Finding Fraudsters:

Detecting Credit Card Fraud with Machine Learning

Zamiul Alam, Alex Gekow, Kazi Aatish Imroz, Roshan Joshi, Kayode oyedele

Outline

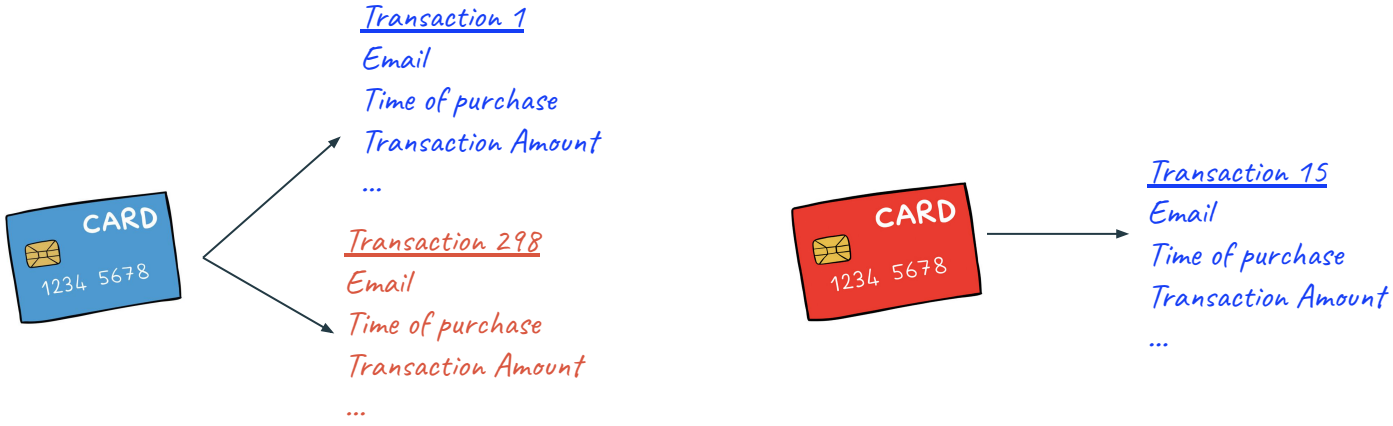
1. Problem Statement
2. Data Exploration
3. Data Preparation
4. Modeling
5. Results

IEEE-CIS Fraud Detection

Credit cards are susceptible to *fraudulent transactions*, transactions purchased without the consent of the owner

The IEEE-CIS fraud detection dataset includes a list of *legitimate* and *fraudulent* transactions with information regarding each transaction, but the associated credit card is **not** included

Each transaction may or may not be made from a unique credit card in the dataset



Dataset

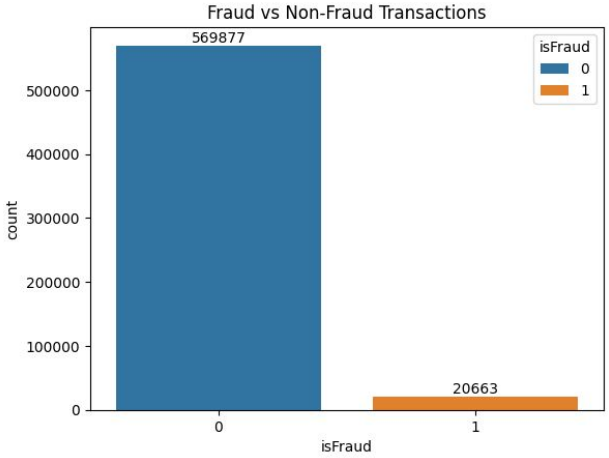
Each transaction includes information - most of which whose *meaning is masked*:

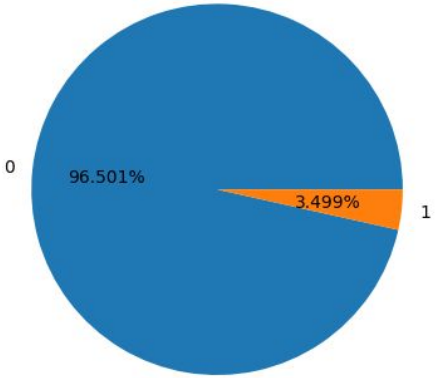
- Transaction Amount
- 15 Time related features (time of purchase, time since credit card was created, etc.) **masked**
- Categorical features:
  - Product category, Credit card Info, Locations, email domains **masked**
  - Purchase device info (if purchase made on a computer) **masked**
- Identity information (IP, ISP, Proxy, browser, os, etc.) **masked**
- 339 Vesta engineered features **masked**

Huge challenge to do EDA on hundreds of variables, most of which whose meanings are unknown!

