FRAUD TRANSACTION DETECTION

COMPUTING FOR DATA SCIENCES (CDS) PROJECT

Anudeep (19BM6JP54)
Swadesh (19BM6JP63)

Neha (19BM6JP30)
Vaibhav (19BM6JP24)

Srijan (19BM6JP19)

Introduction to Problem

- Fraud is a **billion-dollar business** and it is increasing every year.
- Fraud transactions are also **increasing in volume**.
- Identity fraud is a growing concern affecting both businesses and customers.
- Identifying a fraudulent transaction is becoming seemingly important.
- PwC global economic crime survey of **2018** found that **49%** of the 7,200 companies they surveyed had experienced fraud of some kind.
- Raised from **36% in 2016** indicating and increase from one-third to one-half of population.

Objective

Developing a model that can classify a transaction as fraudulent or

non-fraudulent.

Data Source

- Vesta Corporation, e-commerce payment solutions company provided the dataset.
- The dataset is available in Kaggle.
- However the variable names are masked to protect the privacy of the customers.
- The dataset consists of 590400 rows and 432 independent variables alongside a binary target variable.

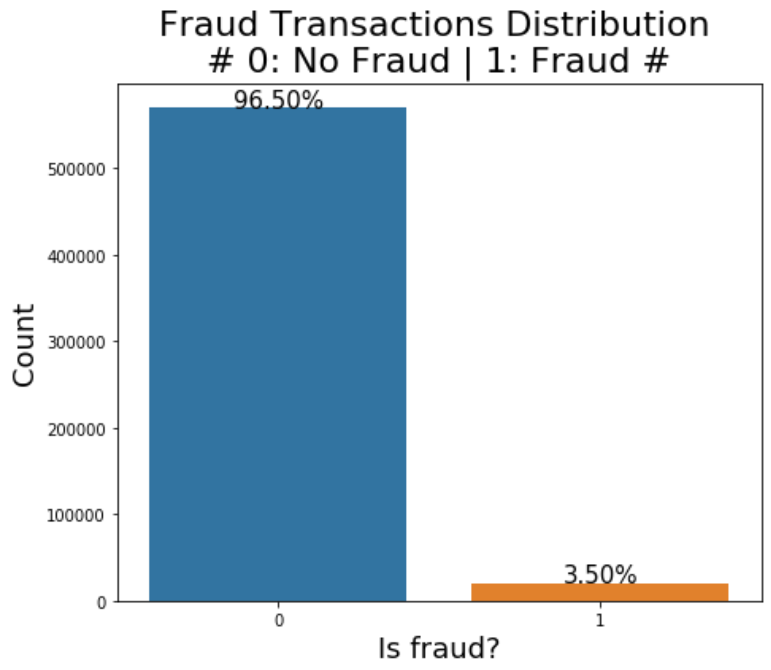
Description of Data

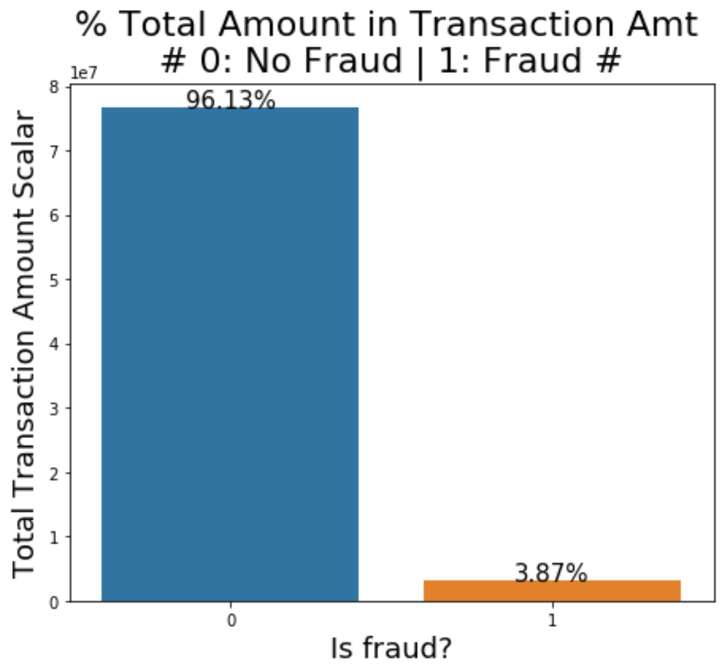
- TransactionDT: timedelta from a given reference datetime (not an actual timestamp)
- TransactionAMT: transaction payment amount in USD
- ProductCD: product code, the product for each transaction
- card1 - card6: payment card information, such as card type, card category, issue bank, country, etc.
- addr: address
- P_ and R_emaildomain: purchaser and recipient email domain
- C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The 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.
- idxx: network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc)

Memory Reduction

- Converting the datatype of columns to optimal datatype
- For Example:
- Variable taking int64 can be given int16 depending on the maximum value
- Reduces the memory allotted for the data
- This is important because the data size is large
- Might speed up the execution in low storage environments.

