Algoritmos de Clasificación para Autorizar Transacciones de Tarjetas de Crédito, Débito y Prepagas

EMMANUEL ZABALA CHIARADIA

Universidad Torcuato Di Tella

Master en Management + Analytics

2020

Resumen

Cuando llega el momento de decidir si una transacción será rechazada o no, lo más importante es la precisión. Los falsos rechazos pueden afectar la relación con los tarjetahabientes, por lo tanto, evaluar múltiples factores, incluyendo información del consumidor, comercio y emisor a través de la experiencia de compra puede mejorar la experiencia del tarjetahabiente y aprobar más transacciones genuinas, sin incrementar el riesgo. Los modelos de machine learning son capaces de detectar anomalías en las compras basándose en diferentes fuentes de datos para un consumidor en particular. Esto incluye utilizar información de riesgo, datos de geolocalización de la transacción, datos del comercio, del dispositivo desde el cual se realiza la transacción, la hora del día, y el tipo de compra. El algoritmo también aprovecha los segmentos originados por datos del valor del consumidor, o los agrupamientos de consumidores en diferentes niveles basado en el valor potencial que ellos tienen para la compañía en el futuro. Los rechazos en autorizaciones de transacciones causados por los fraudes con tarjetas y los falsos rechazos cuestan a los consumidores y a las compañías financiera billones de dólares por año. Los sistemas de autorizaciones de transacciones se han vuelto fundamentales para los bancos y otras entidades financieras que buscan minimizar sus pérdidas. En este paper, se utilizarán varias técnicas de machine learning para clasificación como la Regresión Logística (LR), Árboles de Decisión (DT), Random Forest (RF) y XGBoost (XGB). Después de haber efectuado varias pruebas y comparaciones, elegimos a los árboles de decisión como el mejor clasificador para construir nuestro modelo de autorización de transacciones. La evaluación

de su performance fue realizada sobre un dataset real que contenía transacciones de tarjetas de crédito, débito y prepagas para demostrar los beneficios de un algoritmo de árbol para resolver este problema de una manera rápida y precisa, minimizando costos. Utilizando KPIs específicos se compara la performance entre los modelos para decidir cuál es el de mayor precisión al momento de clasificar si una transacción deberá ser aprobada o rechazada.

Credit, Debit and Prepaid Card Transaction Authorization Algorithms

EMMANUEL ZABALA CHIARADIA

Universidad Torcuato Di Tella

Master's in Management + Analytics

2020

Abstract

When a decision to decline a transaction is going to be made, accuracy is what matters the most. False declines can damage the relationship with cardholders, so evaluating multiple factors, including information about the consumer, merchant, and issuer, throughout the shopping experience can enhance that cardholder experience and approve more genuine transactions, without increasing risk. Machine learning models seem able to detect abnormal shopping behaviors based on diverse data sources within a single account. These include risk assessment data, geolocational data, merchant information, device data, time of day, and the type of purchase. The algorithm also leverages customer value segmentation, or the grouping of customers into tiers based on the amount of potential value they offer the company in the future. Declines on transactions authorizations caused by card frauds and false declines costs consumers and financial companies billions of dollars annually. Transaction authorizations systems have become essential for banks and financial institutions to minimize their losses. In this paper, various machine learning techniques were used for classification such as Logistic Regression (LR), Decision Trees (DT), Random Forest (RF) and XGBoost (XGB). After several trials and comparisons, we introduced Decision Trees, as the best classifier to construct the transaction authorization model. The performance evaluation is done on real life credit, debit, and prepaid cards transactions dataset to demonstrate the benefit of a tree algorithm on solving this problem in a fast and accurate way minimizing costs. Performance was compared using defined KPI's to decide which method was the most accurate predicting whether a transaction will be approved or declined.

Contents

1. PROBLEM INTRODUCTION

We are on the express process to begin a cashless society and credit card brands are the main enablers to make this happen. While this can be received as good news, on the other hand, fraudulent transactions are on the rise as well. For example, when authorizing transactions, failing to detect fraudulent ones may have an economic impact from a few to thousands of dollars, depending on the particular transaction and card holder. As we can see on Figure 1, non-cash transactions are being forecasted on the rise and, in current pandemic scenario, most of them are card not present transactions (ecommerce). Even with EMV[1] smart chips and contactless technologies being implemented, there are still high amount of money lost from credit card fraud and false declines. Mentioned technologies drive a reduction on fraud losses for card present[2] transactions but the EMV implementation is expected to lead on increase in card not present fraud. Card not present[3] fraud includes telephone, internet, and mail order transactions in which the cardholder does not physically present the card to the merchant. According to a 2017 report by the US Payments Forum, the increased security of chip cards forced criminals to shift the focus of their activities to card not present transactions.

Credit card fraud happens basically in two types: application fraud and transaction fraud. Application fraud is like identity fraud that one person uses another person's personal data to obtain a new card. Transaction fraud happens when a card is stolen, or a lost card is obtained to conduct fraudulent transactions. A fraudster will try to abuse the card as much as possible in a short period of time before the card is detected and suspended. So, the target will be to spot abnormal transactions made in a short period of time. With this goal, by aggregating transactions over a period, it will be possible to discover abrupt changes.

[1]EMV: developed and managed by American Express, Discover, JCB, Mastercard, UnionPay, and Visa — is a global standard for credit cards that uses computer chips to authenticate (and secure) chip-card transactions.
[2]Card present: are those transactions in which a credit card is physically present.
[3]Card not present: is a payment card transaction made where the cardholder does not or cannot physically present the card for a merchant's visual examination at the time that an order is given, and payment effected. It is most used for payments made over Internet, but also mail-order transactions by mail or fax, or over the telephone.

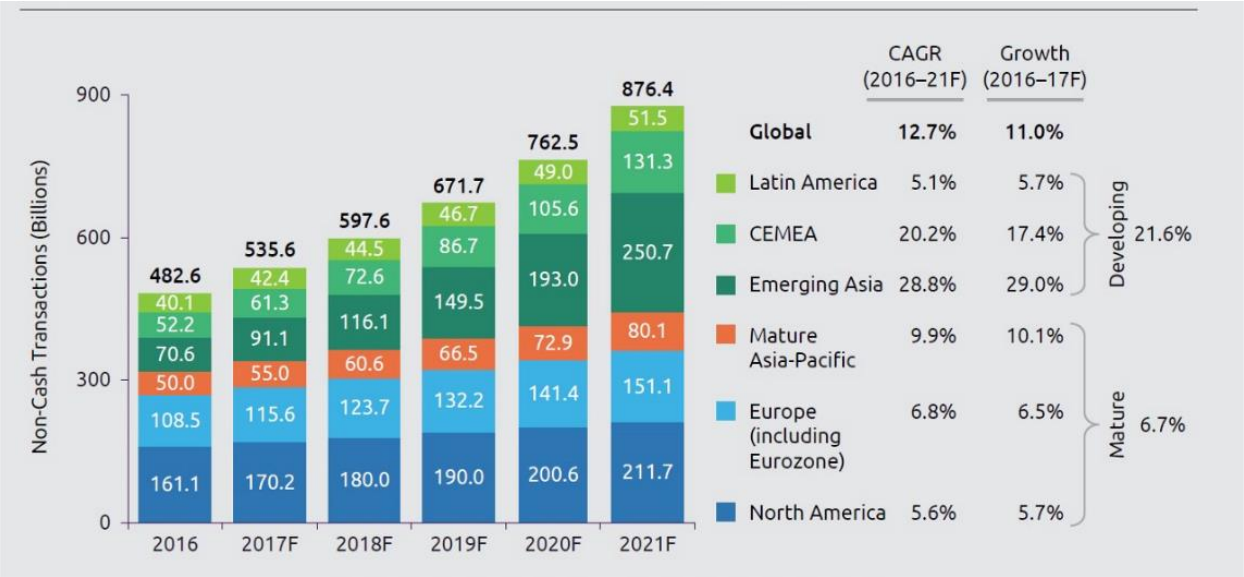


Figure 1. Number of worldwide non-cash transactions (billions) by region, 2016-2021F. Source: World Payments Report 2017, Capgemini.

Currently, over one third of consumers stop shopping at certain retailers after being falsely declined and 20% of cardholders switch their banks after experiencing a fraud. The implementation of a credit card fraud detection algorithm will save money to both merchants and issuers improving consumer's satisfaction, enhancing payment experience, and reducing false declines rates among transactions. The average monthly amount spent on the card after two or more false-positive denials declines by 15 percent on average in the six months after the false denials.

The transaction score or probability defined by the machine learning algorithm will help issuers to decline fraudulent transactions in real time and prevent false declines of legitimate transactions. The false declines (false positives) are also an important issue to be addressed driving consumer complaints with card issuer and sometimes making them change from one bank to another. When a transaction receives a score that is below the client´s designated threshold defined by the issuer, the card will be declined. This model intakes data from credit, debit and prepaid card transactions and then determines the likelihood that it is declined or approved based on correlations created between features from historical data for a certain customer profile. Analyzing historical data on

consumer shopping behavior, abnormalities will be detected to identify a possible fraud that must be classified as a declined transaction.

