Joël Holla
December 12, 2018

Definition

Project Overview

[1] Imagine being in a place that you have never been to before and getting restaurant recommendations served up, based on your personal preferences. The recommendation comes with an attached discount from your credit card provider for a local place around the corner. Companies such as Elo, one of the largest payment brands in Brazil, has built partnerships with merchants in order to offer promotions or discounts to cardholders. We must ask ourselves the questions: Do these promotions work for either the consumer or the merchant? Does the consumer enjoy their experience? Personalization is the key that will help us unlock these questions. Elo has built machine learning models to understand the most important aspects and preferences in their customers' lifecycle. So far none of these models is specifically tailored for an individual or profile.

[2] Personalized marketing, also known as one-to-one marketing is one of the many tools that is being used by companies. Individual marketing is a marketing strategy by which companies leverage data analysis and digital technology to deliver personalized messages and product offerings. This strategy is dependent on many different types of technologies for data collection, data classification, data analysis, data transfer, and data scalability.

This type of marketing is not only a benefit for both the different businesses and companies, it also can greatly aid and save time for the consumer. Before the Internet, consumers did not face such a wide range of variety and volume of products and services. In this modern age, however, for example, a single retail website can offer thousands of different products and services. Personalized marketing helps to bridge the gap between the vastness of what is available and the needs of individual consumers, which can greatly enhance the consumer's experience.

A few examples of machine learning problems that are similar to this project are the following:
[3] IBM, through the use of IBM Watson and the plethora of customer data, using the power of artificial intelligence (e.g. predictive analytics) to improve how they manage customer relationships to increase customer loyalty. This is done by examining the different type of loyalty behaviours and the factors associated with it.
[4] Airbnb, a leader among p2p residential real estate focus their efforts on enhancing the search experience by using the AI models to analyze more than a hundred signals at once to help personalize the search in real time.
[5] A study that was done in 2007, by Wouter Buckinx, Geert Verstraeten and Dirk Van den Poel, predicting customer loyalty using the internal transactional database.
[6] A project by Ashish Gandhe describes the work to learn to predict whether a given yelp user visiting a restaurant will like it or not.

My personal motivation with this project is to see how data can aid businesses sectors, such as retail, grow and adapt to this ever-changing world.

Problem Statement

The goal is to create an algorithm to identify and serve the most relevant opportunities to individual customers by uncovering signal of customer loyalty.

1. Download and preprocess the datasets.
2. Train the model that can determine the customer loyalty score.

The final model is expected to improve the customers' lives, help companies such as Elo reduce unwanted campaigns, and to create the right experience for their customers.

As this is a regression problem, the approach I will take to model and create predictions to determine the customer loyalty score are the following: I will try several different algorithms such as a Ridge Regression, LightGBM and XGBoost and compare them to a benchmark model of a Linear Regression. I will then tune the best performing model to see if I can a greater performance model.

Metrics

The metric that will be used to evaluate how well the model fits the data, i.e.to see how close the predicted values by the model are to the actual values will be the **RMSE (Root Means Square Error).**
The mathematical definition for RMSE can be defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

where,
n = number of observations
y_i = actual value of target variable
$y_i\ hat$ = predicted values of target variable

In this project, I will calculate the RMSE, by calculating the square root of mean square error () function provided in the metrics module of sci-kit learn library.

[7] The RMSE is a good metric for this type of problem because takes the squared errors which makes it sensitive to outliers in error distribution proving this metric to be a good representation of error distribution. This means that it's an appropriate metric for samples with a normal distribution. The main advantage of the RMSE metrics over several other metrics is that it doesn't use the absolute error. This is a highly undesirable metric to use when we are aiming to minimize the cost function or optimizing model performance because it becomes difficult to calculate the gradient of absolute errors.

Analysis

Data Exploration

Input:
There are 5 datasets which are as followed:

Train.csv:

- Numbers of rows = 201917
- Number of columns = 6
- Highly relevant as this is the data, I will be testing my models on.

Columns	Description
card_id	Unique card identifier
first_active_month	'YYYY-MM', month of first purchase
feature_1	Anonymized card categorical feature
feature_2	Anonymized card categorical feature
feature_3	Anonymized card categorical feature
target	Loyalty numerical score calculated 2 months after historical and evaluation period

Test.csv:

- Numbers of rows = 123623
- Number of columns = 5
- Highly relevant as this is the data, I will be testing my models on.