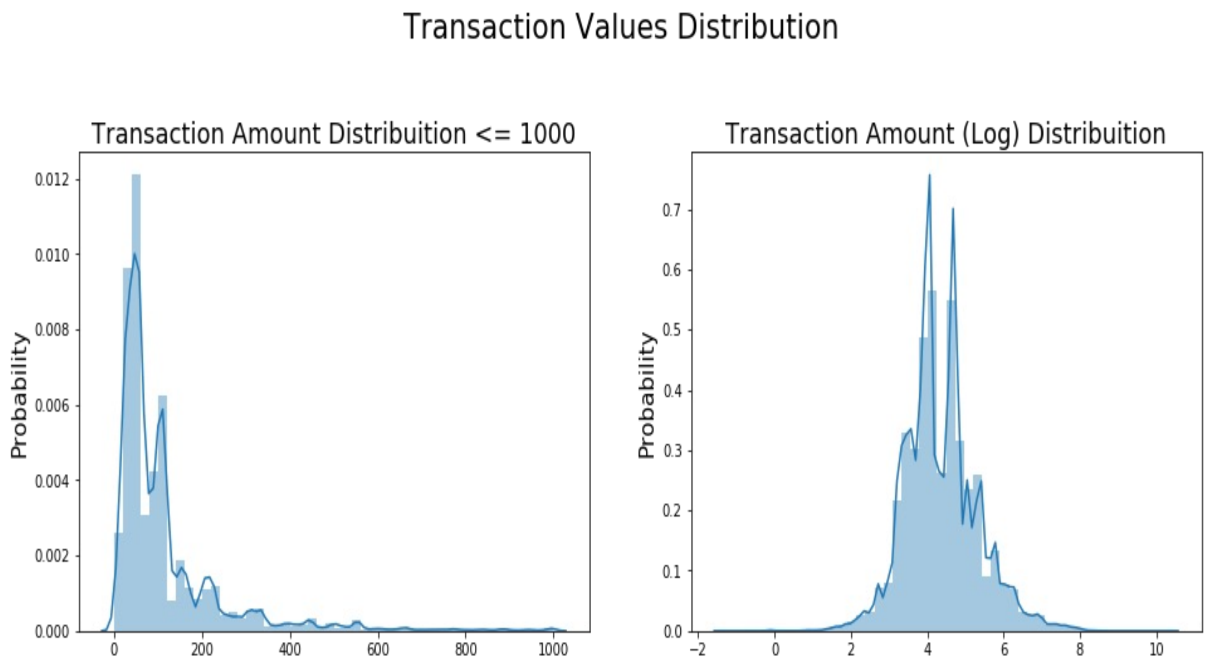
First Look at Data



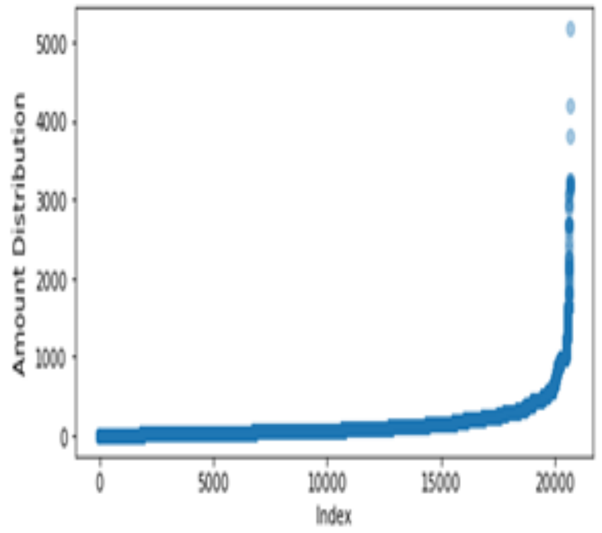


- Highly imbalanced data - 3.5% of total transactions were fraudulent
- Fraud amount accounts for 3.87% percent of total amount

