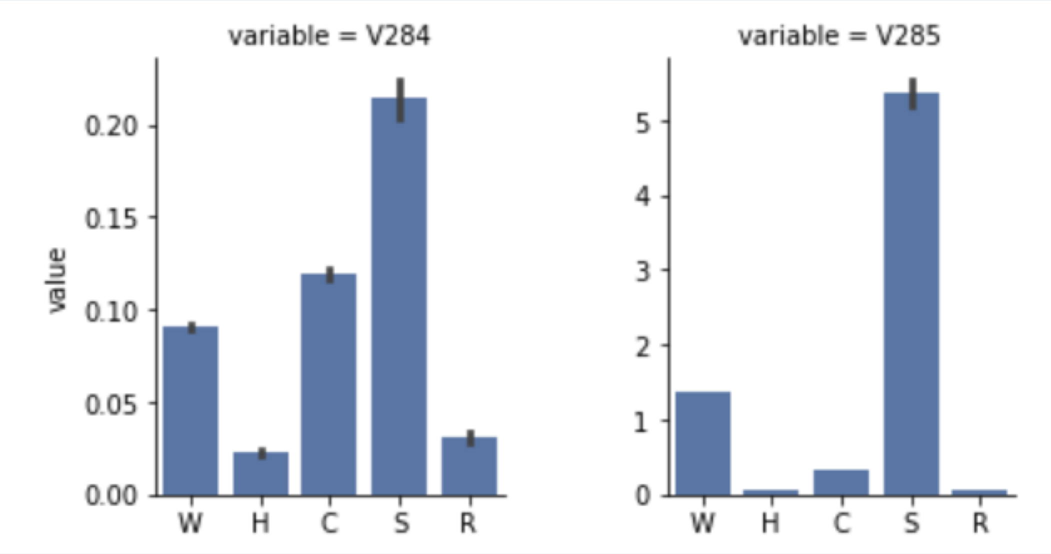
Given the sparsity and anonymity of our data, feature engineering was a central focus of the project.

The most intuitive way to tackle this was first engineering on known features, namely, the TransactionDT and TransactionAmt.



Engineering on Anonymized V-columns

Considering that V columns occupied most of our data and engineered by the company themselves we indexed on the trends in those values to impute.

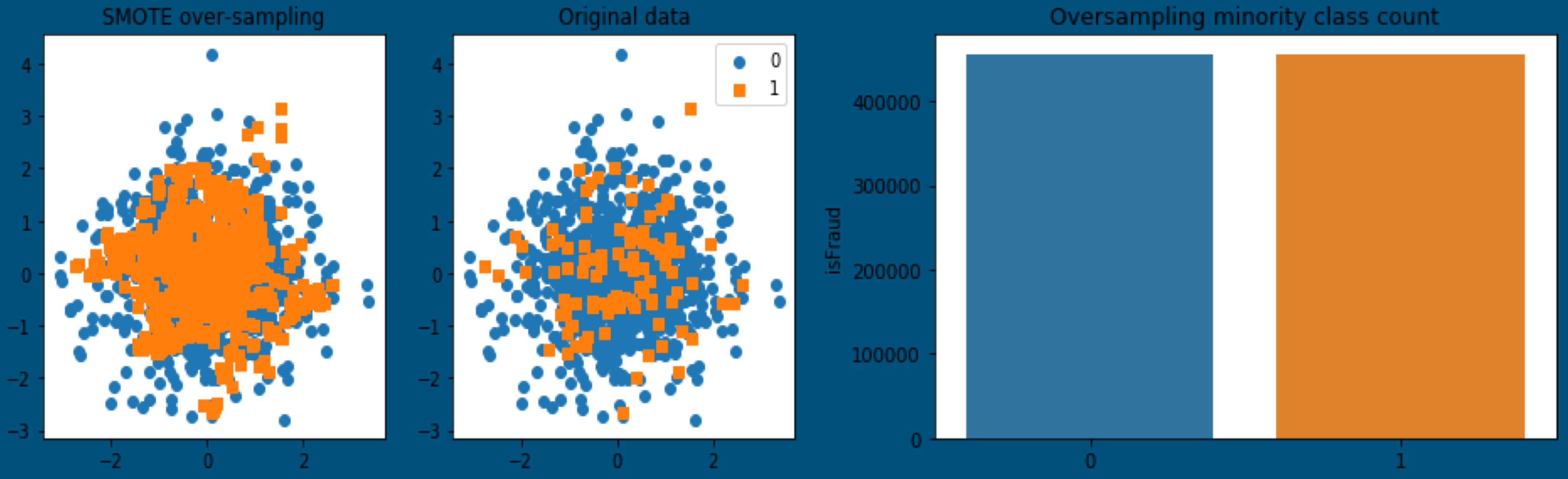


Dimensionality Reduction

- Attempted:
  - PCA
  - Lasso
  - Sparse PCA
- It was essential to run PCA before balancing the created values from oversampling wouldn't be influenced by uninformative features.

Balancing Techniques

- To help with the imbalance we showed earlier, we used balancing techniques such as oversampling the minority class.
- We ultimately used SMOTE, which is shown graphically below:



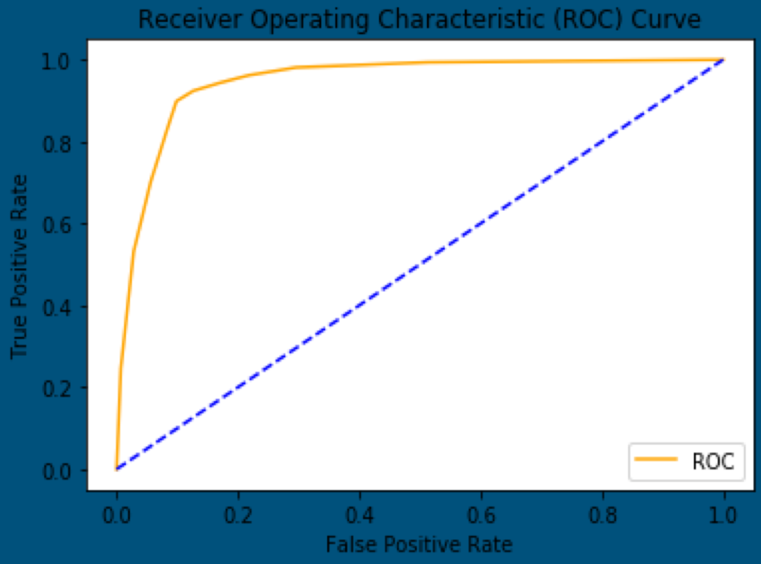
Modeling

- XGBoost
- Accuracy: 0.98
- Precision: 0.75
- Recall: 0.45

- LightGBM
- AUC: 0.972328

Scoring

- Competition submissions evaluated on area under the ROC curve between predicted probability and the observed target.
- A graphical example made on sample data is shown below:



Future Work

- If we had more time to work on this project:
- We would utilize cloud computing services.
    - Before oversampling and feature engineering, our workspace could only handle so much before running into Memory Errors.
    - The amount of data also made grid-searching impossible unless only considering a very limited range of hyperparameters.
- Further feature engineer to help deal with the sparsity of our data.
- Proper optimization techniques
    - GridSearchCV, bayes_opt, GPyOpt, stratified KFold