Elo Customer Loyalty
Milestone Report 2

Problem Statement

Elo is one of the top credit and debit card companies in Brazil. Currently they are aiming to introduce a new perk to users that will give them discounts to merchants in their area. Currently all users are being given the same promotion no matter the difference in users. Elo is seeking to improve the system by recommending merchants that users will be interested in.

The initial step to Elo's goal is for them to better understand customer loyalty. This information can be used to create specific marketing campaigns aimed at unique customers. This custom tailored marketing should be far more effective than Elo's current campaigns. Users will be more engaged with Elo and its services. Potentially ads can be tailored to unique market segments, based off the data, to increase ad-clicks and gain new clients.

Data

The data is available on the kaggle competition page. There are 4 csv files of use for the analysis. Training data, testing data, information on a customers historical transactions, as well as transactions on new merchants they've visited.

Majority of the potential features for this analysis are anonymous. The training data and testing data contain anonymous features 1, 2, 3, along with first active month, referring to the date the user registered their card with Elo.

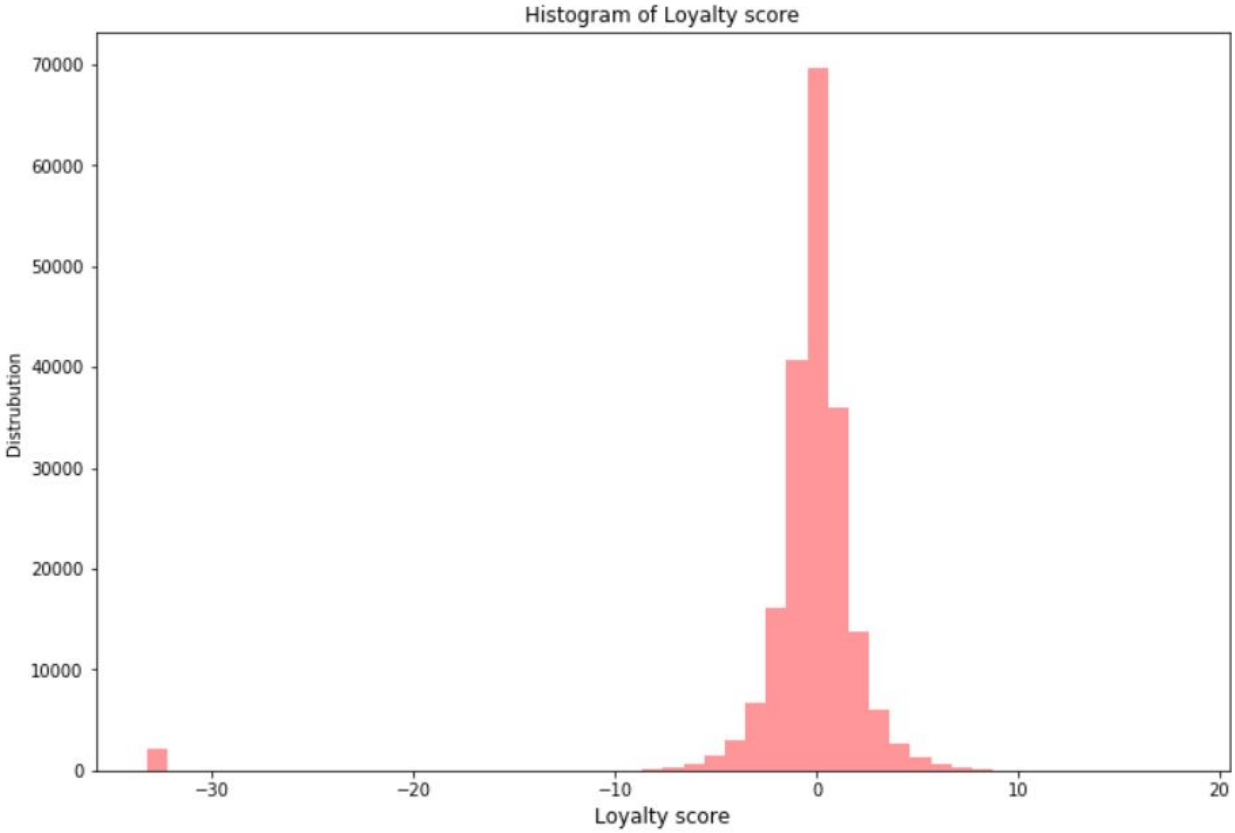
Data Wrangling / EDA

As this is a kaggle competition, the data was already in a very ready to use format with little wrangling required. I began by converting the first active month into datetime format, and adding additional columns showing only the day, month, year, and elapsed time from the end date of observations.

	first_active_month	card_id	feature_1	feature_2	feature_3	target
0	2017-06	C_ID_92a2005557	5	2	1	-0.820283
1	2017-01	C_ID_3d0044924f	4	1	0	0.392913
2	2016-08	C_ID_d639edf6cd	2	2	0	0.688056
3	2017-09	C_ID_186d6a6901	4	3	0	0.142495
4	2017-11	C_ID_cdbd2c0db2	1	3	0	-0.159749

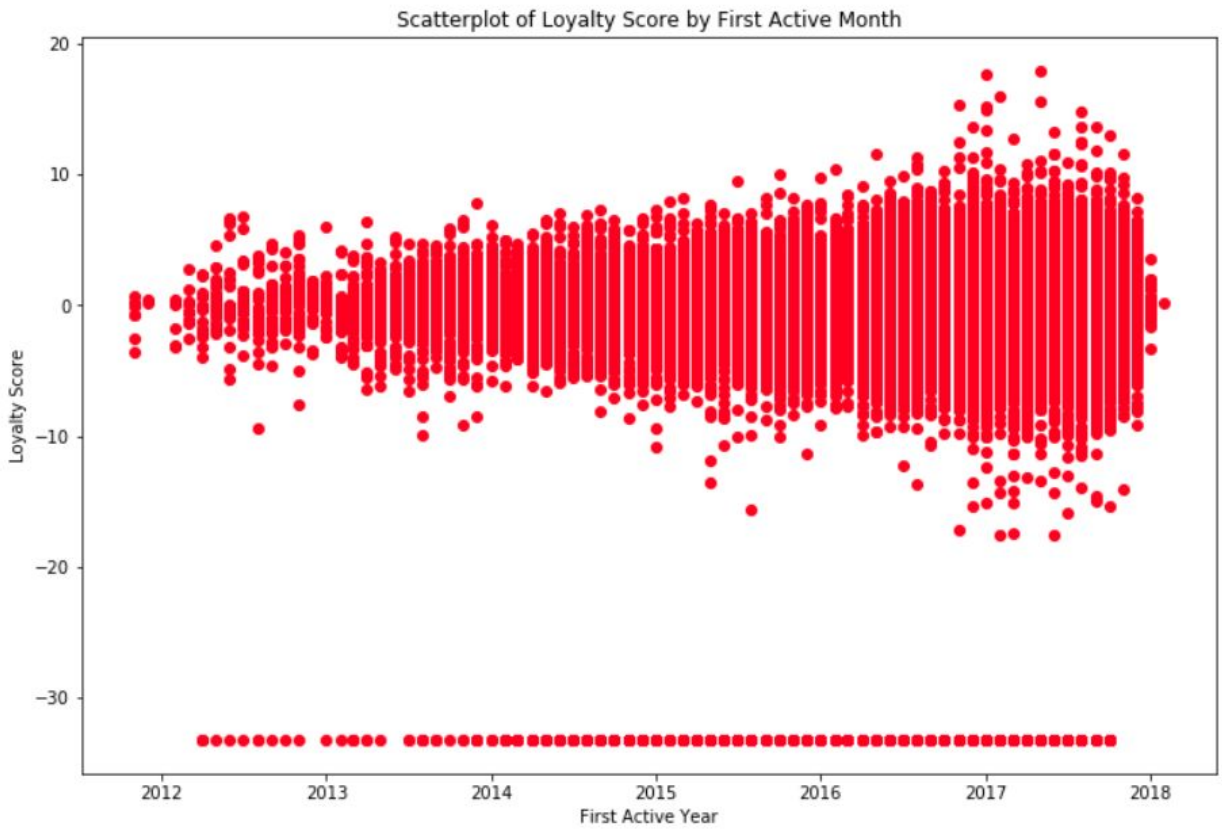
	first_active_month	card_id	feature_1	feature_2	feature_3	target	year	month	elapsed_time
0	2017-06-01	C_ID_92a2005557	5	2	1	-0.820283	2017	6	245
1	2017-01-01	C_ID_3d0044924f	4	1	0	0.392913	2017	1	396