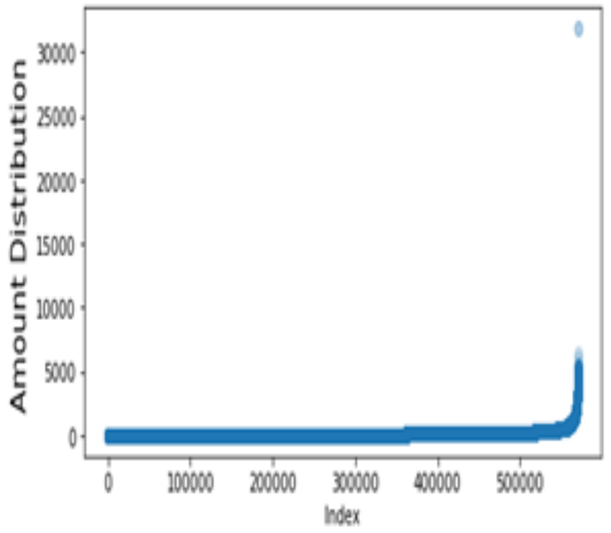
Transaction Amount Distribution



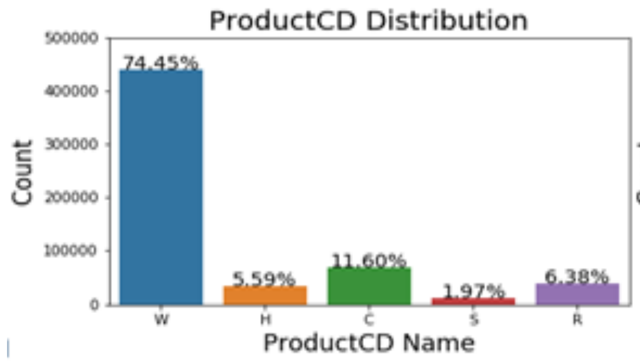
FRAUD TRANSACTION

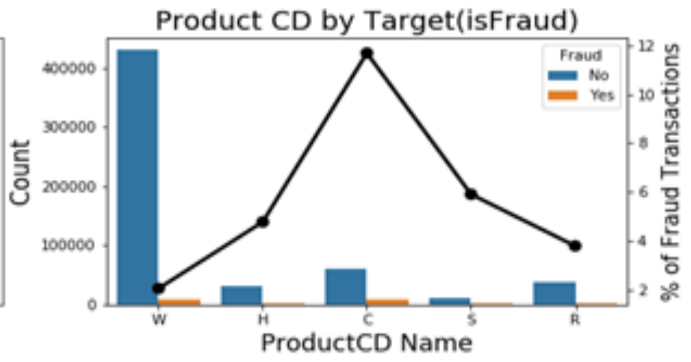


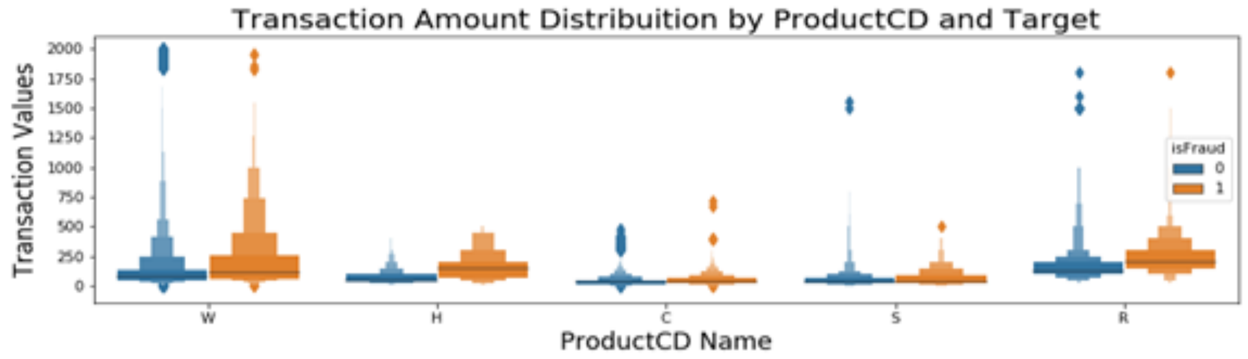
NON-FRAUD TRANSACTION



Product CD Distribution

