Ravelin Data Science Test

Introduction

Hi, Thanks for reading the report. This report outlines my approach to Ravelin's Fraud Detection data science test. The goal of the project was to correctly classify customer's label as "fraudulent" or not. We are given 168 customers with their orders, payment methods and transactions. As we are deciding whether a customer is a fraud or not, the project falls under the category of binary classification problem.

The main challenges of the project are three-fold:

- The given data (customer.json) is deeply nested and there is no easy nor direct way to flatten the data to dataframe.
- The task highly depends on feature engineering and how to combine informations of orders with customers.
- The dataset size is small which creates a risk of overfitting.

Approach

Exploratory Data Analysis

In this section I highlighted the things that I find interesting and useful about the data. I also write some reccommendations for preprocessing section.

There are 107 normal and 61 fraudulent customers. The data has imbalance.

2) DataFrame for Customers: 3)

```
customers.head()
```

	customerEmail	customerPhone	customerDevice	customerIPAddress	customerBillingAddress
0	josephhoward@yahoo.com	400-108-5415	yyeiaxpltf82440jnb3v	8.129.104.40	5493 Jones Islands\nBrownside, CA 51896
1	evansjeffery@yahoo.com	1-788-091-7546	r0jpm7xaeqqa3kr6mzum	219.173.211.202	356 Elizabeth Inlet Suite 120\nPort Joshuabury...
2	andersonwilliam@yahoo.com	024.420.0375	4m7h5ipl1shyavt6vv2r	67b7:3db8:67e0:3bea:b9d0:90c1:2b60:b9f0	8478 Sean Ridges Apt. 441\nDavisberg, PR 72250
3	rubenjuarez@yahoo.com	670.664.8168x94985	slovx60t0i558may4ks0	95de:8565:5a66:792c:26e0:6cfb:7d87:11af	7769 Elizabeth Bridge Apt. 343\nNortonstad, FM...
4	uchen@malone.com	1-981-877-0870	j0pd24k5h8dl2fqu0cz4	196.89.235.192	148 Russell Lodge Apt. 445\nPort Jenniferside,...

customerEmail and *customerIPAdress* need to be preprocessed as *email_domain* and *IpAddress_Type*. There are 3 duplicants with the same IPaddress.

3) DataFrame for Orders:

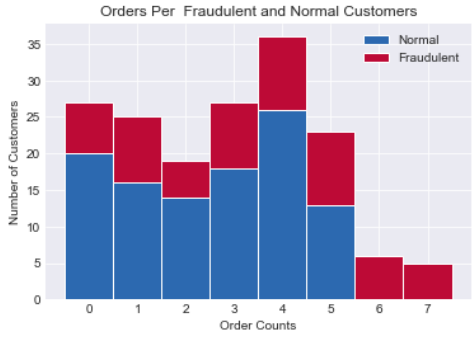
```
orders.head()
```

	orderId	orderAmount	orderState	orderShippingAddress
0	vjbdvd	18	pending	5493 Jones Islands\nBrownside, CA 51896
1	yp6x27	26	fulfilled	5493 Jones Islands\nBrownside, CA 51896
2	nlghpa	45	fulfilled	898 Henry Ports\nNew Keithview, CA 95893-2497
3	uw0eeb	23	fulfilled	356 Elizabeth Inlet Suite 120\nPort Joshuabury...
4	bn44oh	43	fulfilled	5093 Bryan Forks\nJoshuaton, FM 01565-9801

OrderId is important to match the orders with transactions. *orderState* should be engineered as categorical feature. *orderAmount* also may give some helpful information about the order and the customer.

Although there are only 168 customers there are 478 orders. Here are stacked histograms to show number of orders per each customer :

Counting Every Order



:----------:|

Fraudulent customers spread homogeneously through data.