Columns	Description
card_id	Unique card identifier
first_active_month	'YYYY-MM', month of first purchase
feature_1	Anonymized card categorical feature
feature_2	Anonymized card categorical feature
feature_3	Anonymized card categorical feature

Historical_transaction.csv:

Columns	Description
card_id	Card identifier
month_lag	month lag to reference date
purchase_date	Purchase date
authorized_flag	Y' if approved, 'N' if denied
category_3	anonymized category
installments	number of installments of purchase
category_1	anonymized category
merchant_category_id	Merchant category identifier (anonymized)
subsector_id	Merchant category group identifier (anonymized)
merchant_id	Merchant identifier (anonymized)
purchase_amount	Normalized purchase amount
city_id	City identifier (anonymized)
state_id	State identifier (anonymized)
category_2	anonymized category

New_merchant_transactions.csv

Columns	Description

card_id	Card identifier
month_lag	month lag to reference date
purchase_date	Purchase date
authorized_flag	Y' if approved, 'N' if denied
category_3	anonymized category
installments	number of installments of purchase
category_1	anonymized category
merchant_category_id	Merchant category identifier (anonymized)
subsector_id	Merchant category group identifier (anonymized)
merchant_id	Merchant identifier (anonymized)
purchase_amount	Normalized purchase amount
city_id	City identifier (anonymized)
state_id	State identifier (anonymized)
category_2	anonymized category

Merchants.csv:

	merchants.csv	Unnamed: 1
0	NaN	NaN
1	Columns	Description
2	merchant_id	Unique merchant identifier
3	merchant_group_id	Merchant group (anonymized)
4	merchant_category_id	Unique identifier for merchant category (anony...
5	subsector_id	Merchant category group (anonymized)
6	numerical_1	anonymized measure
7	numerical_2	anonymized measure
8	category_1	anonymized category
9	most_recent_sales_range	Range of revenue (monetary units) in last acti...
10	most_recent_purchases_range	Range of quantity of transactions in last acti...
11	avg_sales_lag3	Monthly average of revenue in last 3 months di...
12	avg_purchases_lag3	Monthly average of transactions in last 3 mont...
13	active_months_lag3	Quantity of active months within last 3 months
14	avg_sales_lag6	Monthly average of revenue in last 6 months di...
15	avg_purchases_lag6	Monthly average of transactions in last 6 mont...
16	active_months_lag6	Quantity of active months within last 6 months
17	avg_sales_lag12	Monthly average of revenue in last 12 months d...
18	avg_purchases_lag12	Monthly average of transactions in last 12 mon...
19	active_months_lag12	Quantity of active months within last 12 months
20	category_4	anonymized category
21	city_id	City identifier (anonymized)
22	state_id	State identifier (anonymized)
23	category_2	anonymized category

The goal of this project is to train a model that will have the ability to predict a *loyalty score* or the 'target' column from the training_set for each of card_id represented in the test_set. The train.csv is the dataset that the model will be trained on, the training_set is made of 201917 records and the test_set is made of 123623 records.

The historical_transaction.csv contains up to 3 months' worth of historical transactions for each card_id. The new_merchant_transactions.csv contains two months' worth of data for each card_id containing ALL purchases that the card_id made at merchant_id's that were not visited in the historical data. The historical_transactions.csv and new_merchant_transactions.csv are designed to be joined with the train.csv and the test.csv as will be shown in the exploratory visualization below.

Output:

After merging the datasets together, the final training_set has 43 columns and 201917 rows, with 20 features and one response variable. Fig 1. shows a complete list after combining the historical_transactions.csv and new_merchant_transactions.csv.

```
first_active_month            201917 non-null datetime64[ns]
card_id                       201917 non-null object
feature_1                     201917 non-null int64
feature_2                     201917 non-null int64
feature_3                     201917 non-null int64
target                        201917 non-null float64
year                          201917 non-null int64
month                         201917 non-null int64
elasped_time                  201917 non-null int64
historical_trans_count        201917 non-null int64
hist_authorized_flag-sum      201917 non-null int64
hist_authorized_flag-mean     201917 non-null float64
hist_merchant_id-nunique      201917 non-null int64
hist_city_id-nunique          201917 non-null int64
hist_purchase_amount-sum      201917 non-null float64
hist_purchase_amount-median   201917 non-null float64
hist_purchase_amount-max      201917 non-null float64
hist_purchase_amount-min      201917 non-null float64
hist_purchase_amount-std      201917 non-null float64
hist_installments-sum         201917 non-null int64
hist_installments-median      201917 non-null float64
hist_installments-max         201917 non-null int64
hist_installments-min         201917 non-null int64
hist_installments-std         201917 non-null float64
hist_purchase_date-ptp        201917 non-null float64
hist_month_lag-min            201917 non-null int64
hist_month_lag-max            201917 non-null int64
new_transactions_count        179986 non-null float64
new_authorized_flag-sum       179986 non-null float64
new_authorized_flag-mean      179986 non-null float64
new_merchant_id-nunique       179986 non-null float64
new_purchase_amount-sum       179986 non-null float64
new_purchase_amount-mean      179986 non-null float64
new_purchase_amount-max       179986 non-null float64
new_purchase_amount-min       179986 non-null float64
new_purchase_amount-std       153199 non-null float64
new_installments-sum          179986 non-null float64
new_installments-mean         179986 non-null float64
new_installments-max          179986 non-null float64
new_installments-min          179986 non-null float64
new_installments-std          153199 non-null float64
new_month_lag-min             179986 non-null float64
new_month_lag-max             179986 non-null float64
```

Fig 1.