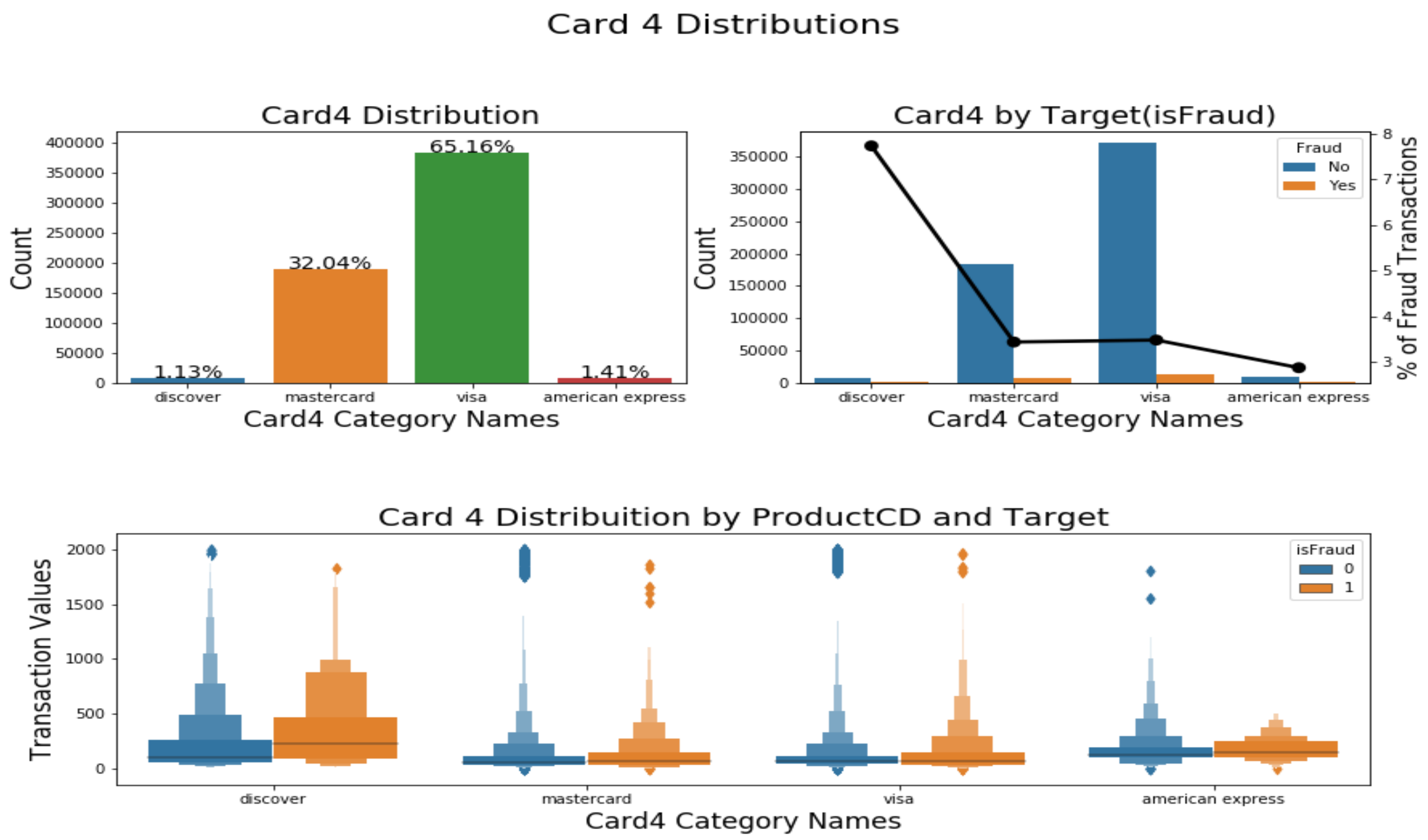






- Product code, the product for each transaction
- Product W, and C are the most present productCD elements
- productCD element C has more chance of being a fraud element
- W, H and R the distribution of Fraud values are slightly higher than the Non-Fraud transactions

