CAPSTONE 2 PROJECT REPORT

TABLE OF CONTENTS

1. PROBLEM STATEMENT

Imagine standing at the check-out counter at the grocery store with a long line behind you and the cashier not-so-quietly announces that your card has been declined. At this moment, you probably aren't thinking about the data science that determined your fate.

Embarrassed, and certain you have the funds to cover everything needed for an epic nacho party for 50 of your closest friends, you try your card again. Same result. As you step aside and allow the cashier to tend to the next customer, you receive a text message from your bank. "Press 1 if you really tried to spend $500 on cheddar cheese."

The problem here in my project is to detect the fraudulent credit card transactions saving millions of dollars for individuals and companies. I am predicting the probability that an online transaction is fraudulent, as denoted by the binary target "isFraud".

2. DATA WRANGLING

2.1. Data Set

The dataset used in this project is obtained from Kaggle.
https://www.kaggle.com/c/ieee-fraud-detection/data

The data is broken into two files “identity” and “transaction”, which are joined by “TransactionID”. Not all transactions have corresponding identity information.

Categorical Features - Transaction

ProductCD
card1 - card6
addr1, addr2
P_emaildomain
R_emaildomain
M1 - M9

Categorical Features - Identity

DeviceType
DeviceInfo
id_12 - id_38

The TransactionDT feature is a timedelta from a given reference datetime (not an actual timestamp).

2.2. Data Descriptions

<u>**Transaction Table**</u>

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
dist: distance
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.

Categorical Features:

ProductCD
card1 - card6
addr1, addr2
P_emaildomain
R_emaildomain
M1 - M9

Identity Table *

Variables in this table are identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions.
They're collected by Vesta’s fraud protection system and digital security partners.
(The field names are masked and pairwise dictionary will not be provided for privacy protection and contract agreement)

Categorical Features:

DeviceType
DeviceInfo
id_12 - id_38

2.3. Data Cleaning

2.3.1. Data Type Correction

Numerical columns are of type “float64” and the categorical columns are of type “object”. So I didn't have to change any data type.

2.3.2. NaN Imputation

The dataset had a lot of missing data. There are around 500 columns when “train_transaction” and “train_identity” data sets are merged. Out of these 500 columns 12 of them had more than 90% missing data. So I decided to drop these columns. After dropping some columns, if less than 5% of the column is missing then I filled the column with “0”. However for the remaining columns, I replaced the Nan’s with the mean of the column.

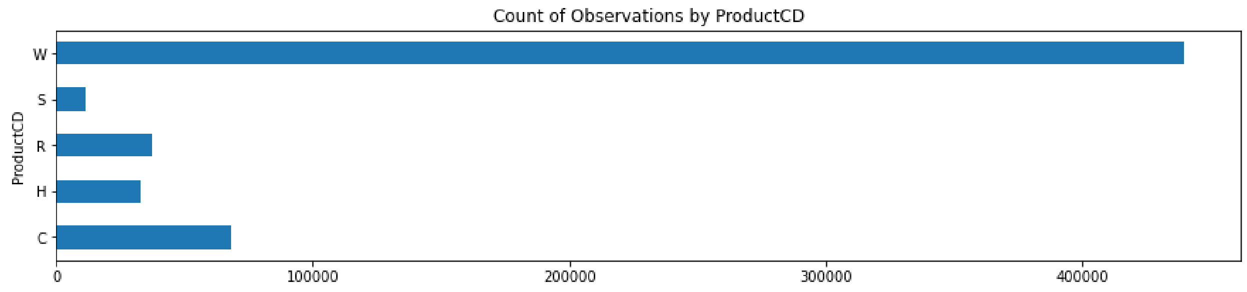
I checked if there was any duplicate rows. There was no duplicate rows.

3. EXPLORATORY DATA ANALYSIS

Since the column names were masked due to confidentiality issues, it was not easy to do some exploratory data analysis. However I still plot some graphs about the categorical columns. Below are my findings.