How do we find features relevant for fraud detection and use them to our advantage?

Exploratory Data Analysis



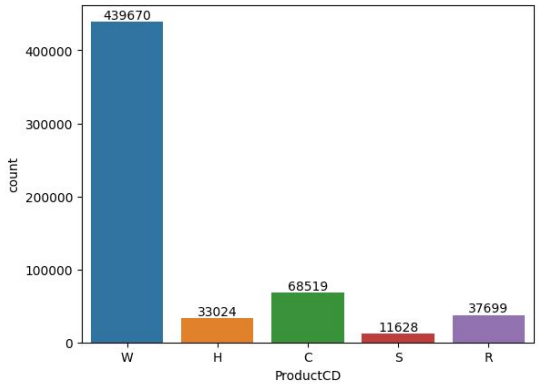


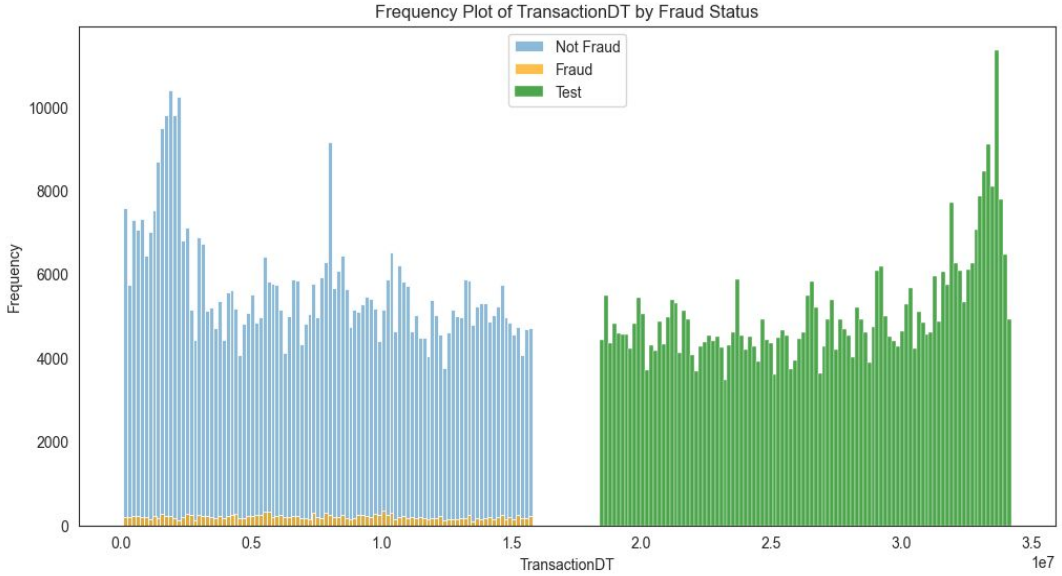
- The test dataset contained 590,540 transactions.
- Non-fraud transactions were denoted by 0 and fraud transactions by 1.
- Fraudulent transactions account for roughly 3.5% of all transactions.
- So, we had a highly imbalanced dataset.

Exploratory Data Analysis

- The column, TransactionDT was given in seconds and we found the data to be forward in time. The train and test data were separated by a month.

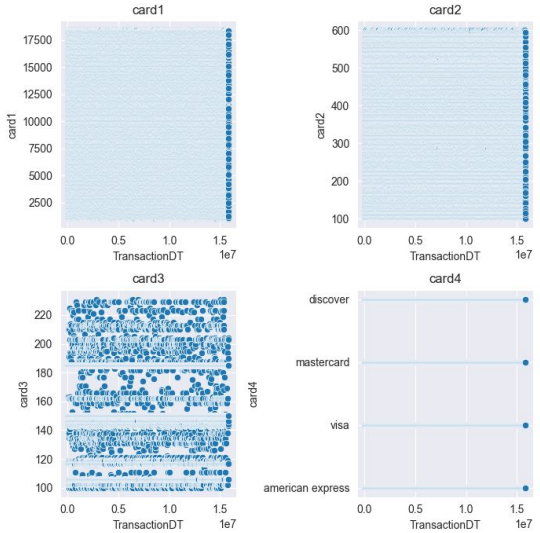
- We made countplots of various features such as the one for ProductCD as shown here.