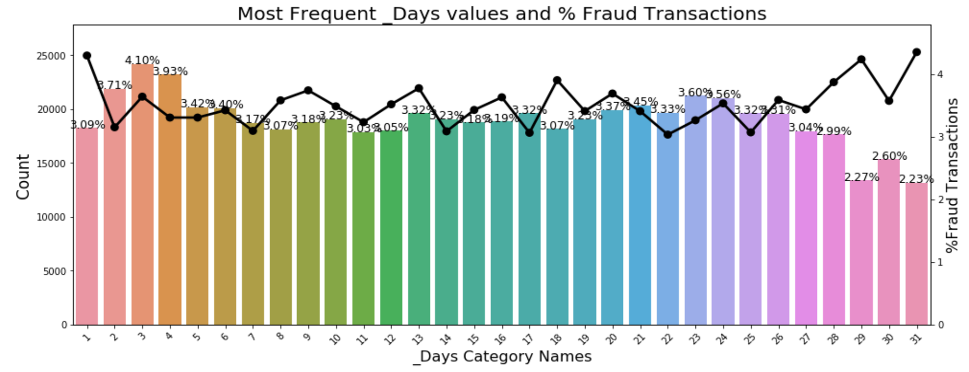
Card_4 distribution analysis

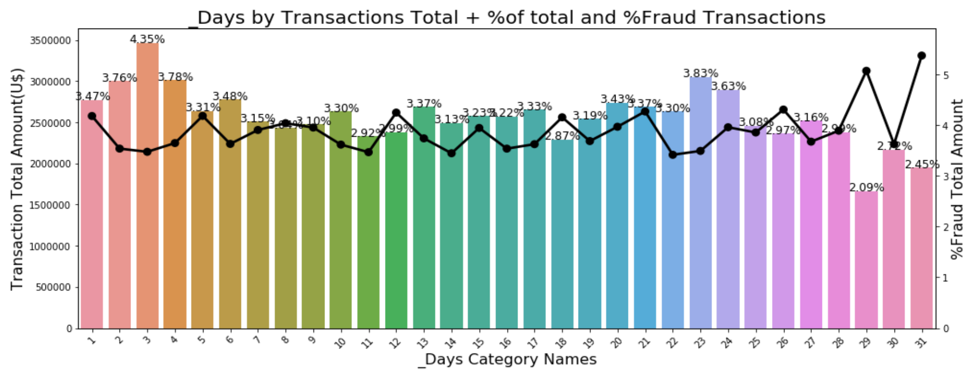


Visa and mastercard are the most popular cards used, but a discover card is more likely to produce a fraudulent transaction, whereas AmEx card is least likely to produce a fraudulent transaction. We have a highest value in discover(~8%) against ~3.5% of Mastercard and Visa and 2.87% in American Express