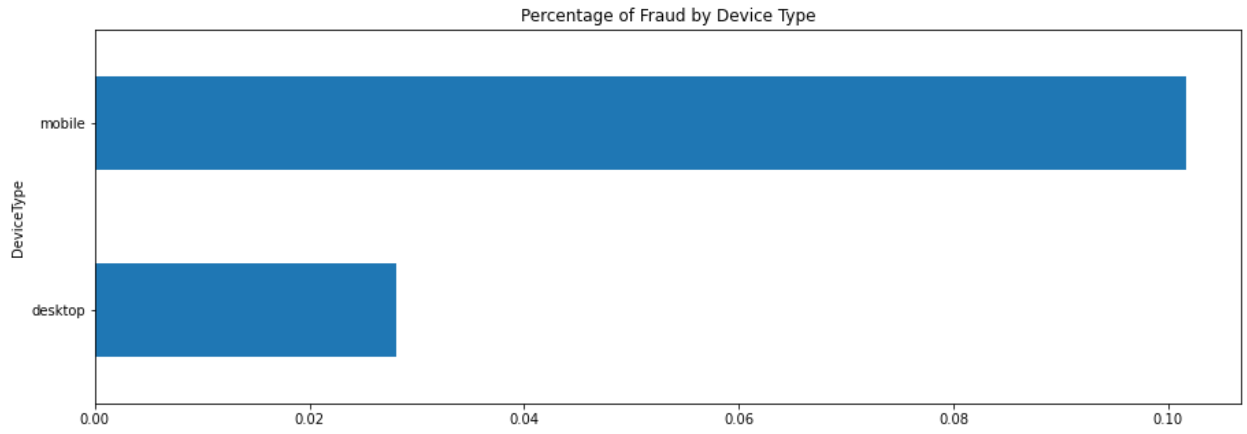
ProductCD

For now we don't know exactly what these values represent.
W has the most number of observations, C the least.
ProductCD C has the most fraud with >11%
ProductCD W has the least with ~2%



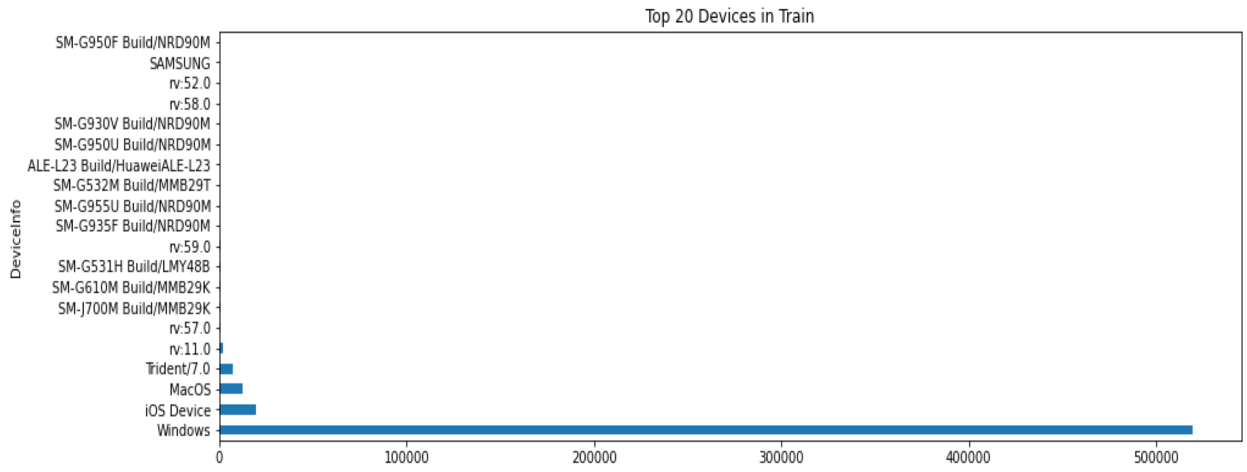
Device Type

0.10% of the mobile transactions are fraud whereas around 0.03 % are from desktop. The fraud rate in mobile transactions is 3 times more than desktop transactions.