4) DataFrame for Transactions:

```
transactions.head()
```

	transactionId	orderId	paymentMethodId	transactionAmount	transactionFailed
0	a9lcj51r	vjbdvd	wt07xm68b	18	False
1	y4wcv03i	yp6x27	wt07xm68b	26	False
2	5mi94sfw	nlghpa	41ug157xz	45	False
3	br8ba1nu	uw0eeb	41ug157xz	23	False
4	a33145ss	bn44oh	y3xp697jx	43	True

5) DataFrame for PaymentMethods:

```
payment_methods.head()
```

	paymentMethodId	paymentMethodRegistrationFailure	paymentMethodType	paymentMethodProvider	paymentMethodIssuer
0	wt07xm68b	True	card	JCB 16 digit	Citizens First Banks
1	y3xp697jx	True	bitcoin	VISA 16 digit	Solace Banks
2	6krszxc05	False	card	VISA 16 digit	Vertex Bancorp
3	5z1szj2he	False	card	Diners Club / Carte Blanche	His Majesty Bank Corp.
4	m52tx8e1s	False	card	Mastercard	Vertex Bancorp

- I have found that for all orders *transactionAmounts* are equal between transactions.
- I have found that all transactions which has same *orderShippingAdresses* belongs to the same *orderId.*

Preprocessing

After expanding all of the datasets it is time to merge them by using identical columns or creating custom features. I merged the transactions which has the same orderId into one row. For that I created a custom function to create following features:

- orderID
- transaction_count
- orderAmount
- fullfilled_order_count
- pending_order_count
- failed_order_count
- orderShippingAddress
- paymentMethod_count
- transactionFailed_count
- paymentMethodProvider_count
- paymentMethodType_apple
- paymentMethodType_btc
- paymentMethodType_card
- paymentMethodType_paypal

I created a dataframe which has 478 rows × 14 columns.

```
group_list.head()
```

	orderId	transaction_count	orderAmount	fullfilled_order_count	pending_order_count	failed_order_count	orderShippingAddress	paymentMethod_count	trans
0	hoxg2j	1	31	1	0	0	0008 Jessica Stream\nMcbridetown, MH 04017	1	
1	deh1ao	2	42	2	0	0	34944 Glenn Mountain\nSherryhaven, TN 36252	1	
2	sd4v17	1	58	1	0	0	2182 Samantha Gardens\nElliottview, MP 18669	1	
3	pc6hb6	1	46	1	0	0	687 Rogers Bridge Suite 780\nValdezburgh, IN 2...	1	
4	2528x8	1	32	1	0	0	148 Russell Lodge Apt. 445\nPort Jenniferside,...	1	

Then again I created a custom merging function to group orders and add to each customer. Here are the features that I created from them:

- customerEmail
- customerPhone
- customerDevice
- customerIPAddress
- order_counts
- transaction_counts
- orderAmount_sum
- orderAmount_mean
- fulfilled_order_counts
- pending_order_counts
- failed_order_counts
- shipping_adresses_count
- shipping_adress_same
- paymentMethod_counts
- paymentMethodProvider_counts
- paymentMethodType_apples
- paymentMethodType_btcs
- ,paymentMethodType_cards
- paymentMethodType_paypals

These features were mostly sum of the features that I extracted before. Thats why most of them as 's on their name. For example it was *fullfilled_order_count* and I sum them and created *fullfilled_order_counts*

Before starting the training I changed customerEmail to customerDomain, customerPhone to customerPhoneType, customerIPAddress to customerIPAddressType and deleted the customerDevice

```
customer_list['customerEmail'].value_counts()
```

```
gmail          31
yahoo          30
hotmail        24
wright          3
clark           2
               ..
pena            1
stokes          1
dunn            1
russo           1
jones-lloyd     1
Name: customerEmail, Length: 80, dtype: int64
```

Changed the *customerEmail* to **other** if it is not one of the main 3 provider (*gmail* , *yahoo*, or *hotmail*)

Dropped *CustomerPhone* and *customerDevice* columns

Changed the *CustomerIPAddress* to two different categories as **digits** and **digits_and_letters**

```
customer_list['customerIPAddress'].value_counts()
```

```
digits_and_letters    91
digits                77
Name: customerIPAddress, dtype: int64
```

This shows how many orders have one transaction. Added as *order_transaction_same* column

```
(customer_list['order_counts'] == customer_list['transaction_counts']).value_counts()
```

```
True     89
False    79
dtype: int64
```