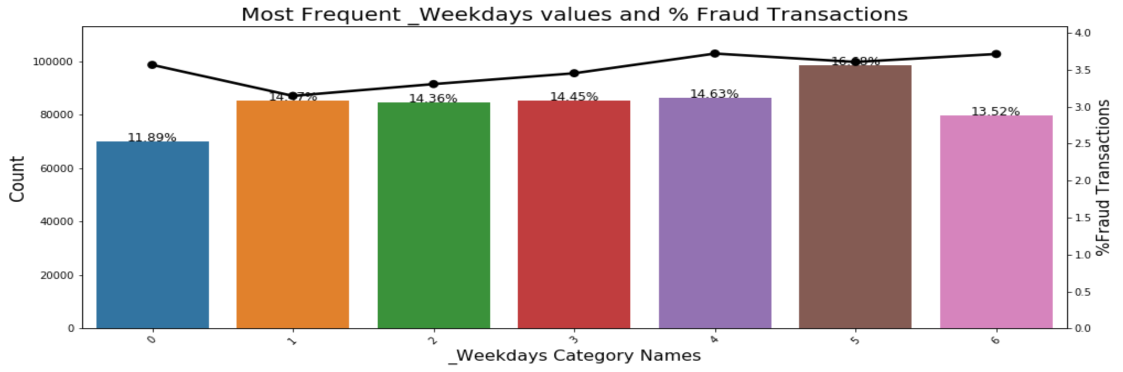
Time Series Analysis of transactions

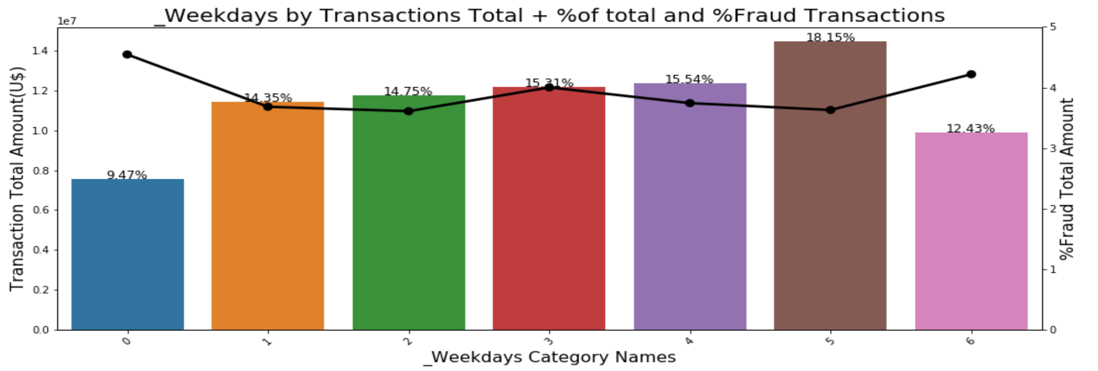
_Days Distributions



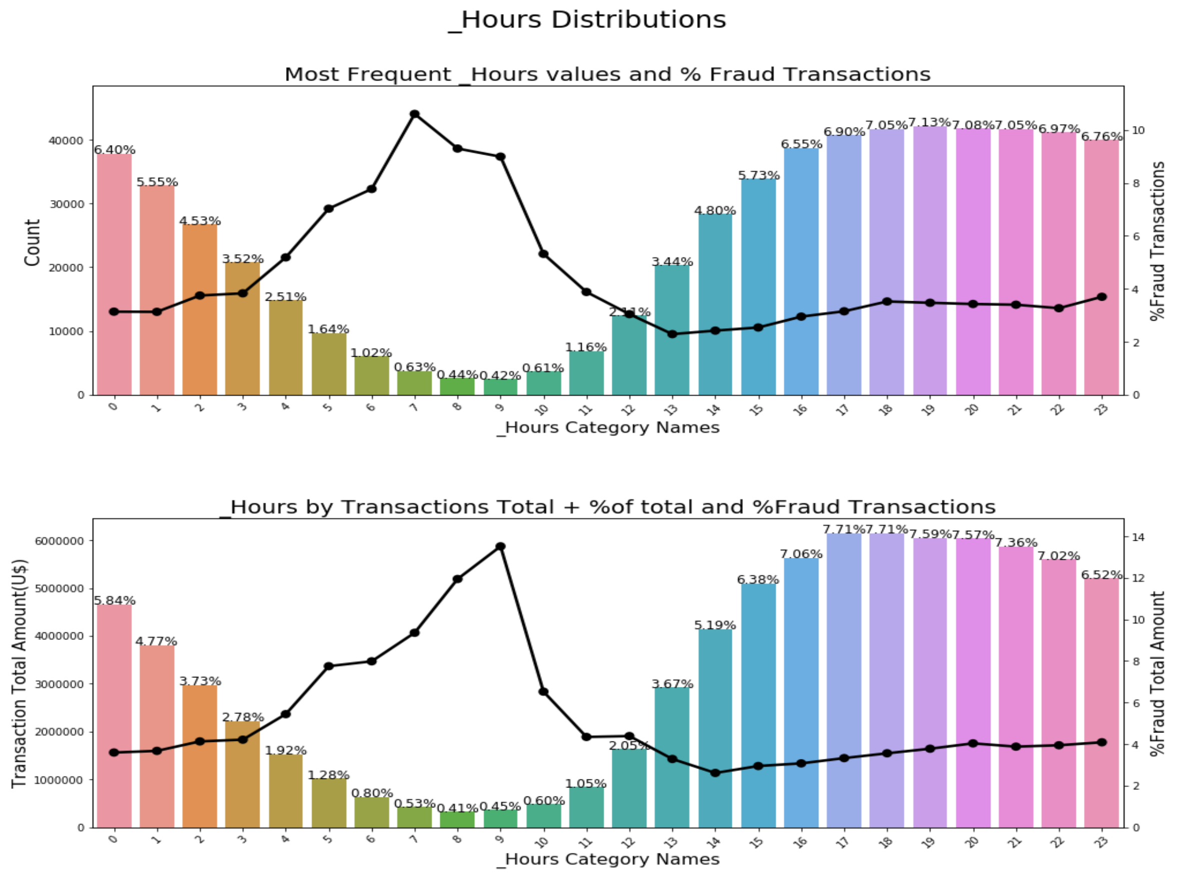


_Weekdays Distributions





Time Series Analysis of transactions



Hourly data clearly shows that there is a rising trend of transaction being fraudulent from 12 AM to 7 AM and transaction being fraudulent gradually onwards. The total transactions occurring are also low during those periods.
Time was registered according to the local time zones, so no time zone correction was essential here.

Feature Engineering/ Selection

Feature Engineering

- Variables that **did not show any significance** variance, in EDA, in detecting a fraud transaction were dropped.
- Mapped all the different email ids to different domains.
- Mapped all the different model of phones and OS under same umbrellas.
- Created **Mean** and **Standard deviation** values of Transactions amount based on card_1, card_4 address, email id (can be treated as unique ids)
- Used TimeDelta variable to pull out detail of activation date of card, latest transaction day etc.

Shrinking V elements using PCA

There are **331 V variables**, and no information about the features is provided.

Tried shrinking it with PCA to 30 variables

- Covers ~91% of the total variance in the data

Result**: Accuracy reduced sharply**

Potential Reason: Important information, useful for predicting a fraud transaction, has been lost.

- **Null valued columns have a pattern** in detecting a fraud transaction
- Some features are **highly correlated** among themselves, and contributing to overfitting