Exploratory Visualization

The plot below shows how the transactions are distributed based on the usage of the credit card in the first month. This is helpful for predicting the trends in the data.

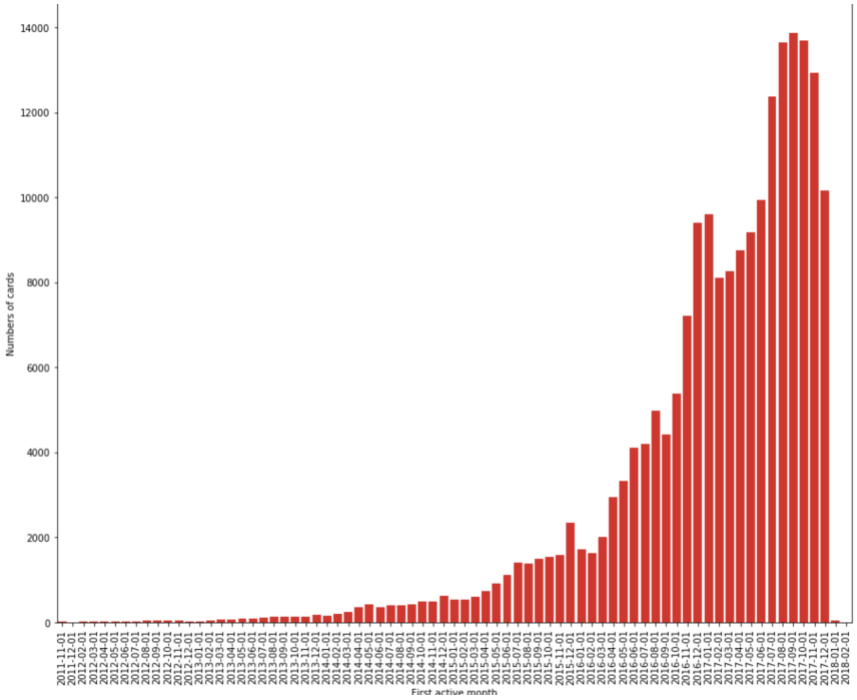
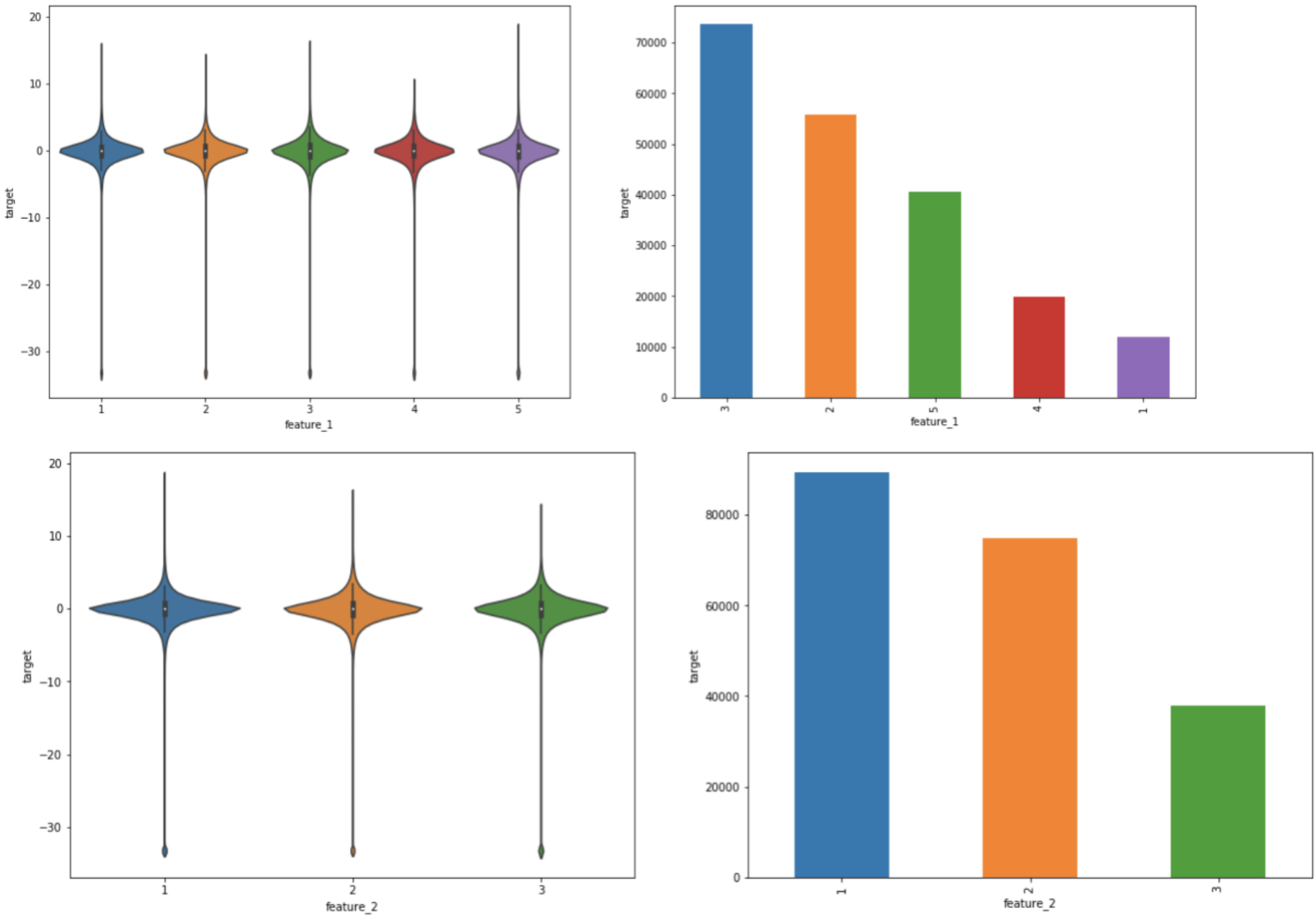
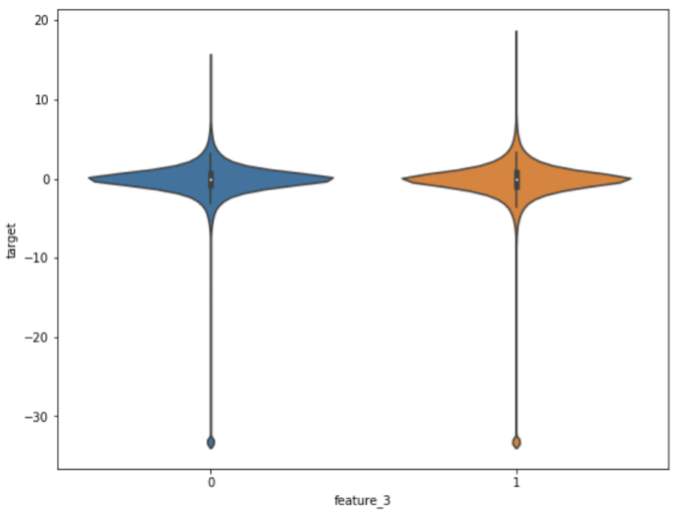
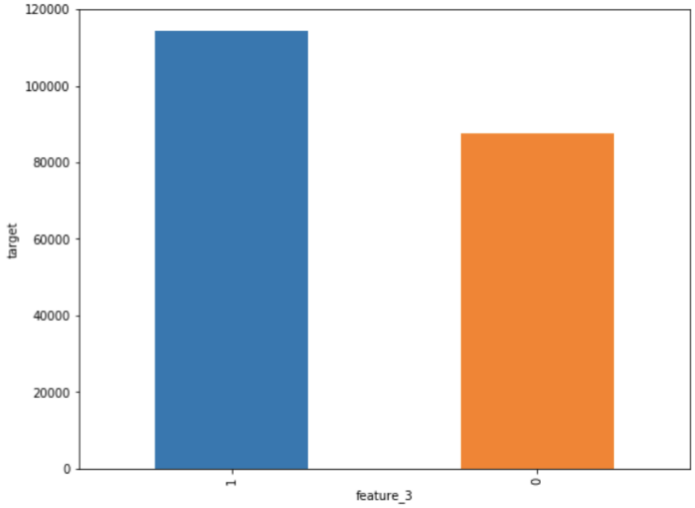


Fig 2.

It is visible to see that the most first purchases came in the years of 2016 and 2017. The trend started to increase in 2011 at a steady pace and continued until 2014. In 2015 the number of purchases increased at a much faster rate than before which could be indicative of the amount of trust that customers place in Elo's services.