The target is the loyalty score of the customers, which appears to be in logarithmic form. First observation shows a relatively large outlier spike at -33.



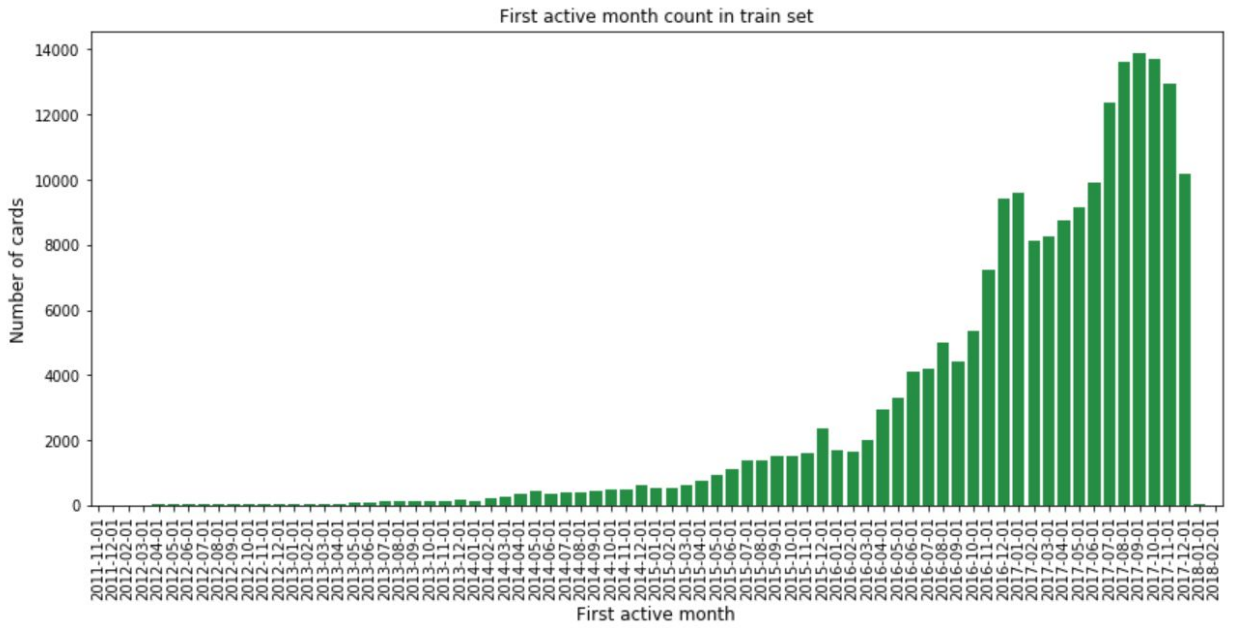
Using the pandas value_counts method on values below -30 it shows that there are 2,207 observations of -33, and 199,710 observations of those higher (ranging from -8 to 10). So 1% of the observations are an outlier at -33. What is causing this spike I am unsure of at this moment. It might be worth excluding these values, but at 1% these are a pretty high count, and might pertain to some important correlation.