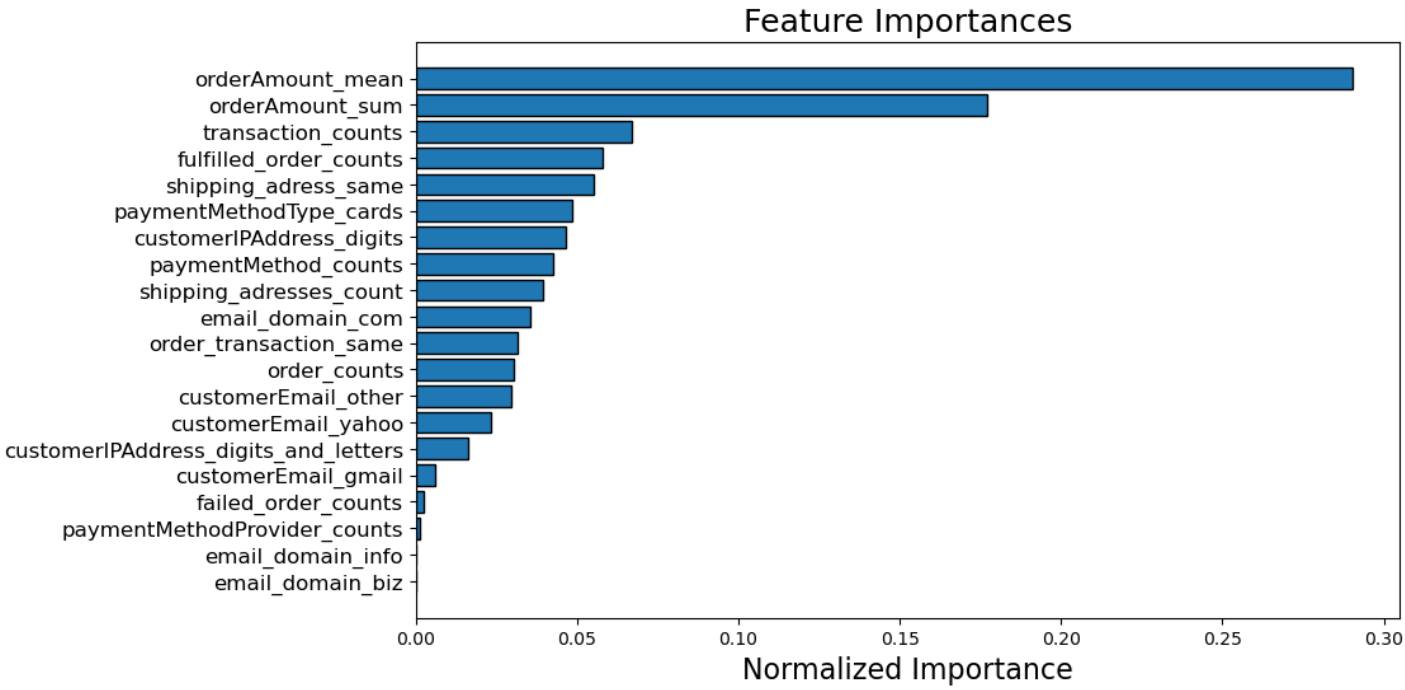
In the end of our preprocessing section we have a dataset of 168 rows x 28 columns including *fraudulent* column.

```
customer_list.head()
```

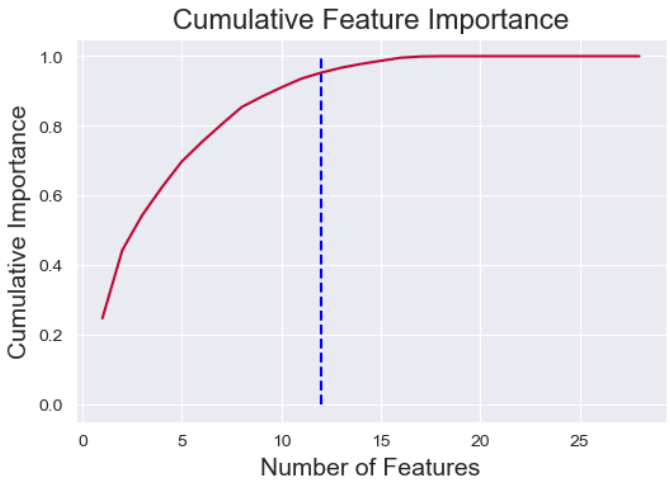
	customerEmail	customerIPAddress	order_counts	transaction_counts	orderAmount_sum	orderAmount_mean	fulfilled_order_counts	pending_order_counts
0	yahoo	digits_and_letters	2	2	44	22.000000	1	1
1	yahoo	digits_and_letters	3	3	111	37.000000	3	0
2	yahoo	digits	3	5	82	27.333333	4	0
3	yahoo	digits	3	3	85	28.333333	3	0
4	other	digits_and_letters	7	7	411	58.714286	6	0