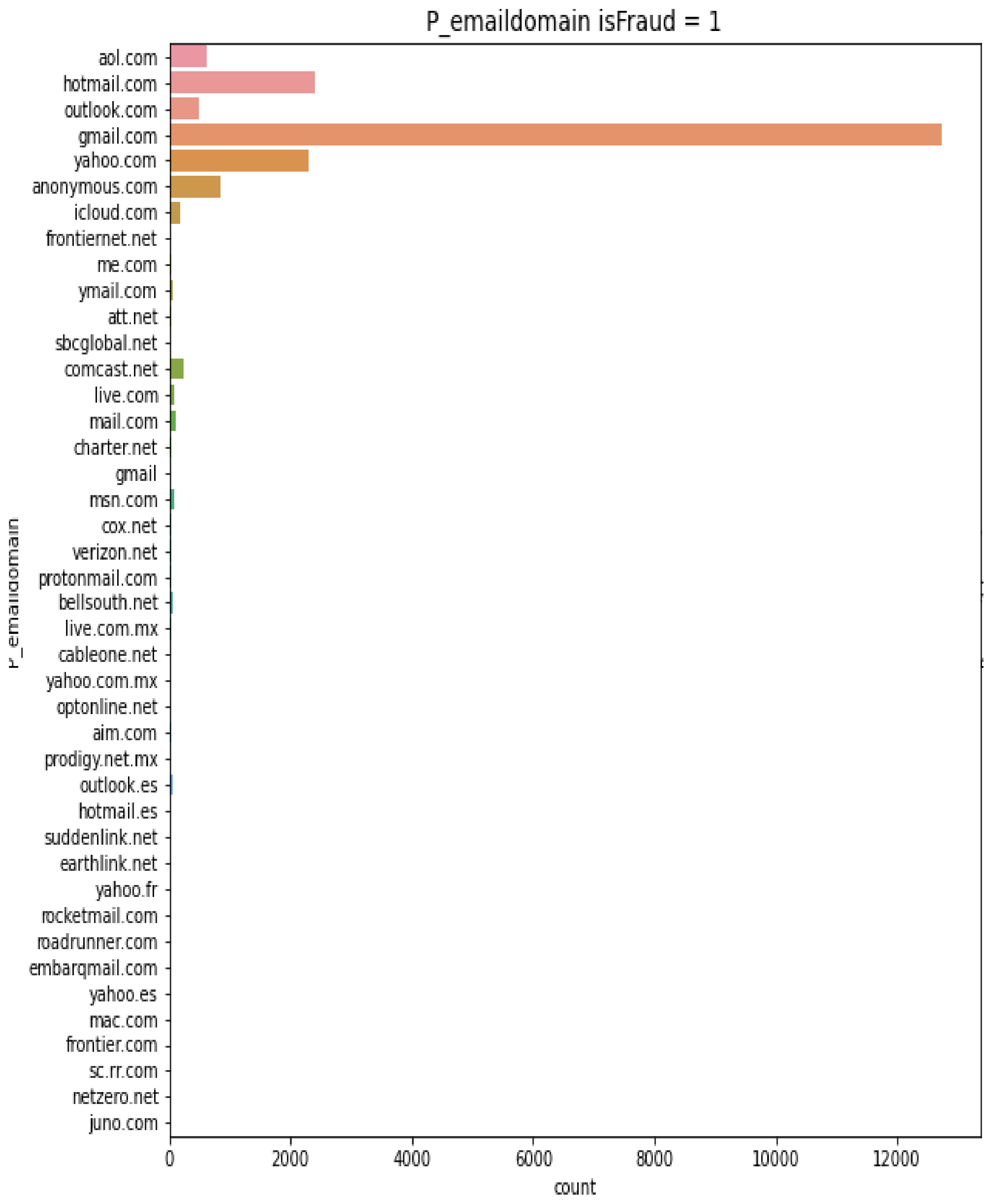


Device Info

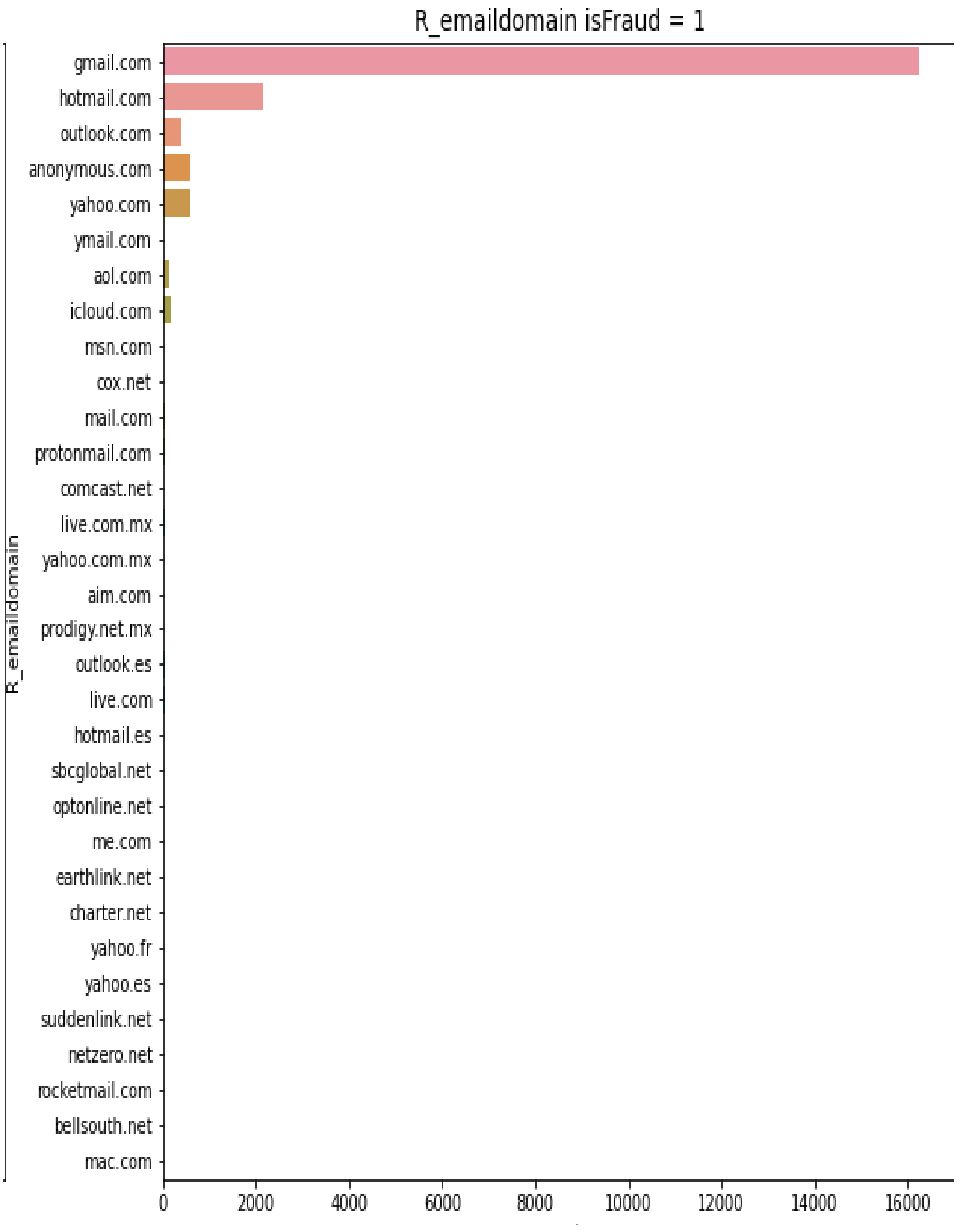
Quiet a big number of transactions are done with a device running Windows OS.