The growth of the Internet of Things (IoT) means that payment systems will have to handle an increasing number of automated transactions. This means AI routines will have to get stronger and faster to cope with the demand and increasingly complex use cases. The future world is getting more and more complicated with a fridge making transactions, and a car driving itself to the charging station and making a transaction there. These are all going to be autonomous transactions, all the data that is going to come out of these transactions will be extremely useful in helping the company's decisioning and also helping consumers manage their day-to-day lives better.

2. OBJECTIVE

The main motivation of this paper is to find reliable classifications models using machine learning that will be able to authorize legitimate transactions and identify fraudulent or suspicious ones accurately and decline them. The decisions of whether to decline or approve a transaction are based on a constantly flowing stream of data, and self-teaching algorithms, rather than a static sample dataset and fixed rules. This new functionality can help improve the accuracy of real-time approvals of genuine transactions and reduce false declines.

3. ROADMAP

The following steps were followed to apply machine learning algorithms on our dataset to solve the detailed problem:

1. State the question and determine required data
2. Acquire the data in an accessible format
3. Identify and correct missing data points/anomalies as required

4. Prepare the data for the machine learning model
5. Establish a baseline model aim to exceed
6. Train the model on the training data
7. Make predictions on the test data
8. Compare predictions to the known test set targets and calculate performance metrics
9. If performance is not satisfactory, adjust the model, acquire more data, or try a different modeling technique
10. Interpret model and report results visually and numerically

4. DATA ACQUISITION

The data will be collected from a credit card company data warehouse. The main objective of having three different products being analyzed is to also identify if consumer behaviors differ from credit or debit to prepaid cardholders being the last ones more prone to have declines since prepaid cards needs to have balance available in order to be used. The timeframe of this database will be 6 months to gather a full profile of different cardholders. A random sample of the full database will be taken to minimize computer processing costs. Each row of this database has the data and characteristics of 1 transaction made on a merchant by a single cardholder, so the size ("n") of the database will be considerable. This will be multiplied by the amount of transactions that each card had over the past 6 months, considering the fact that currently the total amount of credit, debit and prepaid cards issued by the company that owns the database is almost 3.5MM. Also, each row has several covariates that will be used to improve the model prediction accuracy, most relevant of them are listed on appendix.

Predict variable (desired target)

- Authorization Response: Auth_Resp (categorical: "approved", "declined")

- Authorization Response Code: Auth_Resp_Action_Code (numeric: 1= “approve”, 0= “decline”)

Txn_Time	Resp_Cd	Auth_Resp	Card_Pres_Cd	Merch_ID	Merch_Name	Merch_City	Merch_Country
45021	0	Approved or completed successfully	1	N1WYWUNP6AAYPPJ	NEURONUP.COM	BARCELONA	ESP
50537	0	Approved or completed successfully	1	N1WYWUNP6AAYPPJ	NEURONUP.COM	BARCELONA	ESP
83350	0	Approved or completed successfully	1	12289894	QR	CIUDAD AUTONOMA BUENOS AIRES	ARG
83706	0	Approved or completed successfully	1	12289894	RUBENJUVESRL	CIUDAD AUTONOMA BUENOS AIRES	ARG

Figure 2. Raw dataset sample.

5. EXPLORATORY DATA ANALYSIS (EDA)

First, we get the numerical features statistics of the dataset focusing on two main covariates. The full 6-month dataset has 68.9MM rows, each of those represents a single transaction made by a cardholder. A random sample dataset will be used in this paper to optimize computational resources available. It contains 516,271 transactions and 17 covariates. The mean value of all transactions is $32.46, 75% of them are below $20.70 threshold. The biggest transaction recorded in this sample dataset amount to $25,922.

Also, we can observe the median value of the feature Txn_Time that is transaction time being at 14:41 or 2:41 p.m.

```
dff.describe()
```

	Txn_Cnt	Txn_Time	Resp_Cd	Card_Pres_Cd	Txn_Amount	Auth_Resp_Action_Code
count	516271.0	516271.000000	516271.000000	516271.0	516271.000000	516271.000000
mean	1.0	136737.514817	21.609839	1.0	32.462346	0.529023
std	0.0	63910.028939	26.977212	0.0	161.230012	0.499157
min	1.0	0.000000	0.000000	1.0	0.000000	0.000000
25%	1.0	101506.000000	0.000000	1.0	2.070000	0.000000
50%	1.0	144121.000000	5.000000	1.0	5.510000	1.000000
75%	1.0	185955.000000	51.000000	1.0	20.700000	1.000000
max	1.0	235959.000000	91.000000	1.0	25922.520000	1.000000

Figure 3. Numeric features statistics.

Then, we search for missing or null values on our dataset and proceed to exclude those rows from our dataset since there were a few of them, only 0.01% of dataset.

Class (Approve/Decline)

Note how balanced is our dataset, we have 47.1% of declined transactions and 52.9% of approved ones. This result was not expected prior to begin this analysis; this means that almost 47 of 100 transactions are being declined in the online commerce. Being already aware of the consequences of false positives on customers this is a powerful insight to consider.

```
Decline 47.1 % of the dataset
Approve 52.9 % of the dataset
```

Figure 4. Dataset dependent variable balance.

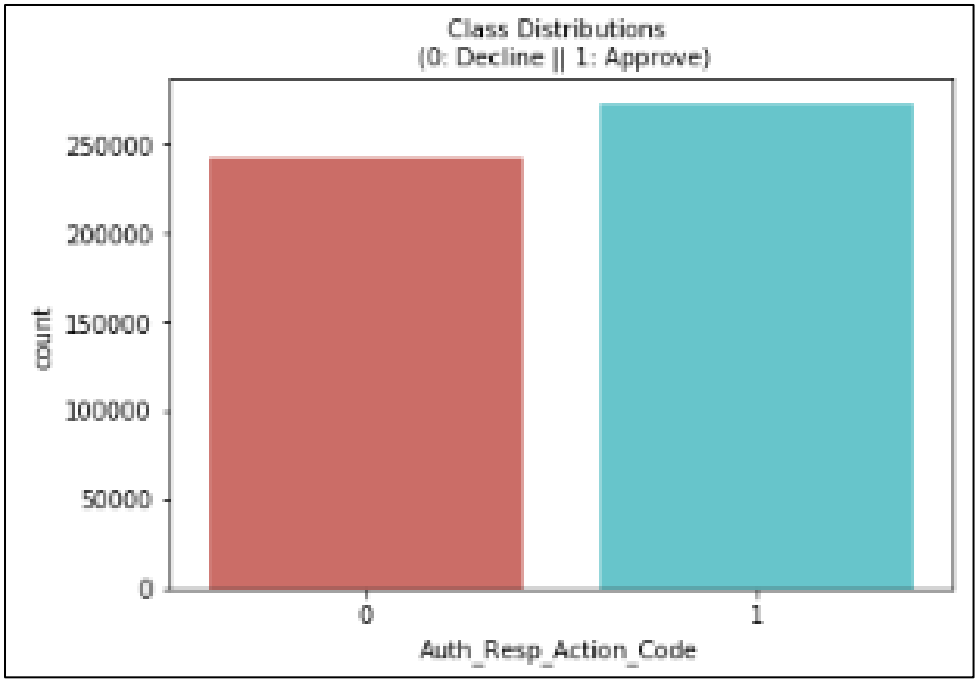


Figure 5. Dataset dependent variable class distribution.

```
dff.groupby('Auth_Resp_Action_Code').mean()
```

Auth_Resp_Action_Code	Txn_Cnt	Txn_Time	Resp_Cd	Card_Pres_Cd	Txn_Amount
0	1.0	136030.730745	38.887252	1.0	33.789843
1	1.0	137366.749549	6.228128	1.0	31.280504

Figure 6. Dependent variable statistics.

Observations:

- The average transaction amount for declined transactions is almost $2.5 higher than approved transactions.
- Even though we are considering time as a continue numeric variable, we can see that decline and approve transactions are on average occurring at the same time of the day.

Also, we can calculate categorical means for other variables such as product to get more detailed sense of the dataset.

```
dff.groupby('Prod_Name').mean()
```

Prod_Name	Txn_Cnt	Txn_Time	Resp_Cd	Card_Pres_Cd	Txn_Amount	Auth_Resp_Action_Code
BLACK	1.0	138602.715164	6.362725	1.0	71.445491	0.810271
CORPORATE	1.0	129601.710084	12.988445	1.0	93.552878	0.537815
DEBIT BLACK	1.0	149816.204819	3.662651	1.0	85.963373	0.614458
DEBIT PLATINUM	1.0	143789.225166	5.238411	1.0	29.261987	0.675497
DEBIT STANDARD	1.0	143233.932414	7.020776	1.0	32.537273	0.728031
GOLD	1.0	138373.557192	11.282744	1.0	51.404516	0.709552
PLATINUM	1.0	139596.130543	8.780957	1.0	57.695073	0.757847
PREPAID	1.0	135433.318468	33.705964	1.0	12.386195	0.348474
PRODUCT NOT CLASSIFIED	1.0	123358.500000	14.000000	1.0	1.252500	0.000000
STANDARD	1.0	136153.660701	14.698666	1.0	44.900502	0.614450

Figure 7. Numeric variables statistics by product name

Observations:

- The highest average transaction amount, $93.55 belongs to Corporate sub-product
- The lowest average transaction amount, $12.38 belongs to Prepaid product
- Product not classified will be ignored on following analysis

Transaction Amount

We can observe the distribution of transaction amounts. As we can see in Figure 8, most transactions are relatively small and only a tiny fraction of transactions comes even close to the maximum. The distribution of the monetary value of all transactions is heavily right-skewed. Most daily transactions are not extremely expensive (most are <$50), but it is likely where most fraudulent transactions are occurring as well.