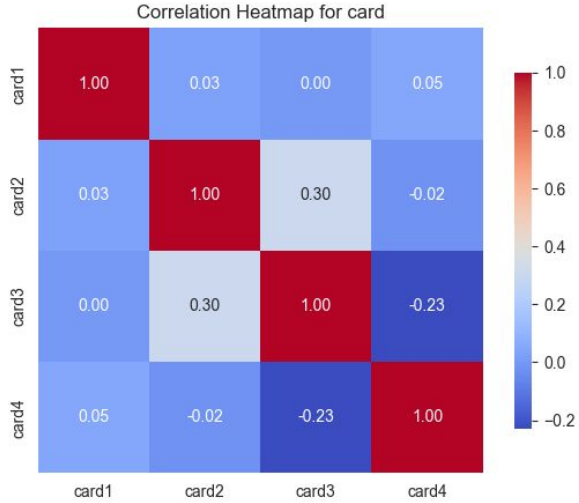


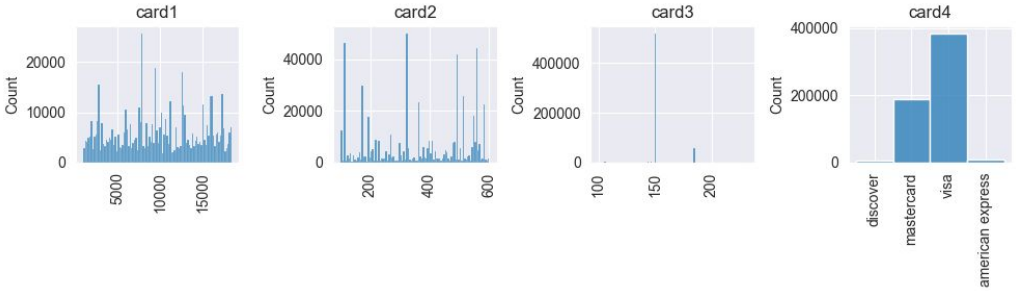


Exploratory Data Analysis

- Since most of the features were masked, we utilized correlation heatmaps, countplots and scatterplots to get some idea about the features. We show card1 to card4 as an example.







Feature Selection

Many features are highly correlated

	V330	V331	V332	V333	V334	V335	V336	V337	V338	V339
V330	1.00	0.77	0.86	0.89	0.05	0.65	0.41	0.25	0.78	0.61
V331	0.77	1.00	0.95	0.94	0.06	0.59	0.39	0.31	0.75	0.57
V332	0.86	0.95	1.00	0.99	0.05	0.65	0.41	0.26	0.80	0.58
V333	0.89	0.94	0.99	1.00	0.06	0.66	0.43	0.30	0.82	0.62
V334	0.05	0.06	0.05	0.06	1.00	0.70	0.91	0.04	0.05	0.06
V335	0.65	0.59	0.65	0.66	0.70	1.00	0.91	0.17	0.55	0.41
V336	0.41	0.39	0.41	0.43	0.91	0.91	1.00	0.12	0.35	0.27
V337	0.25	0.31	0.26	0.30	0.04	0.17	0.12	1.00	0.74	0.91
V338	0.78	0.75	0.80	0.82	0.05	0.55	0.35	0.74	1.00	0.94
V339	0.61	0.57	0.58	0.62	0.06	0.41	0.27	0.91	0.94	1.00

Many features are sparse

	V330	V331	V332	V333	V334	V335	V336	V337	V338	V339
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
590535	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
590536	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
590537	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
590538	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
590539	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Try removing features if they are

1. highly correlated with other features with more unique values
2. Highly correlated with other features with a stronger correlation to the target (isFraud)
   a. Pearson Correlation
   b. Distance Correlation
3. Show low feature importance in a simple classifier

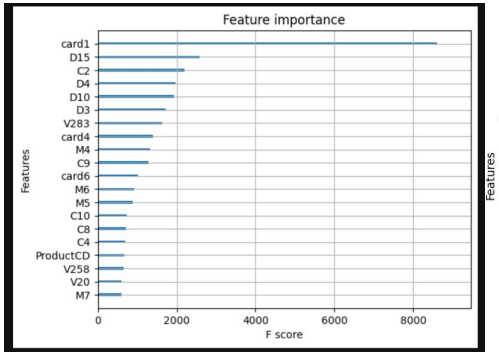
Consider dropping sparse features if they