Observations for the plots above show the features of the train.csv:

1. There are only a handful of different values for each of the three features.
2. All the values are discrete values.
3. All of the features are relatively evenly distributed.

This suggests these features are categorical and have been label encoded. This analysis could also indicate that these features aren't good at predicting target values. I will likely need to add new features and feature engineering to make better predictions.

The table below show descriptive statistics on the training set before the data was merged with the historical_transactions.csv and new_merchant_transactions.csv.

	feature_1	feature_2	feature_3	target
count	201917.000000	201917.000000	201917.000000	201917.000000
mean	3.105311	1.745410	0.565569	-0.393636
std	1.186160	0.751362	0.495683	3.850500
min	1.000000	1.000000	0.000000	-33.219281
25%	2.000000	1.000000	0.000000	-0.883110
50%	3.000000	2.000000	1.000000	-0.023437
75%	4.000000	2.000000	1.000000	0.765453
max	5.000000	3.000000	1.000000	17.965068

This table shows that between the original features in the training data there is not a lot of variance, which as previously mentioned could indicate the need for additional features.

	feature_1	feature_2	feature_3	target	year	month	elasped_time	historical_trans_count	hist_authorized_flag-sum
count	201917.000000	201917.000000	201917.000000	201917.000000	201917.000000	201917.000000	201917.000000	201917.0	201917.000000
mean	3.105311	1.745410	0.565569	-0.393636	2016.509298	7.378745	381.978981	1.0	81.558982
std	1.186160	0.751362	0.495683	3.850500	0.788199	3.340718	293.710176	0.0	99.243357
min	1.000000	1.000000	0.000000	-33.219281	2011.000000	1.000000	0.000000	1.0	2.000000
25%	2.000000	1.000000	0.000000	-0.883110	2016.000000	5.000000	153.000000	1.0	23.000000
50%	3.000000	2.000000	1.000000	-0.023437	2017.000000	8.000000	306.000000	1.0	48.000000
75%	4.000000	2.000000	1.000000	0.765453	2017.000000	10.000000	488.000000	1.0	100.000000
max	5.000000	3.000000	1.000000	17.965068	2018.000000	12.000000	2284.000000	1.0	2537.000000

This table shows that 'elasped_time' appears to have a lot more variance than the original features, which could be indicative that this is a good predictor.

Algorithms and Techniques

[8] This project is an example of a regression because a regression analysis estimates the relationship between two or more variables. Regression analysis also allows us to compare the effects of variables measured on different scales such as the effect of price changes and the number of promotional activities. These benefits help market researchers/ data analysts/ data scientists to eliminate and evaluate the best set of variables to be used for building predictive models. The dataset that is used in this project is a perfect an example of evaluating the best set of variables to use to build a predictive model of customer loyalty.

There are various kinds of regression techniques available to make predictions. A Decision Tree is a robust and transparent Machine Learning model. The tree starts with a single node and then branches out, with a decision being made at every branch. As decisions trees are so robust, imagine if you could combine multiple decision trees together to achieve a better prediction for any given problem. These models are called ensemble models.

Ensemble

[9] In statistics and machine learning, ensemble algorithms use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

Supervised learning algorithms are most commonly described as performing the task of searching through a hypothesis space to find a suitable hypothesis that will make good predictions with a particular problem. Ensemble algorithms combine multiple hypotheses to form a (hopefully) better hypothesis. Evaluating the predictions of an ensemble algorithm typically requires more computation than evaluating the predictions of a single model.

Several examples of ensemble techniques that will be used in this project are the following:

Light GBM

[10] The first model is called the Light GBM which is an example of a gradient boosting framework that uses a tree-based learning algorithm. The main difference from other tree-based algorithms is that the Light GBM grows trees vertically while other algorithms grow trees horizontally meaning that the Light GBM grows tree **leaf-wise** while other algorithms grow level-wise. When growing the same leaf, Leaf-wise algorithms can reduce more loss than a level-wise algorithm.

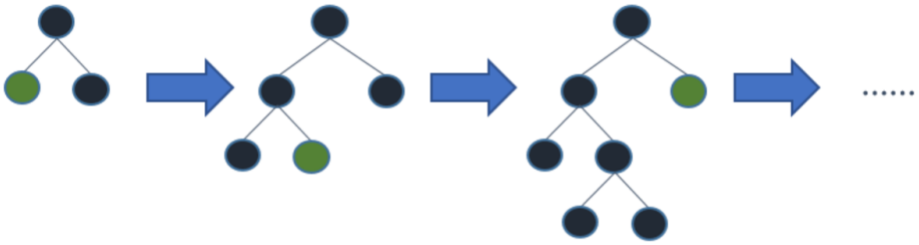


Fig 3.