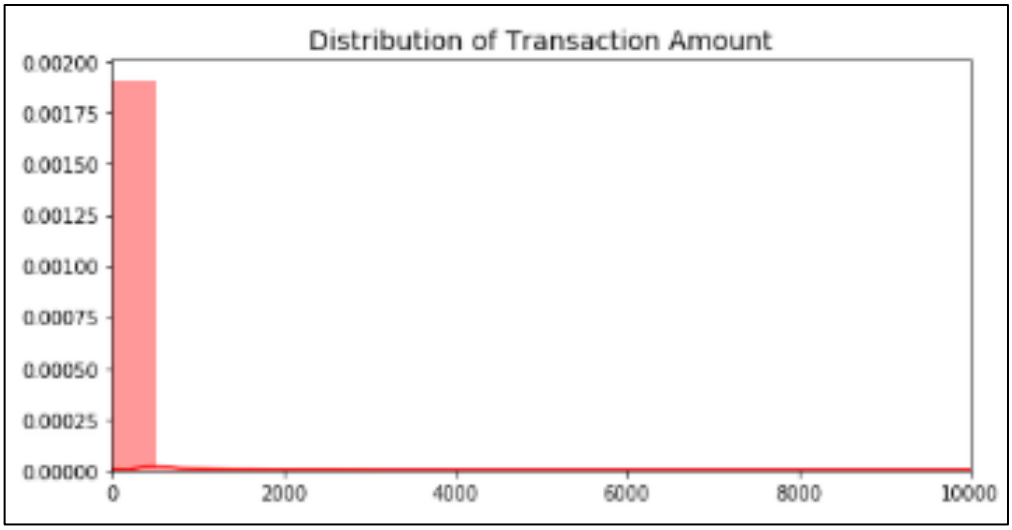


Figure 8. Transaction amount distribution.

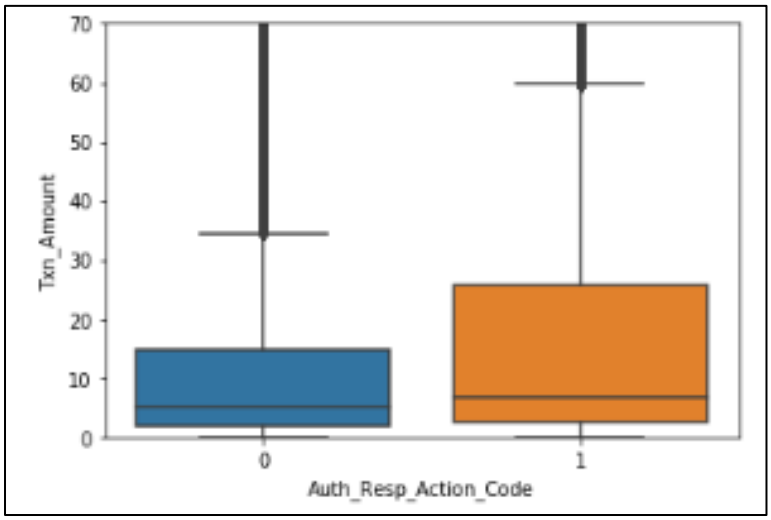


Figure 9. Transaction Amount boxplot (y-axis trimmed)

Transaction Time

Also, we can observe the distribution of transaction times and it's division between approvals and declines on following figures. As we can see on Figure 10, it is reasonable to assume that the drop in volume occurred during the night. Most purchases are made

during the daylight hours, and as people get out of work, school or university and head home, purchasing dwindles down until the next day.

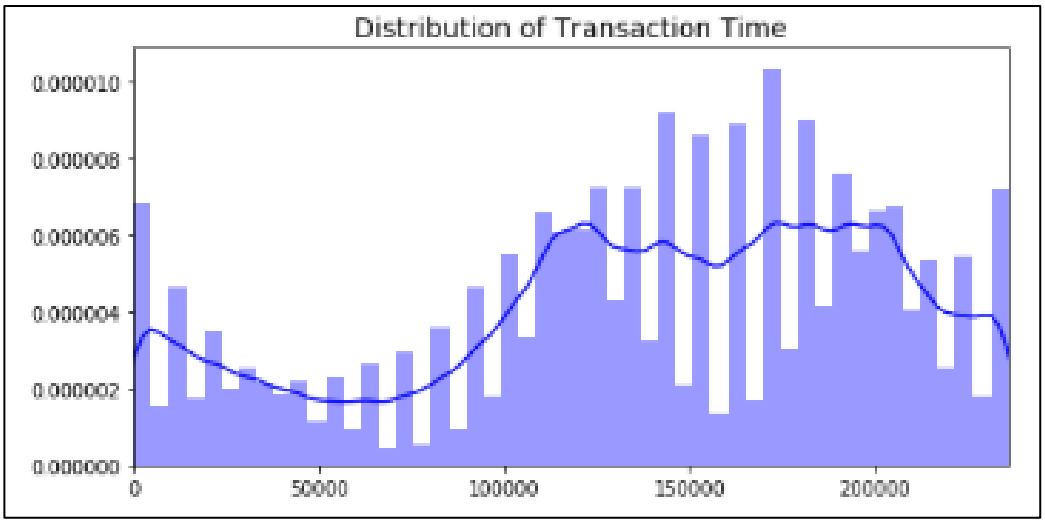


Figure 10. Transaction time distribution

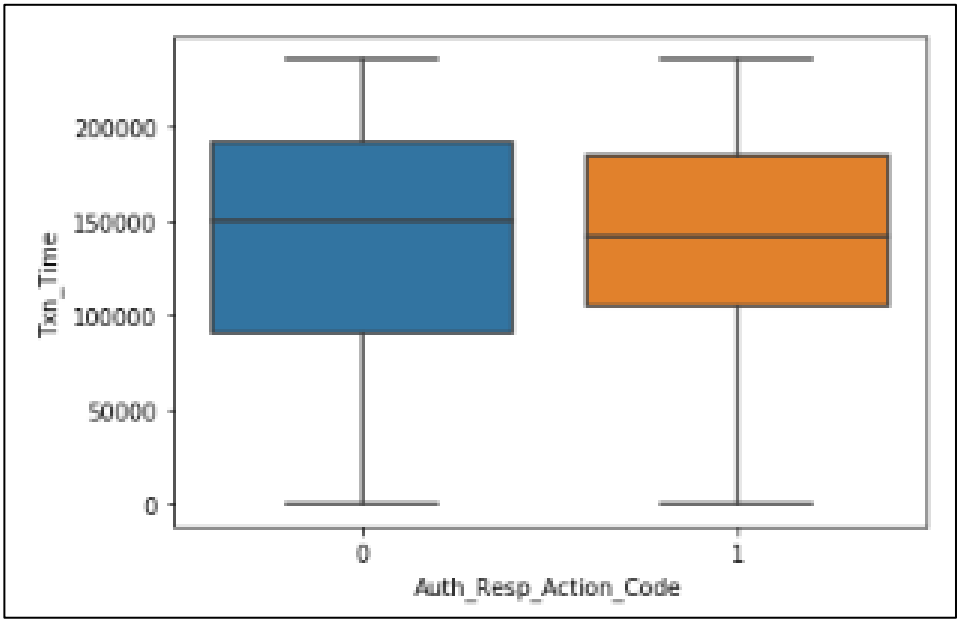


Figure 11. Transaction Time boxplot

Authorization Response Code (0=decline, 1=approve)

We can observe the authorization response code (dependent variable) frequency by product.