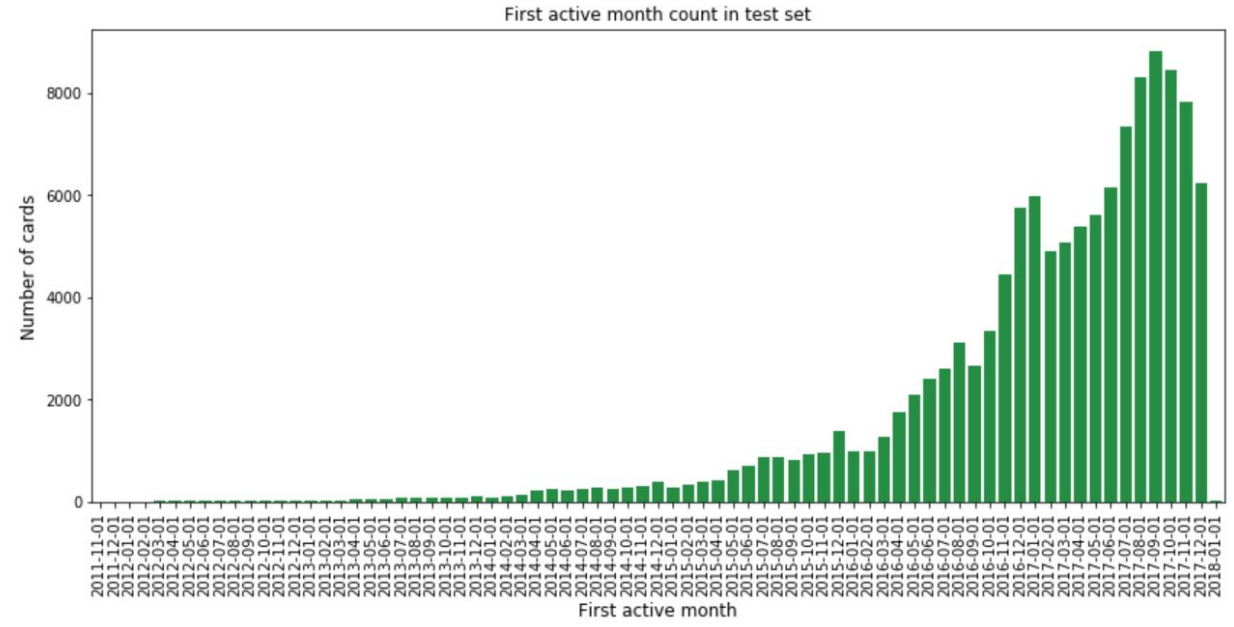
My next step was to observe the change in loyalty score given the first active month of the user.



I can see that the -33 observation remains constant throughout. But the newer the user is, the more spread can be seen from their target score, including some very high values.

Next I wanted to see if the distribution of users relating to first active month is similar between the training and test data.





Luckily we see a nearly identical distribution between both sets.

Seeing the limited features for this analysis, I want to see if I'll be able to utilize the historical transactions and new transactions datasets to gain new insights on user behavior to add more features for my model. Both files contain the following features, many of which are once again anonymous.

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

The authorized flag, category 1, and category 3 columns have string values, so using Pandas map method I convert them to numerical values to use for a regression model. Then i concatenate in both the files together with a key on each file so if I wanted I could still distinguish between which observations were from new merchant visits and which were from previously frequented visits.