As the size of data increases day by day, it is becoming difficult for traditional data science algorithms to give faster results. Light GBM can handle the large size of data and takes lower memory to run. Two other reasons why Light GBM is popular is because it supports GPU learning and it also focuses on the accuracy of results. Light GBM is sensitive to overfitting and can easily overfit small datasets.

XGBoost

[11] The second model is the XGBoost. The XGBoost model is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. The implementation of the algorithm was engineered for the efficiency of compute time and memory resources. The design goal was to make the best use of available data to train the model. XGBoost also offers several advanced features for model tuning, it is robust enough to support fine-tuning in addition to regularization parameters. The algorithm includes the handling of missing values, and the ability to fit and boost on new data added to a trained model.

Ridge Regression

[12] Ridge regression is a technique for analyzing multiple regression data that suffers from multicollinearity. Multicollinearity is a phenomenon in which one feature variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy.

I will be using 5-fold cross-validation. This will split our dataset into 5 parts, train on 4 and test on 1 part and repeat for all combinations of train-test split. Also, I am using the metric of RMSE to evaluate models. This is the square root of the mean square error, which is obtained by taking the number of actual values in the target variable subtracting that by the number of predicted values in the target variable.

Benchmark

The table below highlights performances of various models with the metrics of the root means square error for evaluation of the performance of the model. Each model has the default parameters. I will be using a simple linear regression as a benchmark, to use as a baseline score for my dataset when comparing how well my finalized model performed. Also, since this is a Kaggle competition another good benchmark would be the best Kaggle score for the test set, which comes in at the time of this project 3.713. Using this score I will try to beat the benchmark with hyperparameter tuning.

Algorithms	RMSE
Ridge regression	3.819957469516386
LightGBM (LGBM)	3.7100273584655095
XGBoost (XGB)	3.7503941956232656
Linear Regression (Benchmark)	3.8058572218733397

Methodology

Data Preprocessing

The preprocessing done in "Prepare data" section of the notebook consists of the following steps:

1. Check the quality of the data given and perform data cleaning.
2. Prepare the data by splitting the features and target columns.
3. Splitting the datasets into a training set and a validation set. This was already done by Kaggle which provided the dataset.

Verifying the quality of the data is good, because if there are any missing values, that could greatly affect the outcomes of the model during training. Several things that are done during the data cleaning process are checking to see if there are any non-numeric columns that need to be converted into numeric values. Columns that have more than two values are known as categorical variables, the recommended way to handle such columns is to create as many columns as needed of possible values and assign a 1 to one of the values and 0 to all other possible values. These generate columns that are sometimes called dummy variables, and which I will use the pandas.get_dummies() function to perform this transformation.

Several data preprocessing steps were followed: preprocessing feature columns, identifying feature and target columns, data cleaning and creating training and validation data splits. For more details please reference the attached Jupyter Notebook.

Implementation

[13] The project follows the typical predictive analytics hierarchy as shown in Fig 4:

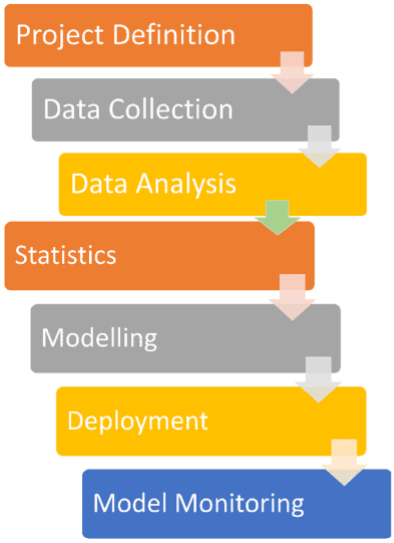


Fig 4.

Define project: Define the project outcomes, identify the datasets that are going to be used.

Data collection: Data mining for predictive analytics prepares the data from multiple sources for analysis.

Data analysis: Data analysis is the process of inspecting, cleaning and modelling data with the objective of discovering any useful insights or information.

Statistics: Statistical analysis enables to validate the assumptions, hypothesis and test them using standard statistical models.

Modelling: Predictive modelling provides the ability to automatically create accurate predictive models about the future.

Following the direction of the arrow, as shown in Fig 4, with the dataset I chose, I first performed exploratory analysis by seeing the distribution of the features that appear to be relevant for the problem I plan to solve. I also did a check of correlations using a heatmap. I then performed data cleaning steps to identify columns that might have any missing values and decided to either fill them or completely drop the columns from the analysis. I preprocessed several feature columns that have the values like 'yes' or 'no' and converted them to binary values such as 1 or 0. I converted the categorical features into dummy variables.