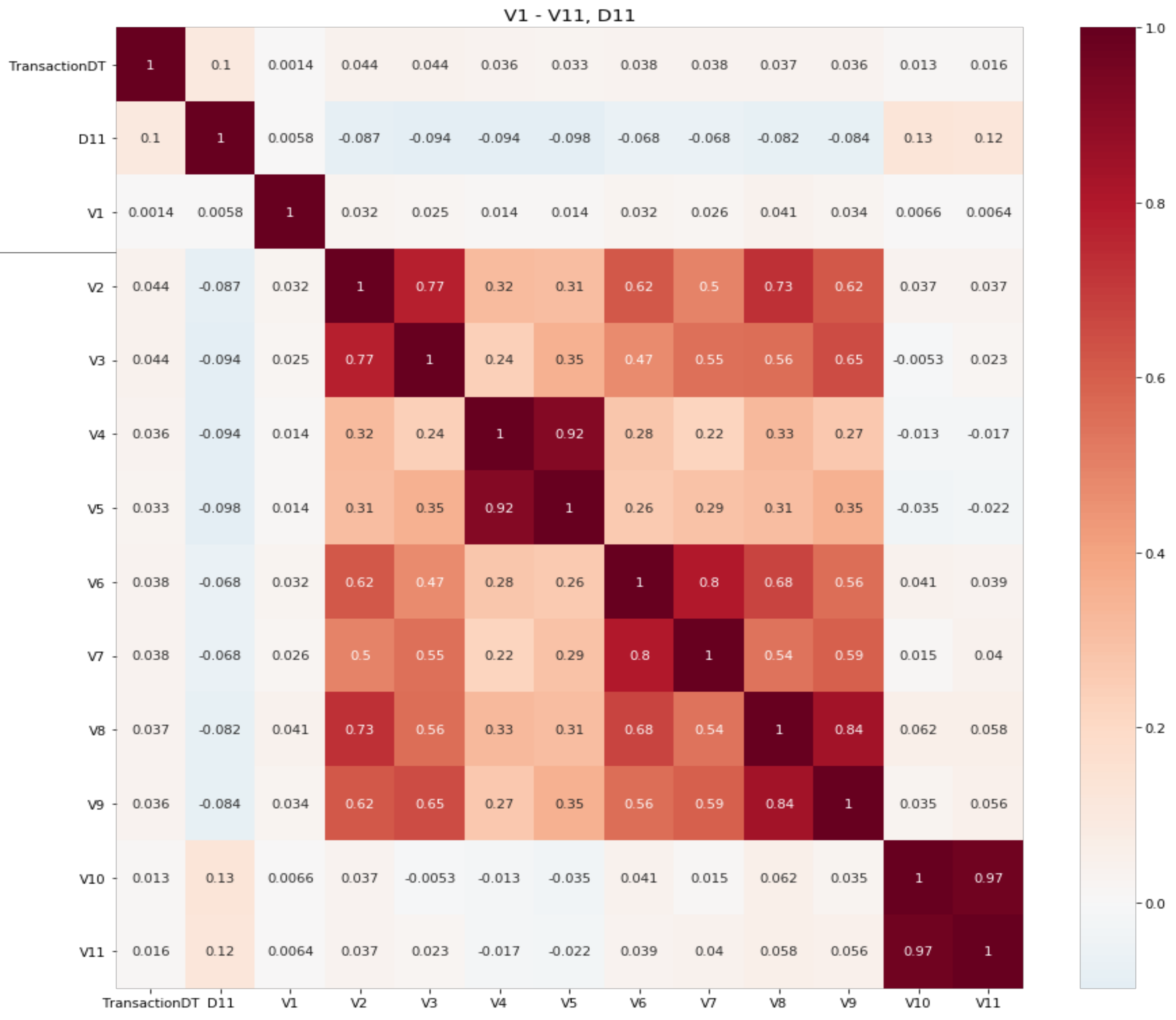
Rejection Of Columns Based On Correlations

One of the high correlation variable from significant correlation variables were dropped from the column.

For eg, in V1-V11 correlation matrix, V4 and V5 are heavily correlated, so V5 was dropped.

Similar approach was applied to all the V columns

One of the variables where correlation was **greater than 0.8 and less than -0.8 was not included** in the model.



	TransactionDT	D11	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
TransactionDT	1	0.1	0.0014	0.044	0.044	0.036	0.033	0.038	0.038	0.037	0.036	0.013	0.016
D11	0.1	1	0.0058	-0.087	-0.094	-0.094	-0.098	-0.068	-0.068	-0.082	-0.084	0.13	0.12
V1	0.0014	0.0058	1	0.032	0.025	0.014	0.014	0.032	0.026	0.041	0.034	0.0066	0.0064
V2	0.044	-0.087	0.032	1	0.77	0.32	0.31	0.62	0.5	0.73	0.62	0.037	0.037
V3	0.044	-0.094	0.025	0.77	1	0.24	0.35	0.47	0.55	0.56	0.65	-0.0053	0.023
V4	0.036	-0.094	0.014	0.32	0.24	1	0.92	0.28	0.22	0.33	0.27	-0.013	-0.017
V5	0.033	-0.098	0.014	0.31	0.35	0.92	1	0.26	0.29	0.31	0.35	-0.035	-0.022
V6	0.038	-0.068	0.032	0.62	0.47	0.28	0.26	1	0.8	0.68	0.56	0.041	0.039
V7	0.038	-0.068	0.026	0.5	0.55	0.22	0.29	0.8	1	0.54	0.59	0.015	0.04
V8	0.037	-0.082	0.041	0.73	0.56	0.33	0.31	0.68	0.54	1	0.84	0.062	0.058
V9	0.036	-0.084	0.034	0.62	0.65	0.27	0.35	0.56	0.59	0.84	1	0.035	0.056
V10	0.013	0.13	0.0066	0.037	-0.0053	-0.013	-0.035	0.041	0.015	0.062	0.035	1	0.97
V11	0.016	0.12	0.0064	0.037	0.023	-0.017	-0.022	0.039	0.04	0.058	0.056	0.97	1

NaN Columns Processing

NaN entry columns do contribute in a fraudulent transaction

Many variables with **large NAN entries**, and which did not have a pattern in the data were not included.