Main Approach

I split the train and test set using sklearn's *train_test_split* method by stratify the dataset equally. The test size is %20 of all data. I ran one of my custom feature importance scripts. Internally it uses an lightgbm to calculate importance of each feature and then I plotted:



Very interestingly the least column I suspect was orderAmounts but they turned to be the most important feature. Here is the cumulative importance of my features:



15 features required for 0.95 of cumulative importance.

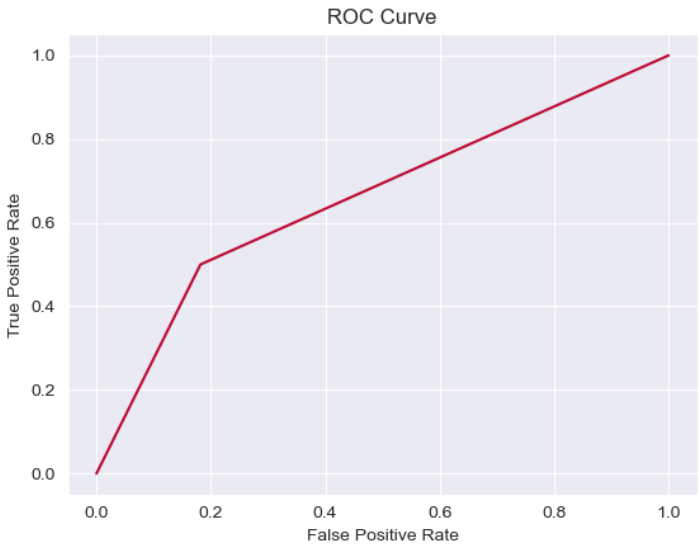
For training I tried Support Vector Classifier and RandomForestClassifier, I didnt want to work with Deep Learning since we don't have enough data to train a better neural network than these relatively simpler algorithms.

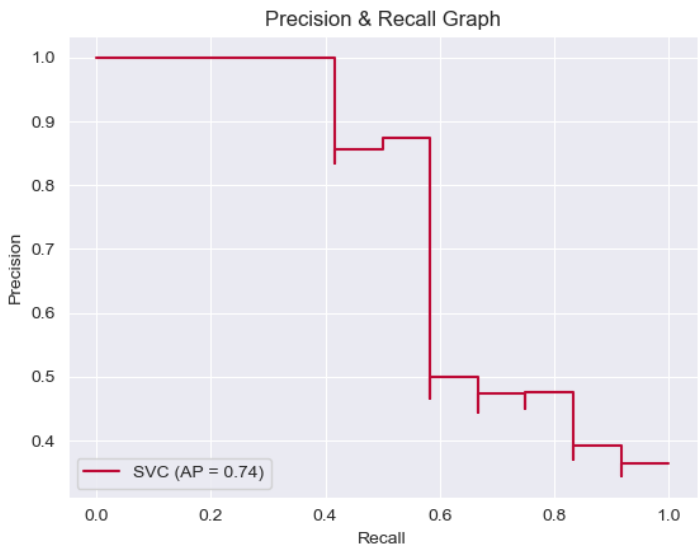
Results and Discussion

The accuracy got reached to %79 with SVC() and %73 with Random Forest Classifier. But in systems such as fraud detection the accuracy metric is not enough to measure model's performance. Here is the confusion matrix of the model:

True Positive: 22	**False Negative: 0**
False Positive: 7	True Negative: 5

The AUC score is 0.70





Conclusion and Future Work

I followed the rules of the test and get things done as soon as possible to create an end-to-end solution. All of the sections: EDA, Preprocessing and Training happened in one iteration so I would definetly go back and play with the feature engineering a bit more after I learned orderAmounts are really important. Then in preprocessing part I can come up with ideas about the columns that I had to drop for example *customerPhone* and *customerDevice*. In the training part, I would try to balance the data by using upsampling techniques so I can have more data for the training. I would use gridSearch to tune hyperparameters of my learning algorithm.