As this is a Kaggle competition the data was already split for me, I then proceeded to the modelling phase. I used mostly ensemble algorithms such as XGBoost and LightGBM since these models are well known to perform well with larger datasets, structured and tabular data. This proved true with this dataset, with the LightGBM having the lowest RMSE error, with the XGBoost close behind. I then preceded to perform parameter tuning of the LightGBM model's hyperparameters in batches using GridSearchCV. This way I could keep track of the best outcomes of each set of parameters and apply it to the next batch.

I faced several challenges during this project they were the following:

The first major challenge that I ran into was that I was unclear on what to do with the multiple data sets that made up the competition. I overcame this particular challenge by looking at the problem and at first noticed that all the datasets had a common column of 'card_id'. This problem was very similar to a relational databases primary and foreign keys. I also researched using different sites such as Stackoverflow and previous Kaggle competitions to work through how to aggregate data and merge them together.

The second major challenge that I faced during this project was that because the training and test sets were provided by Kaggle with the goal of predicting the target column on the test.csv, I had to implement cross-validation to create predictions. This was not something I had encountered before. During previous projects, I had to split the data into a training set and test set, then train the model on the training set and test on the test set. I overcame this by using cross validation, by splitting the training set into 5 folds with 4 parts of each fold used for training and 1 part for validation and predictions.

Refinement

[14] I performed parameter tuning of the hyperparameters of the LightGBM model. The list of the parameters tuned are shown in the table below:

Parameter	Description	Values Tested	Best Value
num_leaves	Maximum tree leaves for base learners.	[50, 65, 75, 85, 130]	75
max_depth	Maximum tree depth for base learners	[4, 8, 12, 15, 20]	8
min_child_sample	Minimum number of data needed in a child (leaf).	[20, 25, 50, 100, 150]	20
min_data_in_leaf	Minimum data in each individual leaf.	[10, 25, 50, 75, 100, 150]	150
learning_rate	Shrinkage rate	[0.005, 0.01, 0.1, 1]	0.1
subsample	Subsample ratio of the training instance	[0.5, 0.7, 1]	0.5
feature_fraction	LightGBM will random select part of features on each iteration	[0.7, 1.0]	1.0
bagging_fraction	Like feature_fraction, but this will random select part of data	[0.7, 1.]	0.7
lambda_l1	L1 regularization	[0, 6]	6

Results

Model Evaluation and Validation

I applied and compared the linear regression model with the rest of the models. The metric that I used is calculated using sklearn wrapper, so the results can be trusted for the model performance. My end goal was to have a tuned model that could beat the untuned benchmark which it did by a significant margin. The solution described below is satisfactory to my initial expectations. I generated a final model with the list of tuned parameters mentioned above. The output of this model came just about 0.1 lower RMSE score. Code snippet of the final model shown below:

```
# Printing the predictions and RMSE error of the model
cv_results_lgbl = np.sqrt(mean_squared_error(oof_lgbl, y_train))
print(cv_results_lgbl)
```

```
Early stopping, best iteration is:
[101]   training's rmse: 3.56252        valid_1's rmse: 3.68167
Fold 1 started at Wed Dec 12 14:52:05 2018
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[95]    training's rmse: 3.55535        valid_1's rmse: 3.74546
Fold 2 started at Wed Dec 12 14:52:09 2018
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[82]    training's rmse: 3.5701 valid_1's rmse: 3.70942
Fold 3 started at Wed Dec 12 14:52:13 2018
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[72]    training's rmse: 3.57342        valid_1's rmse: 3.7358
Fold 4 started at Wed Dec 12 14:52:17 2018
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[88]    training's rmse: 3.5872 valid_1's rmse: 3.65912
3.706435816762611
```

Robustness check:

The best model is trained on reduced features space having only 5 highest ranked features in terms of importance instead of 20 features.

RMSE of untuned model = 3.706435816762611
RMSE of reduced features model = 3.7849583283223325
Difference = 0.078

Therefore, we can see that even though the feature space is reduced drastically (by more than 75%), the relati ve loss in performance is not overly significant.

Justification

There is room to improve on the final results. The tuned final model made a significant improvement over the untuned model. There are more ways to improve the root mean square error, particularly by selecting a subset of features using the ranking obtained from observing feature importance ranking and then performing the same exercise as described below. However, this is out of the scope of this report for now.