Variables whose NAN entries did contribute to null values were included, and filled as -1. The models were intimated that **-1 indicates a missing value** entry.

Class Imbalance

- Class imbalance for target variable was adjusted using oversampling method **SMOTE**.

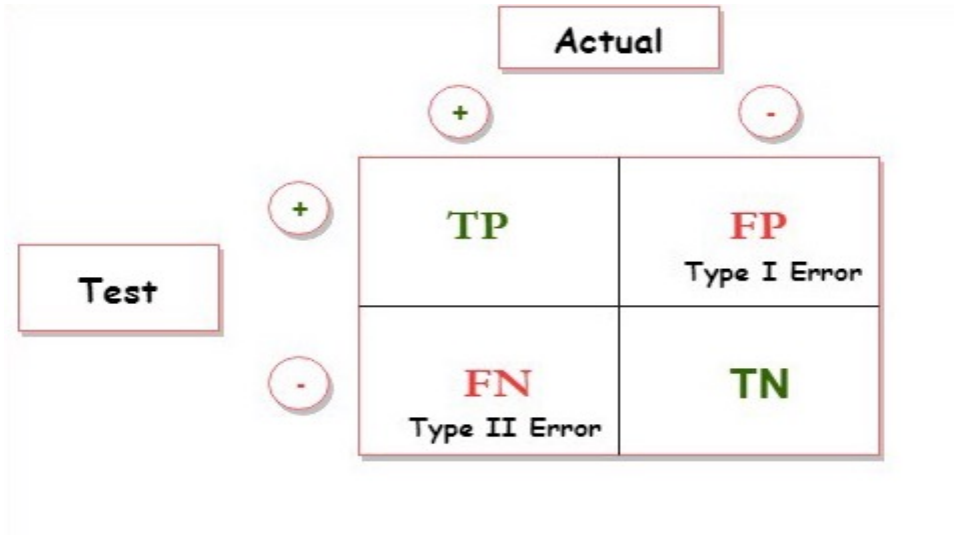
- Challenges:
  - The dataset became more **computationally expensive**.
  - The model still was **not able to make accurate classifications** between the response variables

- To tackle the challenges, we tried **under-sampling** the complete dataset and then assess
  - Model lost necessary information about non fraud transactions
  - Model accuracy went really bad

- So we **rejected** the sampling procedure approach and decided to improve model accuracies using hyperparameter tuning.

Model Evaluation

- Models under consideration: Decision Tree, Random Forest & XGBoost
- Accuracy not the correct measure for an imbalance dataset
- Precision and ROC score is the factor under consideration for measuring the performance of the models



$$\textit{Precision} = \frac{TP}{TP + FP}$$

$$\textit{Recall} = \frac{TP}{TP + FN}$$

Model Comparison

Decision Tree

vs.

Random Forest

Metric	Decision Tree	Random Forest
Test Accuracy	0.966962	0.980221
Train Accuracy	0.999990	0.996852
AUC Score	0.784625	0.731926
Confusion Matrix	[[167653 3297] [2556 3656]]	[[170770 180] [3324 2888]]
F1-Score	0.555412	0.622413
Recall	0.588538	0.464906
Precision	0.525816	0.941329

HYPER PARAMTER TUNING

The various hyper-parameters considered for model tuning for an XGB model were

- n_estimators
- learning_rate
- max_depth
- Subsample
- colsample_bytree
- Gamma
- alpha

FINAL MODEL RESULT
(HYPER PARAMETER TUNED)

96.23%

The precision we achieved using the hyperparameter tuned

XGB Classifier Model

Train Accuracy : 99.73%
Test Accuracy : 98.53%
Area under ROC Curve: 0.8024
F1 Score of 0.7434
Precision of 96.23%
Recall of 60.56%

Challenges

Holistic understanding of the system was lagging due to absence of variable feature characteristics/information

Better model accuracies

Lack of sufficient powered system

Handling large memory and dimension dataset

Future Scope

- Uniquely identifying groups/tuples based on the characteristics.
- Trying deep neural network based algorithms to assess the model more accurately and make better predictions.
- Make use of enhanced ensemble operations to further improve modelling.
- Modifications could be made to modelling approach so that it can work for streaming data, and be brought in live operations.