**Modelling ELO Customer Loyalty**

Contents

[Introduction: 3](#_Toc83827801)

[Source: 3](#_Toc83827802)

[Problem statement: 3](#_Toc83827803)

# Introduction:

Elo is a Brazilian financial services association founded in 2011. It issues credit, debit and prepaid cards. As of March 2019, Elo has issued over 120 million cards.

Cards issued by Elo being one of the most prolific transaction media in Brazil, the company has over a period of time built up partnerships with numerous merchants introducing promotional offers, discounts, additional customer services etc. All this is expected to help Elo increase its business, customer base and customer retention.

Towards that end Elo has developed a metric, called customer loyalty. The more the customer loyalty of a certain customer the more business Elo can expect from that particular individual. Elo has also tried to design a predictive model for this metric, so that decisions regarding discounts, promotional offers etc., can be made thereby increasing Elo’s market. However, it is not up to the mark and our aim is to come up with a better model.

## Source:

The source of this problem statement and the relevant datasets is the following competition in Kaggle – [Elo Merchant Category Recommendation](https://www.kaggle.com/c/elo-merchant-category-recommendation/).

# Problem statement:

We have the following datasets at our disposal-

* historical\_transactions.csv – Contains the data for transactions performed on each card-id over a period of 14 months (from .. to ..).
* new\_merchant\_transactions.csv – contains the data for transactions of each card-id over the next two months
* train.csv – contains training data i.e., a few anonymous features and the target loyalty score, which ELO was using to try predict the loyalty scores of card-holders.
* test.csv – contains test data; similar to train.csv except that the target scores are not present here.

First, it was observed that the training set being used by ELO could not predict the loyalty scores at all. Hence the following strategy was chosen to build a predictive model –

1. Use historical\_transactions.csv and new\_merchant\_transactions.csv to produce new features for the cards.
2. Fit a model (along with hyperparameter tuning) to understand the predictive power of the features.
3. Run feature selection so as to improve model test set performance.

There were many repetitions of this feature creation 🡪 model fit 🡪 feature selection 🡪 feature creation 🡪 … cycle unless an acceptable accuracy was reached.

Each of these steps will be described in detail in the upcoming sections.

# Technologies used

## Spark and non-spark versions

There were two versions of this project – spark and non-spark. In the spark version we used PySpark 3.1.2 – the Dataframe API and MLLib module. However, using PySpark and that too in Colab environment posed a number of challenges –

* Customizing the spark environment settings like number of workers, memory/worker etc., is not possible in Colab (there was only one driver). Besides, a distributed Hadoop system, like that provided in Azure Databricks, was also not available. Hence it was not possible to avail the speed of distributed processing facility of Spark. On the contrary, lazy evaluation with only a single available driver and no workers, started posing memory issues.
* Many useful features like null-imputation, feature importance, cross-validation have not been implemented in MLLib as richly as in sklearn. For e.g., -
  1. Null imputation does not support categorical features and only has naive imputation techniques like mean, median and mode imputation (see documentation [here](https://spark.apache.org/docs/latest/ml-features.html#imputer)).
  2. Cross-validation in MLLib can only be used in hyperparameter tuning mode and not as just a method to get test set performance estimates (see the question [here](https://stackoverflow.com/questions/69148599/is-there-a-way-to-use-spark-mllib-crossvalidator-without-parameter-grid)).

All these shortcomings led to finding alternatives. Using some memory reductions, pandas was able to handle the datasets. Hence we switched to the usual non-distributed platform using numpy, pandas, sklearn etc., during the final model-training phase.

Please find here the spark and non-spark versions of the notebook.

## Languages, modules

In the course of this project following languages and modules were used –

* Spark Dataframe API, Spark SQL, MLLib in the spark version