P_emaildomain



R_emaildomain



It seems like fraudsters prefer gmail.

4. PREPROCESSING

First I preprocessed the categorical columns and got their dummy features. In my data set there are 2 classes and these classes are “isFraud=0” and isFraud=1” representing non-fraud and fraud transactions. Due to the nature of Fraud transactions occurring rare compared to non-fraus ones, the classes are imbalanced. More than 99% of the transactions are non-fraud and less than 1% are fraud. So when I take accuracy as my classification metric it may be misleading. To overcome this situation I will do 2 things. First, I will oversample my data by using SMOTE and undersample by using RandomUnderSampler. I will look at Recall Score. So in this step I preprocessed the data by applying oversampling. This imputed some artificial Fraud transactions in my data.

5. MODEL SELECTION

I applied the following models . I also used different hyper-parameters by tuning them After hyper-parameter tuning, I got the following best models as follows.

Model	**Parameter**	**Precision**	**Recall**	**F1**	**Accuracy**
Logistic R. w/o over/under Sampling	default	0.78	0.30	0.44	0.97
Logistic R. with over/under Sampling	default	0.25	0.62	0.35	0.92
Random Forest	default	0.61	0.61	0.61	0.97
XGBoost	default	0.61	0.64	0.63	0.97
XGBoost	'max_depth': 9, 'min_child_weight': 1	0.65	0.70	0.67	0.98

Firstly, I have used Logistic regression without over-under sampling and default parameters. The model performed very poor. It was able to detect 30% of the Fraud transaction. Then I decided to oversampled the test data up to 10% of the train data. Then I undersampled the training data to twice the test data. I applied the logistic regression again on the new data with default parameters. This time model was able to detect 62% of the Fraud transactions but It was classifying the Non-Fraud ones as Fraud.

Next I tried Random Forest and this model was able to detect 62% of the Fraud transactions. And it was also doing fine on classifying the Non-Fraud ones also correctly.

Then I applied XGBoost with default parameters and this model did even better than Random Forest. The model detected the 63% of the Fraud transactions and 99% of the Non-Fraud ones.

Finally I did a GridSearch to tune 2 parameters (“min_child_weight” and “max_depth”). After the grid search the best parameters were as follows: 'max_depth': 9, 'min_child_weight': 1 and the model improved a lot. After the hyper-parameter tuning, the model was able to detect 70% of the Fraud transactions and 99% of the Non-Fraud transactions.

6. FUTURE WORK

Due to the computation limitation of my machine, I couldn’t try all of the parameters and a large set of parameters during hyperparameter tuning. So this work can be done with a stronger machine or in the cloud.

Also Bootstrap sampling with replacement sampling method can also be used for sampling.

Lastly, I need to use PyCaret library to build as many models as possible.