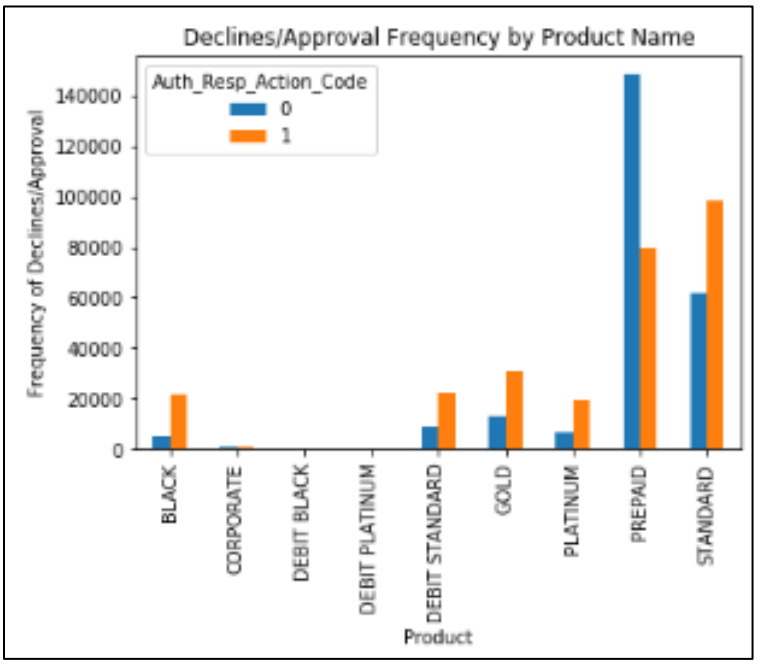


Figure 12. Decline/Approval frequency by product name.

As we can see, most part of the declined transactions are being generated by prepaid cardholders. This is an expected result since prepaid cards need to be charged and have an available balance to make a purchase. Sometimes, due to the consumer behavior of this product, there is not enough money on the card available to cover even a low-ticket purchase. Product name seems to be a good predictor of the outcome variable looking at the stacked bar chart on Figure 13.

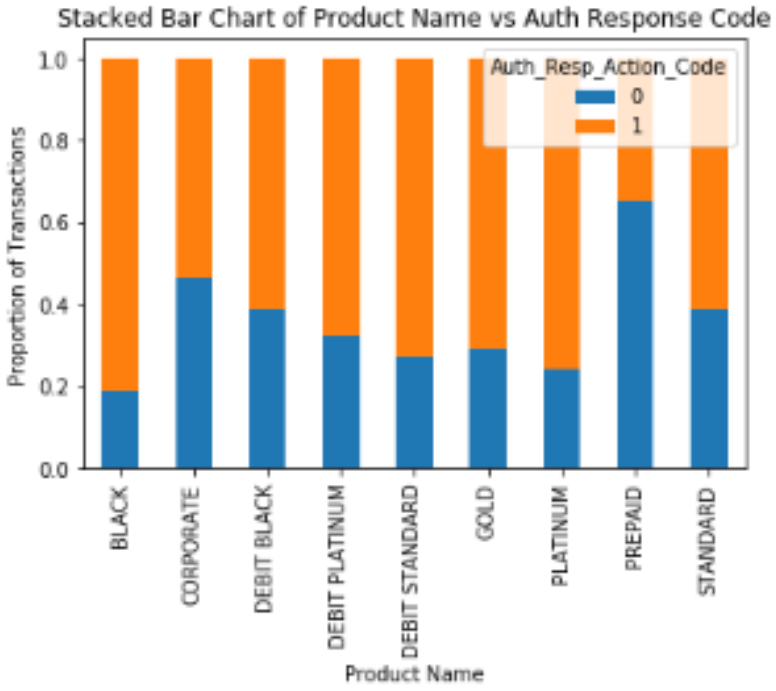
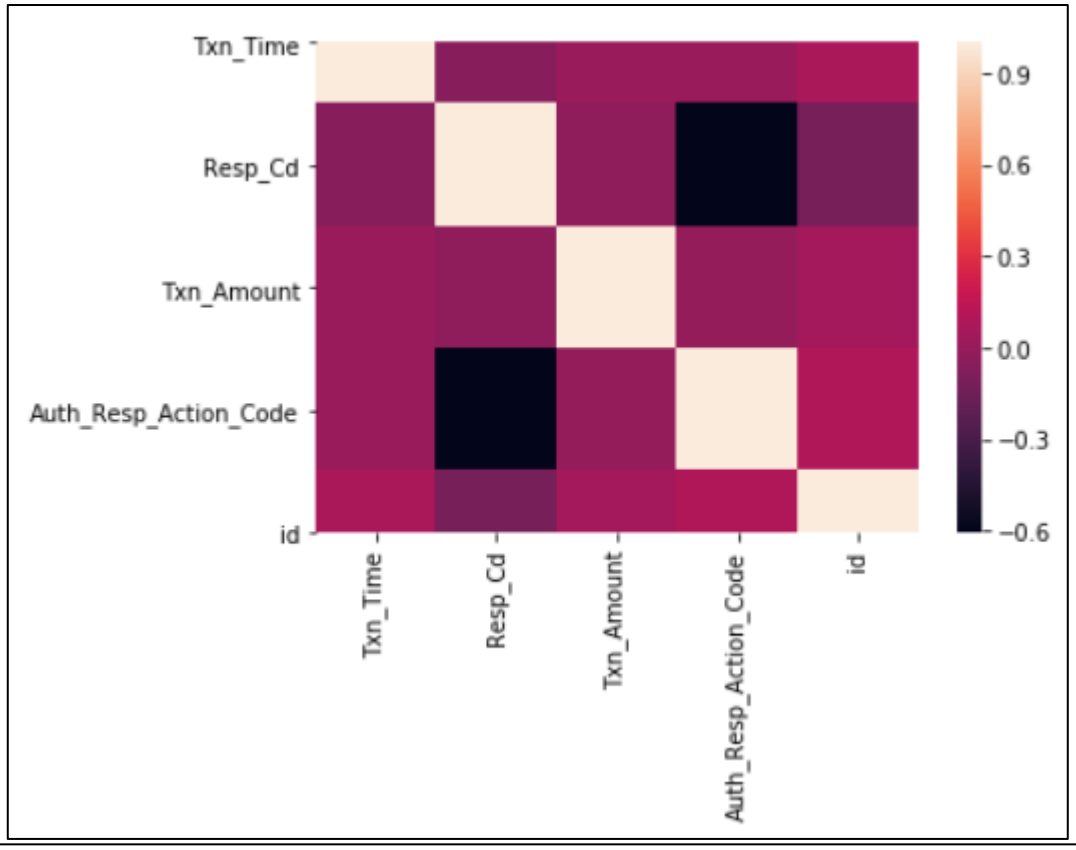


Figure 13. Decline/Approval proportion by product name.

Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regard to our class variable. One of the most visually appealing ways to determine that is by using a heatmap.



	Txn_Time	Resp_Cd	Txn_Amount	Auth_Resp_Action_Code	id
Txn_Time	1.000000	-0.053585	0.015215	0.010435	0.069867
Resp_Cd	-0.053585	1.000000	-0.023237	-0.604289	-0.117557
Txn_Amount	0.015215	-0.023237	1.000000	-0.007769	0.043019
Auth_Resp_Action_Code	0.010435	-0.604289	-0.007769	1.000000	0.094382
id	0.069867	-0.117557	0.043019	0.094382	1.000000

Figure 14. Heatmap of correlation.

As you can see, predictors do not seem to be correlated with the class variable. There seem to be relatively little significant correlations for such number of variables.

Looking on Figure 14 weak correlations spotted were:

- Time & Amount (0.015)
- Amount & Auth_Resp_Action_Code (-0.0078)
- Time & Auth_Resp_Action_Code (0.01)

We do not consider correlation between Auth_Resp_Action_Code and Resp_Code because these two features are on the database to represent a similar information, we will get rid of one of these features to move forward and to avoid any risk of multicollinearity.

- Auth_Resp_Action_Code & Resp_Code (-0.6)

6. DATA PREPARATION / FEATURE ENGINEERING

Merchant clustering

As there are multiple merchants on our dataset, we decided to use K-means clustering to group them and reduce the data dimensionality.

The objective of any clustering algorithm is to ensure that the distance between datapoints in a cluster is extremely low compared to the distance between two clusters. In other words, members of a group are remarkably similar, and members of different groups are extremely dissimilar.

K-means clustering will be used for creating merchants' segments based on their spend, time and product data. K-means clustering is an iterative clustering algorithm where the number of clusters K is predetermined, and the algorithm iteratively assigns each data point to one of the K clusters based on the feature similarity. Using the concept of minimizing within cluster sum of square (WCSS), the decision about the optimum number of clusters K was made. As the number of clusters increase, the WCSS keeps decreasing. The decrease of WCSS is initially steep and then the rate of decrease slows down resulting in an elbow plot. The number of clusters at the elbow formation usually gives an indication on the optimum number of clusters. This combined with business requirement was used to decide on the optimum number of clusters K = 4.