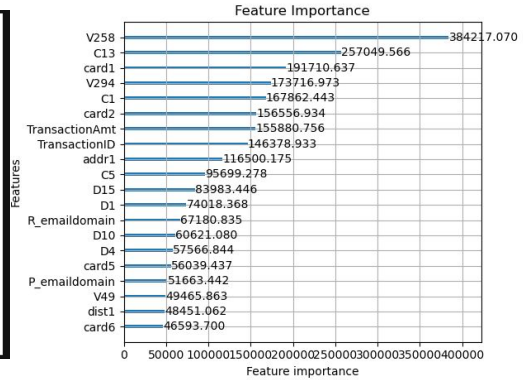
1. Do not improve training
2. Are correlated with more densely populated features

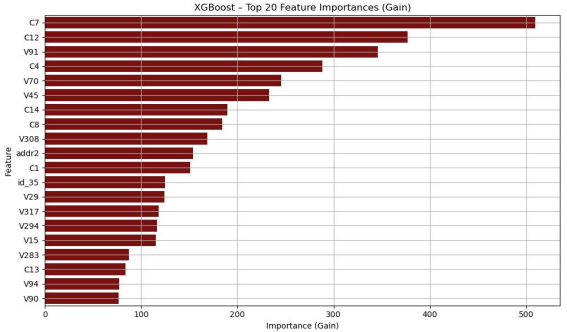
-or-

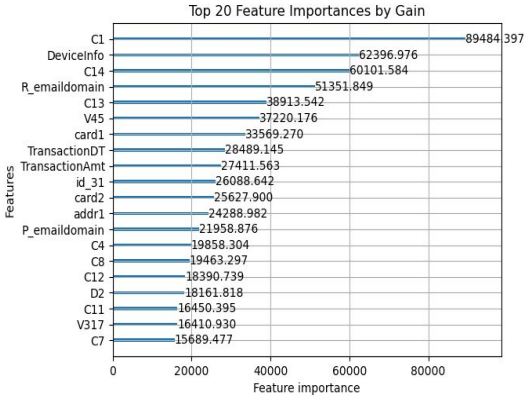
1. Fill NaN with dummy values
2. Replace NaN with mean/median/mode

Feature Selection





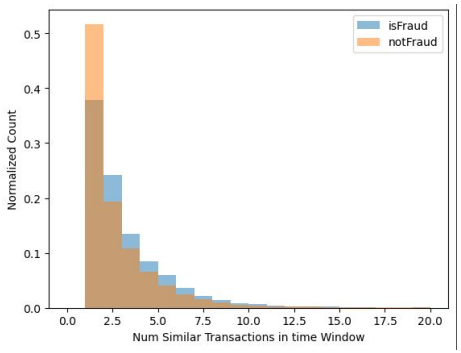




Run several selection algorithms in parallel and **keep all** features which are selected **three** times or **more.**

Ensure any feature which makes the top 5 on the feature importance list (using LightGBM and XGboost) makes the selection.

Feature Engineering



For a given transaction: Check how many other similar transactions occur within +/- 5 minutes

Transactions are grouped together if thought to be from the same credit card

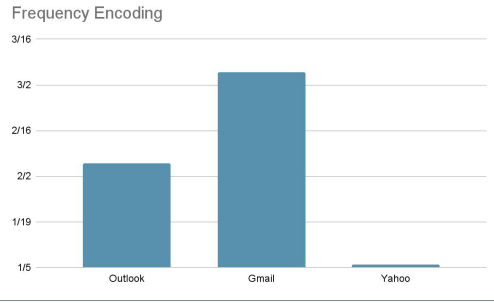
Aggregate information is computed per card

Transaction 1
Transaction Amount
Transaction_Amt_Avg
Transaction_Amt_Stddev

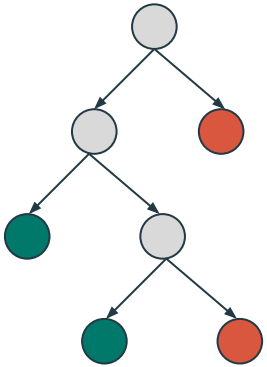
Transaction 298
Transaction Amount
Transaction_Amt_Avg
Transaction_Amt_Stddev