	Benchmark Model (Untuned Linear Regression Model)	**Final Model (Tuned LightGBM Model)**
Root Mean Square Error (RMSE)	3.8058572218733397	3.706435816762611

Conclusion

Free-form Visulization

The importance of each feature was visualized with the following plot:

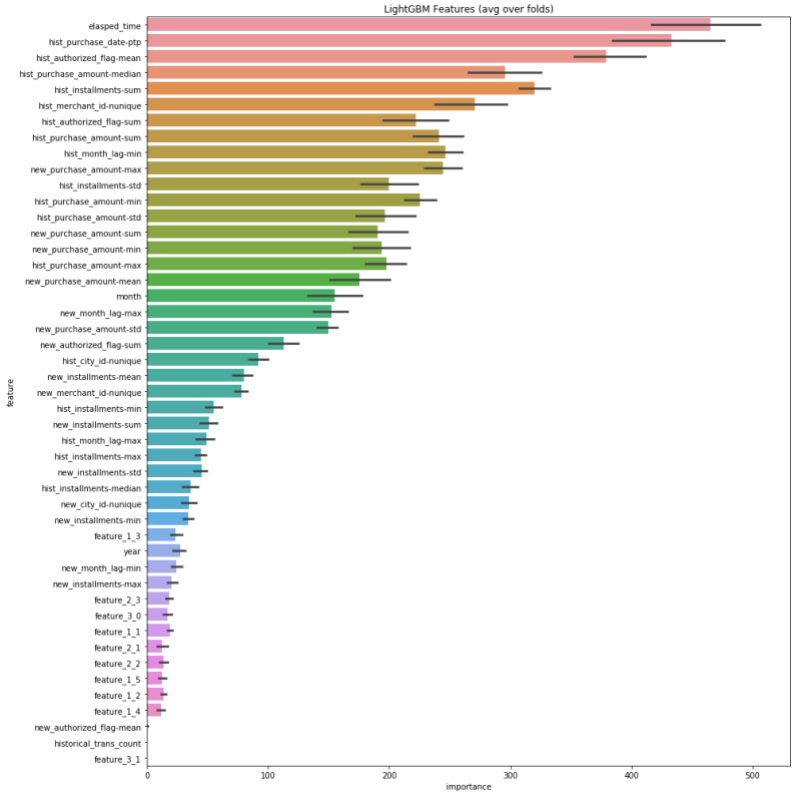


Fig.5 Important Feature ranking

Reflection

The most important and time-consuming part of the problem was data cleaning and processing. Once the data was prepared and ready, the next challenge was to pick an algorithm that could best suit the problem I chose to solve. With experience and prior knowledge and also the outcome of RMSE score on the training data, I observed that the LightGBM performed the best out of all the models tested.

If I look at the important predictors in their order of importance, as per the tree shown in Fig. 5, they are elapsed time, hist_purchase_date-ptp, hist_installament-sum and so on. The purchase date is very important because, when a person makes a purchase this could depend on when they might receive their next paycheck or certain times of the year where there could be very big savings. The installments sum also plays a big role because how much a person has to pay in individual installments makes a big impact on how much they might spend. Based on the data analyzed and the domain. Here are a few suggestions for improving customer loyalty:

- Companies like Elo should have clear promotions and they should put the value proposition up front – offering something free or promoting a bundle service at a discounted price increases sales and customer loyalty. The value proposition should be right up front, with a clear value to the individual.
- Elo could start a rewards program where if customers spend more, they accumulate more rewards that would be used at participating merchants.

Improvements

One of the improvements I could do is to perform more fine-tuning to achieve a lower RMSE to improve overall predictions performance of the model. The other improvement could be to work with a subset of features that are high in variable importance.

References:

[1] https://www.kaggle.com/c/elo-merchant-category-recommendation

[2] https://en.wikipedia.org/wiki/Personalized_marketing

[3] http://businessoverbroadway.com/2018/03/19/using-predictive-analytics-and-artificial-intelligence-to-improve-customer-loyalty/

[4] https://techspective.net/2018/05/16/how-leading-companies-use-ai-for-customer-retention/

[5] https://www.google.ca/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwilmrSotZnfAhVM5YMKHdhvCKoQFjABegQIBxAC&url=https%3A%2F%2Fwww.u-cursos.cl%2Fingenieria%2F2010%2F2%2FIN71K%2F1%2Fmaterial_docente%2Fbajar%3Fid_material%3D307341&usg=AOvVaw2hkrOsW5KIRl0Ob6mVkn60

[6] http://cs229.stanford.edu/proj2014/Ashish%20Gandhe,Restaurant%20Recommendation%20System.pdf

[7] https://towardsdatascience.com/art-of-choosing-metrics-in-supervised-models-part-1-f960ae46902e

[8] https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

[9] https://en.wikipedia.org/wiki/Ensemble_learning

[10] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

[11] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[12] https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf

[13] https://en.wikipedia.org/wiki/Predictive_analytics

[14] https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

[15] https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[16] https://en.wikipedia.org/wiki/Probability_density_function

[17] https://datavizcatalogue.com/methods/violin_plot.html

[18] https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html