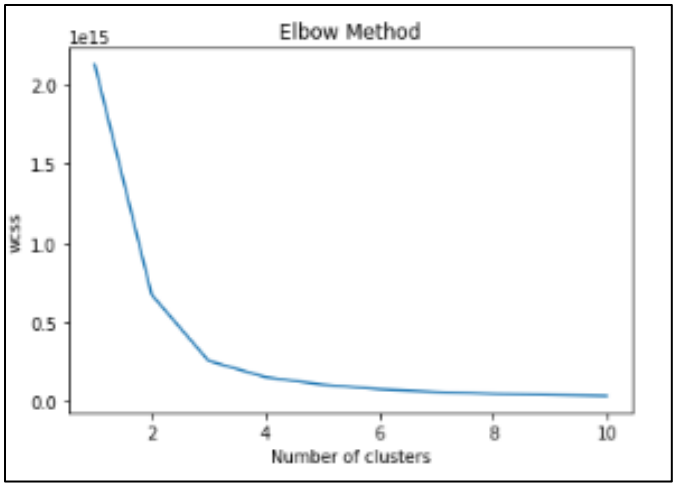


Figure 15. Elbow method

K-means method identifies cluster centroids as follows.

```
kmeans = KMeans(n_clusters=4).fit(Xc)
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[1.10696203e+05 3.71607286e+01 6.09481432e-01 1.17574768e+04]
 [2.10359129e+05 2.83190550e+01 4.96969838e-01 1.22575681e+04]
 [2.67693542e+04 1.98203229e+01 4.29560911e-01 1.11016782e+04]
 [1.63572226e+05 3.87251697e+01 5.34535695e-01 1.19378339e+04]]
```

Figure 16. Clusters centroids

Since transaction amounts are highly right skewed is difficult to visualize the data points in the single graph without over-plotting.

Finally, we proceed to add clusters labels onto our original dataset grouping the merchants into one of the four defined clusters using K-means method. Also, ID column was added representing a unique merchant name on the database.

Merch_ID	Merch_Name	Merch_City	Merch_Country	Txn_Amount	Auth_Resp_Action_Code	Prod_Name	id	cluster
N1WYWUNP6AAYPPJ	NEURONUP.COM	BARCELONA	ESP	0.00	1	CORPORATE	13882	2.0
N1WYWUNP6AAYPPJ	NEURONUP.COM	BARCELONA	ESP	203.33	1	CORPORATE	13882	2.0
12289894	QR	CIUDAD AUTONOMA BUENOS AIRES	ARG	6.87	1	CORPORATE	15987	0.0
12289894	RUBENJUVESRL	CIUDAD AUTONOMA BUENOS AIRES	ARG	26.37	1	CORPORATE	16746	0.0
12289894	TUCUMANTEXTI	CIUDAD AUTONOMA BUENOS AIRES	ARG	104.45	1	CORPORATE	19427	0.0

Figure 17. Cluster label

As we can observe on Figure 17, each merchant name has a unique ID number and a cluster label assigned to them.

Time and amount variables scaling

When working with classification problem like this one, is a common practice to use a scaling tool. Scaling of time and amount variables transforms the data to where there is a mean of 0 and a standard deviation of 1, thus standardizing the data into a normal distribution. Scaling the database prior to running the test provides better results due to the wide range of time and amounts in it.

One hot encoding for categorical features

These are the categorical variables on our dataset that I will need to transform using one-hot encoding to let the machine learning algorithms take the most advantage of this information.

7. MODEL SELECTION

First, merchants will be grouped into different clusters based on the amount, time, and product of the transaction. Using non supervised learning, clusters will be identified creating segments or profiles of merchants based on each consumer historical transactional data and behavior. Then, this cluster label will be added to the dataset as a categorical feature and various supervised models will be implemented to decide if a transaction is declined or not. Machine learning algorithms like logistic regression, decision trees, random forest and boosting machines will be used to build this predictive model in Python.

This model intakes data from credit, debit and prepaid card transactions and then determines the likelihood that a specific transaction is approved based on correlations found between features from historical transactions for a certain customer profile. Then each of them gets a score on the confidence that it is a legitimate one or not.

This software will process the data with key features to determine if the consumer purchase is out of their historical parameters such as time of purchase (day and hour), location, recurrence, and amount as key features to identify abnormalities.

First, it was performed a 50/50 train-test-split on the data set. Then, to avoid overfitting, a resampling technique of k-fold cross validation was used, and model was fitted on k-folds before making predictions for the kth hold out fold. The process is repeated for every single fold and finally obtaining the average of the resulting

predictions. In this paper, some of the most popular classification algorithms were used such as:

- Logistic Regression
- Classification Trees
- Random Forest Classifier
- XGBoost Classifier

Logistic Regression

Brief method explanation

Logistic Regression is a machine learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (approve) or 0 (decline). In other words, the logistic regression model predicts P(Y=1) as a function of X. It requires the dependent variable to be binary and the factor level 1 of it should represent the desired outcome. Logistic regression is a statistical model that tries to minimize cost of how wrong a prediction is.

Train Model

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 18. Logistic regression train hyperparameters

Decision Trees Classifier

Brief method explanation

Decision trees learn how to best split the dataset into smaller and smaller subsets to predict the target value. The condition, or test, is represented as the “leaf” (node) and the

possible outcomes as “branches” (edges). This splitting process continues until no further gain can be made or a preset rule is met, e.g. the maximum depth of the tree is reached.

Train Model

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Figure 19. Classification trees train hyperparameters

Trained decision tree classifier had a max depth of 74 with 56,905 leaves.

Train Tuned Model

Finding the optimal value for max_depth is one way to tune the model. Table 1 below shows the accuracy for decision trees with different values for max_depth.

Tree max depth	Accuracy
1	0.669858446
2	0.686376298
3	0.689690839
4	0.706580891
5	0.712461654
6	0.722087925
…	…
18	0.788989927
19	0.790537496
20	**0.790553168**
21	0.789565858

Table 1. Classification trees max_depth search

It is important to keep in mind that max_depth is not the same thing as depth of a decision tree. Max depth is a way to preprune a decision tree. In other words, if a tree is already as pure as possible at a depth, it will not continue to split.

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=20, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

Figure 20. Classification trees tuned max_depth hyperparameter

Random Forest Classifier

Brief method explanation

Random Forest construct many individual decision trees at training. Predictions from all trees are pooled to make the final prediction, the mode of the classes for classification or the mean prediction for regression. As they use a collection of results to make a final decision, they are referred to as ensemble techniques.

Hyperparameter optimization

Using grid search cross validation algorithm, we look for the best hyperparameters on the train set before fitting it. As a result, we found that the best hyperparameter to use were as follows in Figure 21.

```
CV_rfc.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 100}
```

Figure 21. Random Forest grid search CV best hyperparameters

Train Model

Then, we fitted the random forest classifier with the best hyperparameter configuration found.

```
randomforest.fit(X_train, y_train)

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=8, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

Figure 22. Random Forest train hyperparameters

XGBoost Classifier

Brief method explanation

XGBoost is a decision tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. When it comes to small to medium structured or tabular data, decision tree-based algorithms are considered best in class right now. XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

Hyperparameter searching

Using randomized grid search cross validation algorithm, we look for the best hyperparameters on the train set before fitting it. As a result, we found that the best hyperparameter to use were as follows in Figure 23.

```
Model with rank: 1
Mean validation score: 0.778 (std: 0.019)
Parameters: {'colsample_bytree': 0.9004255409711938, 'gamma': 0.10249214770791049, 'learning_rate': 0.11794431907830398, 'max_depth': 5, 'n_estimators': 97, 'subsample': 0.6052007694042943}
```

Figure 23. XGBoost randomized grid search CV best hyperparameters

Train Model

For testing purposes, two XGBoost models were used on this simulation, one with default hyperparameters and another one with tuned hyperparameters to improve model performance on classifications.

```
xg_class.fit(X_train,y_train)

XGBClassifier(alpha=10, base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.3, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=10, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=0, reg_alpha=10,
              reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
              validate_parameters=False, verbosity=None)
```

Figure 24. XGBoost trained with default hyperparameters

```
xg_classT.fit(X_train,y_train)

XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.9004255409711938, gamma=0,
              gpu_id=-1, importance_type='gain', interaction_constraints=None,
              learning_rate=0.11794431907830398, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=97, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=0.6052007694042943,
              tree_method=None, validate_parameters=False, verbosity=None)
```

Figure 25. XGBoost trained with tuned hyperparameters

8. DETERMINE PERFORMANCE METRICS

Common formulas to be used as performance metrics will be:

TP = True Positive. Approved transactions the model predicted as approved.

TN = True Negative. Declined transactions the model predicted as declined.

FP = False Positive. Declined transactions the model predicted as approved.

FN = False Negative. Approved transactions the model predicted as declined.

Accuracy is one metric for evaluating classification models. It is the fraction of predictions the model gets right. Accuracy it´s a good metric to measure how well a model performs.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.

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

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

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

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0. The F-beta score weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.

$$F1score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Since we will select the model that has a good prediction power and the minimum cost associated, the False Positive rate is also a key KPI to follow on result analysis. As the cost associated to predict as an approval a fraudulent transaction that should be decline cost the full amount of that transaction, we want to minimize this amount of FP to control the cost associated to it. False positive rate is the probability of falsely rejecting the null hypothesis for a particular test. It is calculated as the ratio between the number of negative events wrongly categorized as positive and the total number of actual negative events.

$$FP\ rate = \frac{FP}{FP + TN}$$

The total cost calculation considers the associated cost of incurring into each one of the different situations shown on Table 2.

	Actual Positive $y_i = 1$	Actual Negative $y_i = 0$
Predicted Positive $c_i = 1$	C_{TP_i}	C_{FP_i}
Predicted Negative $c_i = 0$	C_{FN_i}	C_{TN_i}

Table 2. Classification cost matrix

True Positive associated cost is $C_{TP_i} = 0$ since there is no cost associated to a legitimate transaction that is classified as approved by the machine learning algorithm.

False Positive associated cost is $C_{FP_i} = Amount_i$ since there is a cost associated to a declined transaction that is being classified as approved by the algorithm. In real life situation, the issuer claims the full amount of the approved transaction to the company. Therefore, this KPI is one of the most important ones to consider in the result section since economic impacts of misclassification could drive significant financial losses.

Finally, False Negative and True Negative associated cost is $C_{FN_i} = C_{TN_i} = Ca_i$ where Ca is the administrative associated cost to investigate a suspicious transaction. On the studied company, takes an average of fifteen minutes for an analyst to look for further information when a transaction is flagged and decided whether to let it go through or

rejecting it. Customer and issuer can contact support team in real time to ask for action when a False Negative takes place. On the other hand, when a True Negative occurs, the studied company also investigates the origin of it to identify the root cause. This also helps the company to detect fraudsters and improve available detection methods that are set to contain them. Considering the wage per minute of this sample analyst, administrative cost is $Ca_i = \$3$ per transaction misclassified. Then, total cost formula is defined as follows:

$$Total\ Cost = TP * 0 + \Sigma\ Amount(FP)_i + FN * Ca + TN * Ca$$

9. RESULTS

Performance metrics

	Precision	Recall	F1 Score	Accuracy	Total Cost	FP Rate
Logistic Regression	0.29	0.53	0.37	0.53	$ 588,188.24	0.465
Decision Tree	0.76	0.76	0.76	0.76	**$451,948.97**	0.218
Decision Tree Tuned	0.79	0.79	0.79	0.79	$ 457,492.36	**0.198**
Random Forest	0.69	0.69	0.68	0.69	$ 572,257.01	0.321
XGBoost	0.70	0.70	0.69	0.70	$ 553,052.38	0.311
XGBoost Tuned	**0.78**	**0.78**	**0.78**	**0.78**	$ 460,719.43	0.219

Table 3. Model results

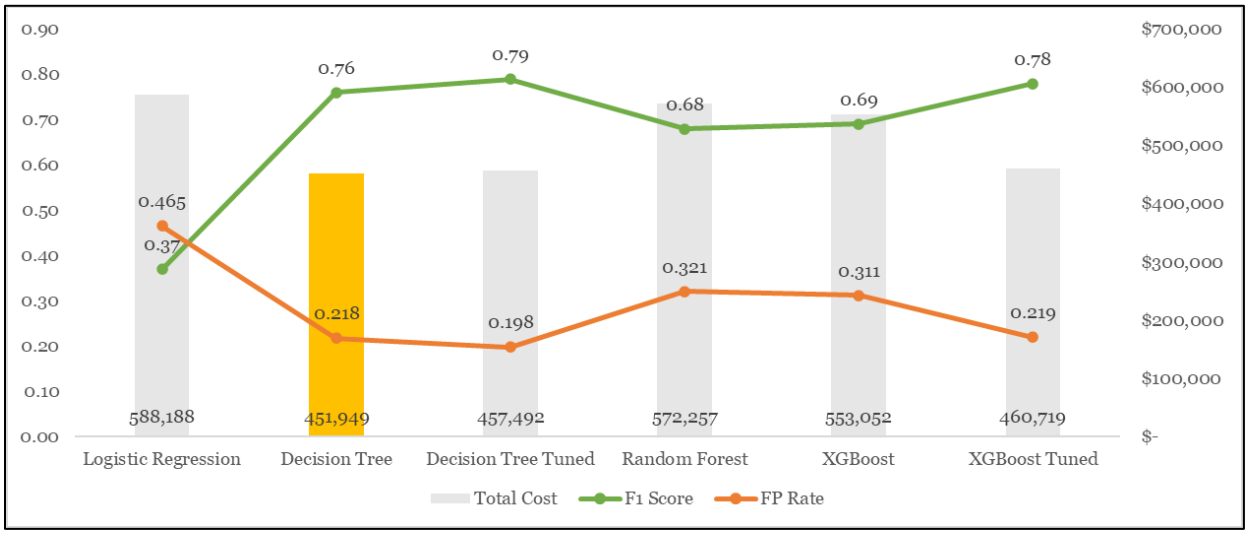


Figure 26. Model results

After training each of the models, these are the results. All the scores except for Logistic Regression model are very promising for our studied dataset. Each model has a low False Positive rate, which is exactly what we are looking for.

Decision tree algorithm outperforms the other in terms on cost reduction ($451,949) being also extremely competitive in terms of F1 Score and FP Rate versus a more complex and black-box algorithm such as XGBoost. This is a huge advantage since a decision tree

is an explainable model where it is easy to understand the most significant variables that define the predictions.

Lastly, we will go over the ROC curve, the Confusion Matrix, and how each model stacks up.

Receiver operating characteristics (ROC)

The ROC is a performance measurement for classification problems at various thresholds. It is essentially a probability curve, and the higher the Area Under the Curve (AUC) score the better the model is at predicting approved/declined transactions.

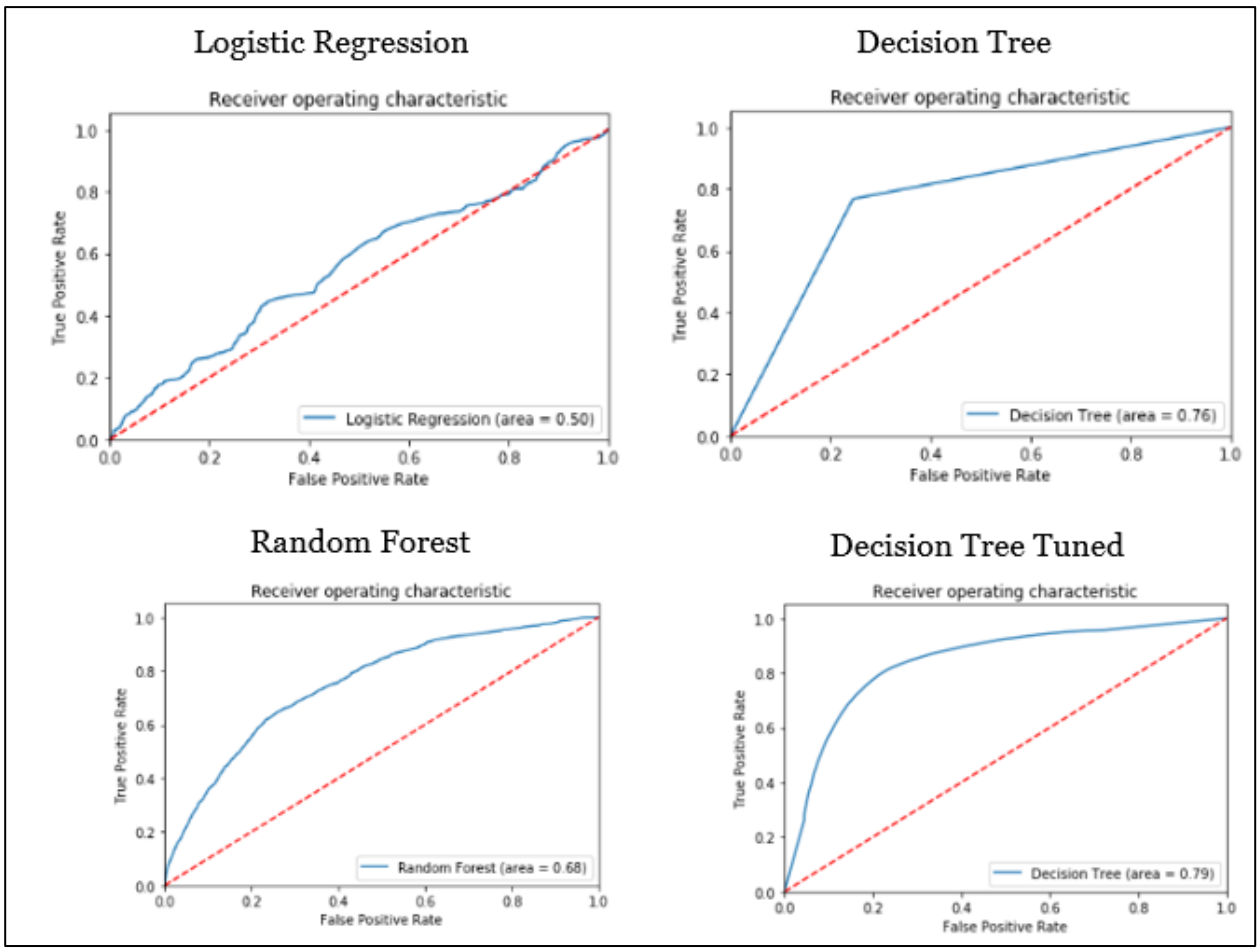


Figure 27. ROC curves

The receiver operating characteristic (ROC) curve is another common tool used with binary classifiers. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

In the graph above, the AUC scores for Decision Tree and Decision Tree Tuned are pretty high, which is what we would like to see. It is important to note that each point on the curve indicates a threshold. As we move further right along the curve, we both capture more True Positives but also incur more False Positives. This means we capture more fraudulent transactions, but also flag even more normal transactions as fraudulent.

Confusion matrix

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. It gives insight not only into the errors being made by your classifier but more importantly the types of errors that are being made. It is this breakdown that overcomes the limitation of using classification accuracy alone.

	TP	FP	FN	TN
Logistic Regression	0%	47%	**0%**	**53%**
Decision Tree	11%	12%	41%	41%
Decision Tree Tuned	**36%**	**11%**	10%	43%
Random Forest	27%	20%	12%	42%
XGBoost	27%	19%	11%	43%
XGBoost Tuned	34%	12%	10%	43%

Table 4. Confusion matrix for all the models

As mentioned before, we are looking for a model that minimize the False Positive amount. Decision trees seems to be the ones that make the lowest amounts of this kind of misclassifications that is why we will prefer them since cost impacts will be diminished.

10. SELECTED MODEL

Decision Tree Classifier