Frequency & Ordinal Encoding Categorical Features

Email domain	Ordinal Label
Outlook	1
Gmail	2
Gmail	2
Yahoo	3
Gmail	2
outlook	1



Model Training & Selection



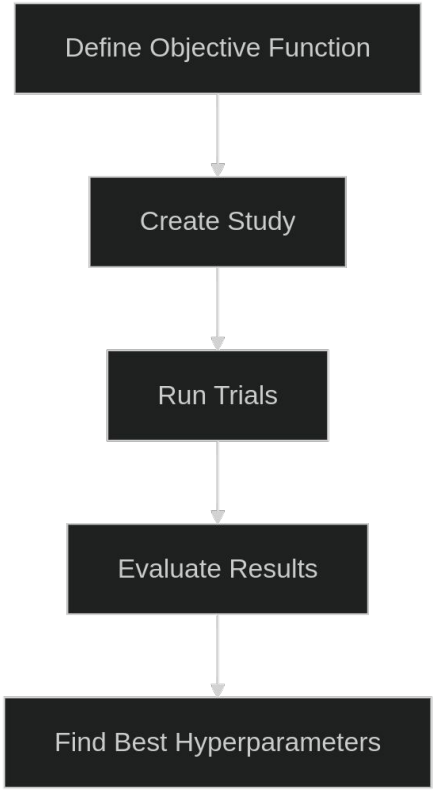
Binary trees are a practical choice when using a difficult to interpret dataset

- Standardization/regularization is not as important as with deep learning
- Binary trees can naturally handle NaN
- Easy to rank feature importance
- Fast to train

XGBoost, LightGBM and CATBoost are used to classify a transaction as fraud

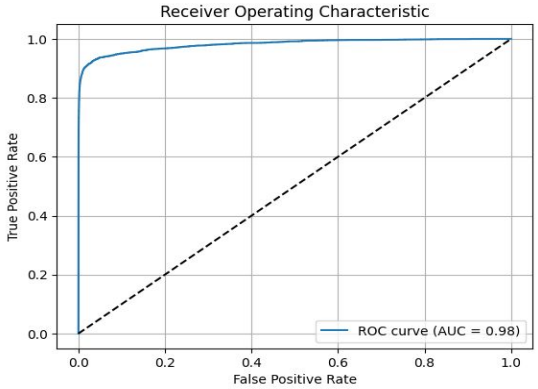
Hyperparameter optimization with optuna

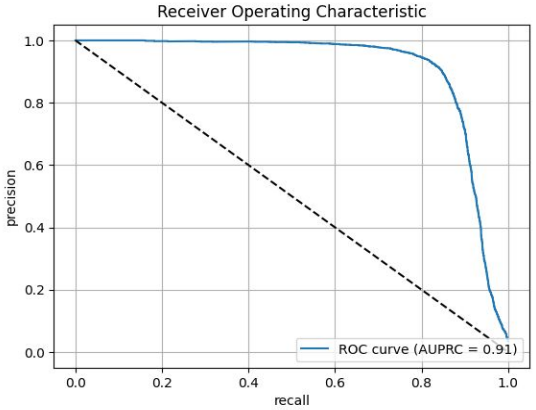
- Library for optimizing BDTs with efficient hyperparameters search algorithms
- Supports parallel processing and visualization.
- Key concepts:
  - Objective function:
    - Takes hyperparameters and returns score (eg: f1 score).
  - Study:
    - Overall optimization based on objective function.
    - Can pass optimization and pruner algorithms
  - Trial:
    - A single execution of objective function.
    - Passed to objective function.
    - Provides interface for parameter suggestion/sampling

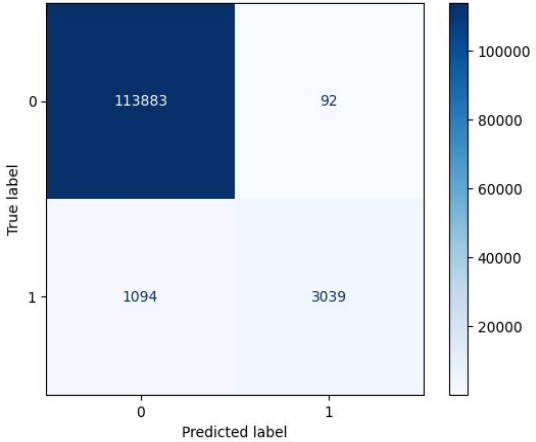


Results (XGBoost)

Results are evaluated on two datasets: subset of the training data & test data provided by kaggle which does not include class labels for isFraud







Kaggle score: 0.93 AUC (public) 0.90 AUC (private)
But AUPRC may be more insightful with unbalanced datasets!

Results

Model	AUC (Private, using kaggle test set)	AUC (Public, using kaggle test set)	AUPRC (Splitting the training set)
XGBoost	0.905	0.928	0.911
LightGBM	0.896	0.925	0.868
CATBoost	0.890	0.914	0.893
Ensemble	0.907	0.933	

Future Directions

- Anomaly detection is by its nature suitable for anomaly detection. Can try and optimize anomaly detection algorithms.
- Given enough time, one could do a full grid search to get the best possible hyperparameters rather than sampling n different combinations of hyperparameters. .

THANK YOU!