```
print(arbol.tree_.max_depth)
74
print(arbol.tree_.n_leaves)
56905
```

Fig 28. Decision Tree max depth and leave number

We decided to select this Decision Tree to use since it is the one that has the minimum total cost among all algorithms tested despite the fact of having the second-best FP rate (see Table 3).

Performance metrics

Accuracy of the decision tree classifier on test set is 0.76.

```
print(classification_report(y_test, predictDT))
              precision    recall  f1-score   support

           0       0.74      0.75      0.75    118749
           1       0.78      0.77      0.77    136490

    accuracy                           0.76    255239
   macro avg       0.76      0.76      0.76    255239
weighted avg       0.76      0.76      0.76    255239
```

Fig 29. Decision Tree classification report

Confusion matrix

Confusion matrix results show that we have 194,339 correct predictions and 60,900 incorrect predictions on our test dataset.

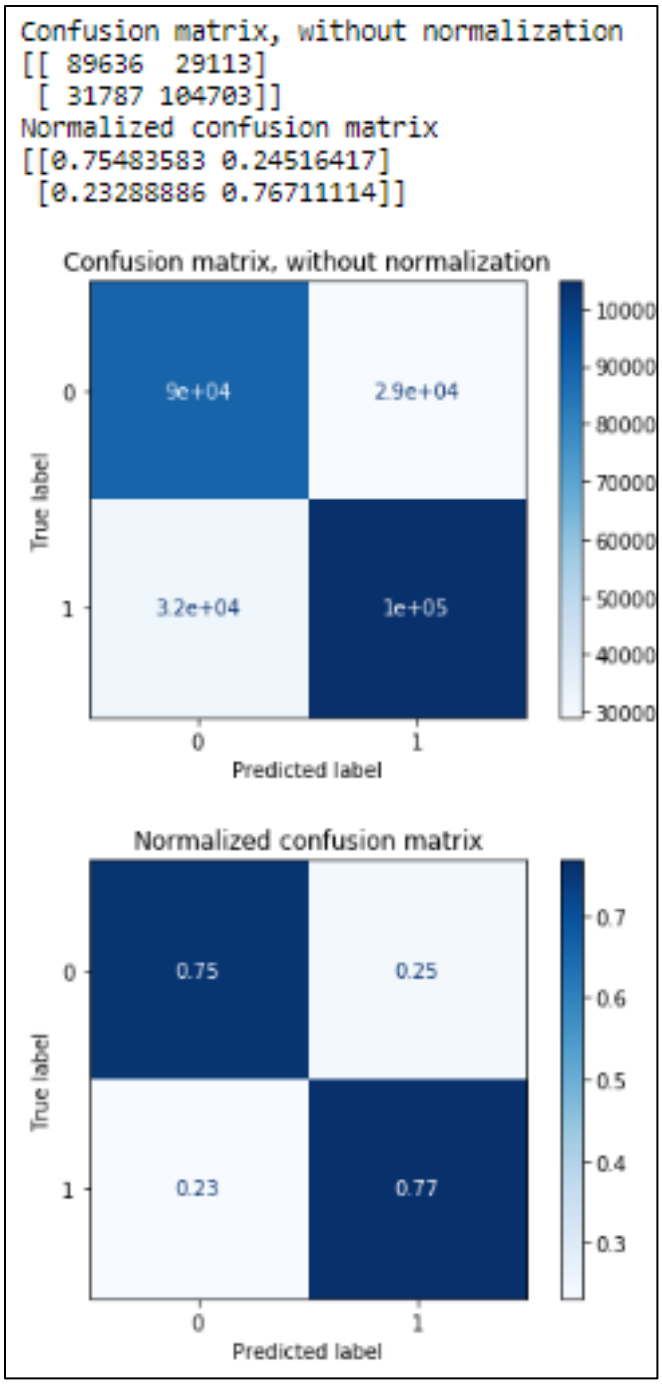


Fig 30. Confusion matrix for decision tree model.

Feature importance

CART algorithm was used for feature importance implemented in scikit-learn as the Decision Tree Classifier classes. After being fitted, the model provides a feature importance property that can be accessed to retrieve the relative importance scores for each input feature.

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature.

	feature	importance
1	scaled_time	0.278
1609	PREPAID	0.200
0	scaled_amount	0.186

Figure 31. Top 3 features sorted by importance of Decision Tree

In the example above and for a particular train test split of dataset, the scaled_time has the highest feature importance weight. If a feature has a low feature importance value, it does not necessarily mean that the feature is not important for prediction, it just means that the particular feature was not chosen at a particularly early level of the tree. It could also be that the feature could be identical or highly correlated with another informative feature. Feature importance values also do not tell which class they are very predictive for or relationships between features which may influence prediction.

11. BUSINESS APPLICATIONS and CONCLUSION

Nowadays most of the transactions take place online, meaning that credit cards and other online payment systems are involved. This method is convenient both for the company and for the consumer. Consumers save time because they do not have to go to the store to make their purchases and companies save money by not owing physical stores and avoiding expensive rental payments. It seems that the digital age brought some highly useful features which changed the way that both companies and consumers interact with each other but with one cost, the necessity to invest significant amounts of money to make sure that all the transactions are legal and non-fraudulent.

The performance of different machine learning algorithms was assessed to minimize the misclassifications costs. Decision tree tuned algorithm was the one that minimized this cost function reducing False Positive cases that are the most expensive ones for the company since full amount of the transaction is lost.

This is useful for credit card business since its core revenue stream comes from charging financial institutions that issue card-branded payment products a fee based on gross dollar volume of activity. Having a trustable algorithm that approves legitimate transactions and rejects fraudulent ones accurately increase brand salience among consumers and issuers driving usage that will increase gross dollar volume.

As we analyzed on the dataset, most of these transactions are from a relatively low value ticket making difficult to classify if only few features are being used. From feature analysis we could find that the datetime of the transaction, prepaid product and the amount are the key features to predict correctly if a transaction should be approved o declined.

Fraud detection is a complex issue that requires a substantial amount of planning before throwing machine learning algorithms at it. Nonetheless, it is also an application of data science and machine learning for the good, which makes sure that the customer's money is safe and not easily tampered with.

Future work streams to improve this paper will be to implement a cost-sensitive approach for the tree-based model. This will be to train a model with a loss function that minimizes

the actual costs instead of misclassification errors. In this case a loss function with the costs associated with each of the four cases (TP, TN, FP, FN) will be provided so that the model can learn to make optimal predictions accordingly.

12. LITERATURE REVIEW

The purpose of this section is to provide a review of the papers that have been published in areas related to the topic. Many empirical studies have been conducted on the subject of “Transaction Authorization” and “Credit Card Fraud” in the last decade. The major emphasis of research has been on various issues like frauds, security, usage pattern, consumer behavior, e-payments and cost sensitive decision making, among others. Most of the literature is focused on solving the fraud detection problem. Nevertheless, using the results and key findings of these papers, different algorithms were applied to solve proposed problem that is to decide whether a new incoming transaction should be approved or declined. The review provides support to the objectives of this study and act as a guidance to the study´s design.

Banhsen [1]. In this paper a new comparison measure that realistically represents the monetary gains and losses due to false declines is proposed. The results of this paper are based on a real-life transactional data provided by a large European card processing company. This paper was used as a reference to compare standard algorithms, using both classical measures and the proposed financial measure. A cost sensitive model is developed in order to integrate real financial costs due to credit card false declines.

Banhsen [2]. The paper proposed a framework that consists in creating different example-dependent cost sensitive Decision Trees on random subsamples of the training set and evaluate the proposed method against state-of-the-art cost sensitive techniques. From this paper was extracted the notion of using a 2x2 cost matrix that represents a binary classification cost. It introduces the costs associated with two types of correct classification, true positives, true negatives, and the two types of misclassification errors, false positives, and false negatives. Conceptually, the cost of correct classification should always be lower than the cost of misclassifications. These are referred to as the reasonableness conditions.

Bhattacharyya [3] did a detailed comparative study of Support Vector Machine and Random Forest along with Logistic Regression. He concluded through experiments that Random Forest technique shows most accuracy followed by a Logistic Regression

and Support Vector Machine. Using this key insight for this paper, I decided to run the three algorithms on test sample to find the top performer transaction authorization model. As stated on the paper and tested on this work, Random Forest outperforms Logistic Regression having a lower false positive rate and also a lower misclassification cost.

Chougule [4] proposed simple K-means algorithm for fraud detection. In this paper he showed how k-means algorithm produced clusters which were then optimized by a genetic algorithm. The notion of creating clusters using k-means method was used on this work to add a new feature to the raw dataset. Extracting key information from the dataset, four clusters were created, and their label attached to each row on the dataset. K-means clustering was used in order to group together the suspected declined transactions into a similar cluster. The output of this stage is used to train the algorithms which then classify the incoming transactions.

Dornadula [5] states that researchers stated using different machine learning methods to detect and analyze frauds in online transactions. The main goal of the paper is to design and develop a novel fraud detection method for transaction data, with the objective of analyzing the past transaction details of the customers and extract their behavioral patterns. As explained on the paper, using the technique of clustering cardholders into different groups based on key features such as transaction amount will improve model prediction power since these groups put together those declined transactions according to their feature's similarities.

Jain [6]. An extensive review is done on the existing and proposed models for credit card authorization and fraud detection. A comparative study on these techniques on the basis of quantitative measurements such as accuracy, detection rate and false alarm rate were topics of this paper. The conclusion of this study explains the drawbacks of existing models and provides a better solution in order to overcome them. Some techniques available to solve stated problem were explained such as K-Nearest Neighbor (KNN) and Decision Trees.

Jagdeesh [7] put a light on credit card fraud which is increasing worldwide. The culprit is not only the outsiders but insider fraudsters who cheat their organization to make

quick incomes. Bank credit card issuers lose about $1.5 to $ 2 billion every year because of fraud. VISA and Mastecard, the two largest credit card issuers lose most. Major credit card declines like unauthorized use of credit cards, insufficient funds, online frauds, shave and paste of card, counterfeiting, mail order fraud are the main causes of transaction declines. The author also discusses the tips for prevention of false declines like using smart cards, computer edits, PIN numbers, and suggests that it is in their own interest that the cardholders should keep their cards safely and use the cards wisely to protect themselves from frauds.

Sahin and Duman [8] proposed fraud detection in credit card using a combination of Support Vector Machines and Decision Trees. Decision Trees outperformed SVMs when the size of data set was considerable. As dataset used for this work had several rows it can be considered as a large dataset. The information and conclusions of this paper helped with the decision of not to include SVM algorithm on model test due to the low performance when comparing to the one that Decision Trees had.

Zareapoor [9] states that there is a lack of published literature on credit card transaction authorization techniques, due to the unavailable credit card transactions datasets for researchers. He proposed a bagging ensemble classifier based on decision tree algorithms as a novel technique in area of transaction authorization. On this work data mining techniques exposed on the paper were used to train the winner Decision Tree and also to tune it trying to get a low false positive rate with a fewer overall cost of misclassifications.

13. REFERENCES

[1] Bahnsen, “Cost sensitive credit card fraud detection using Bayes minimum risk”, 2015.

[2] Bahnsen, “Ensemble of example-dependent cost-sensitive Decision Tress”, 2015.

[3] Bhattacharyya, “Data mining for credit card fraud: a comparative study”, 2011.

[4] Chougle, “Genetic K-Means algorithm for credit card fraud detection”, International journal of computer science and information technologies, 2015.

[5] Dornadula, “Credit card fraud detection using machine learning algorithms”, 2019.

[6] Jain, “A comparative analysis of various credit card fraud detection techniques”, 2019.

[7] Jagdeesh, “Credit card fraud causes and cures from professional’s perspective”, 2005.

[8] Sahin, Duman, “Detecting credit card fraud by Decision Trees and Support Vector Machines”, 2011.

[9] Zareapoor, “Application of credit card detection”, 2015.

14. BIBLIOGRAPHY

- Awoyemi, et al. “Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis.” 2017 International Conference on Computing Networking and Informatics (ICCNI), 2017.
- Brownlee, what is a confusion Matrix in Machine Learning (2016), Machine Learning Mastery, 2016. Available at: *https://machinelearningmastery.com/confusion-matrix-machine-learning/*.
- Chen, Tianqi, and Carlos Guestrin (2016). XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD, 2016.
- Demsar, “Statistical Comparisons of Classifiers over Multiple Data Sets”, Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006.
- Elkan, “The Foundations of Cost-Sensitive Learning”, in Seventeenth International Joint Conference on Artificial Intelligence, pp. 973–978, 2001.
- Galarnyk, Understanding Decision Trees for Classification (Python), Towards data science, 2019. Available at: *https://towardsdatascience.com/understanding-decision-trees-for-classification-python-9663d683c952*.
- Hastie, Tibshirani, and Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. Stanford, CA: Springer, 2009.
- Japkowicz, “Learning from imbalanced data sets: a comparison of various strategies”, Faculty of Computer Science, DalTech, 2000.
- Macaraeg, Credit Card Fraud Detection, Towards data science, 2019. Available at: *https://towardsdatascience.com/credit-card-fraud-detection-a1c7e1b75f59*.
- Machine Learning Group — ULB, Credit Card Fraud Detection, Kaggle, 2018. Available at: *https://www.kaggle.com/mlg-ulb/creditcardfraud*.
- Marr, “The amazing way how Mastercard uses AI to stop fraud and reduce false declines”, Artificial Intelligence in Practice, Wiley, 2018.
- McFarlane, How Mastercard makes money, Investopedia, 2019. Available at: *https://www.investopedia.com/articles/markets/032615/how-mastercard-makes-its-money-ma.asp*.

- Morde, XGBoost Algorithm, Towards data science, 2019. Available at: *https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d*.
- Moser, Fraud Detection with Cost-Sensitive ML, Towards data science, 2019. Available at: *https://towardsdatascience.com/fraud-detection-with-cost-sensitive-machine-learning-24b8760d35d9*.
- Ngai, Hu, Wong, Chen, and Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," Decision Support Systems, vol. 50, no. 3, pp. 559–569, 2011.
- Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, R. Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
- Ronaghan, The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-Learn and Spark, Towards data science, 2018. Available at: *https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3*.
- Vadera, "CSNL: A cost-sensitive non-linear decision tree algorithm," ACM Transactions on Knowledge Discovery from Data, vol. 4, no. 2, pp. 1–25, 2010.
- Whitrow, Hand, Juszczak, Weston, and Adams, "Transaction aggregation as a strategy for credit card fraud detection," Data Mining and Knowledge Discovery, vol. 18, no. 1, pp. 30–55, Jul. 2008.
- Xuan, et al. "Random Forest for Credit Card Fraud Detection." 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018.

15. APPENDIX

Dataset Covariates and Target

```
print(dff.columns)

Index(['Prod_Name_old', 'Txn_Cnt', 'Txn_Date', 'Txn_Time', 'Resp_Cd',
       'Auth_Resp', 'Card_Pres_Cd', 'Merch_ID', 'Merch_Name', 'Merch_City',
       'Merch_Country', 'Iss_Name', 'Product Group', 'Acq_Country_Cd',
       'Acq_Name', 'Txn_Amount', 'Auth_Resp_Action_Code', 'Prod_Name'],
      dtype='object')
```

Figure 32. Raw dataset column names.

Input variables

- Transaction Count: Txn_Cnt (numeric: row counter)
- Transaction Date: Txn_date (datetime)
- Transaction Time: Txn_Time(numeric)
- Response Code: Resp_Cd(numeric)
- Card Present Code: Card_Pres_Cd(numeric: 1=”card not present transaction”; 0=”card present transaction”)
- Merchant Name: Merch_Name (categorical: “Netflix”, “Spotify”, “Uber” among others)
- Merchant ID: Merch_ID merchant identifier, one merchant may have multiple ID´s depending on their product/services (numeric)
- Merchant city: Merch_City (categorical: “San José”, “Stockholm”, “Buenos Aires” and others)
- Merchant Country: Merch_Country (categorical: “USA”, “Sweden”, “Argentina” and others)
- Issuer Name: Iss_Name (categorical: “Issuer 1”, “Issuer 2”, “Issuer 3” and others)
- Product Group: (categorical: “Debit”, “Credit”, “Prepaid”)
- Acquirer Country Code: Acq_Country_Cd (numeric)
- Acquirer Name: Acq_Name(categorical: “Cielo S.A.”, ”Stripe Payments UK”, ”First Data Cono Sur” and others)

- Transaction amount in USD: Txn_Amount (numeric)
- Product segment (categorical: “Black”, “Platinum”, “Gold”, “Standard”, “Debit”, “Prepaid” and “Commercial”)