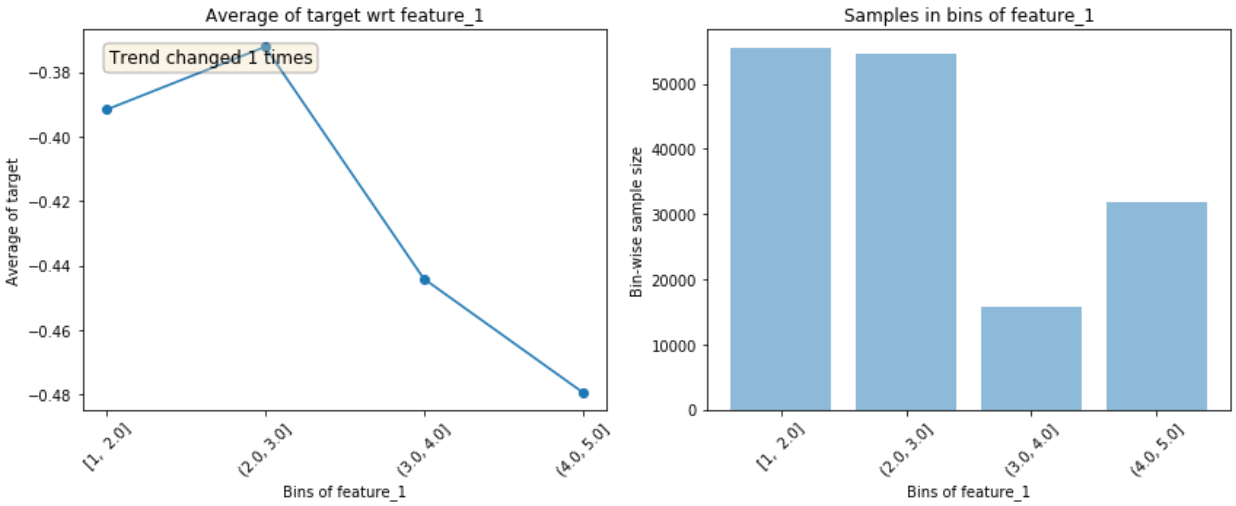
Lastly I create a function inspired from other users kernels to do a large group by function, that would get aggregate stats on many of the columns such as mean, std, median for the numeric features, and unique values for the id columns. Then I can merge these features with my training and test set so I can see their prediction power on the loyalty score.

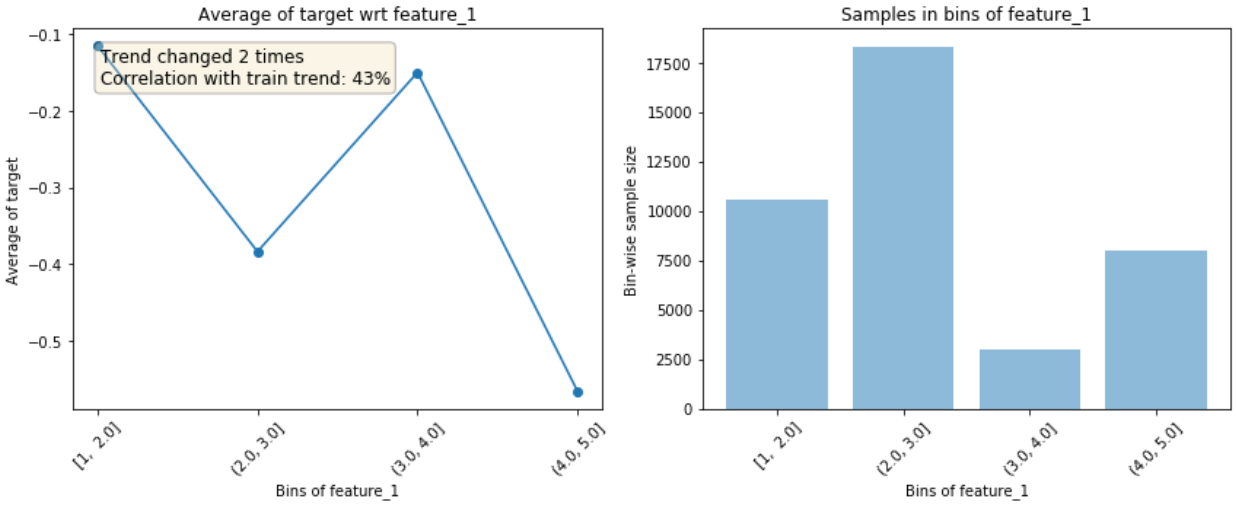
```
def aggregate_historical_transactions(history):

    history.loc[:, 'purchase_date'] = pd.DatetimeIndex(history['purchase_date']).\
                                      astype(np.int64) * 1e-9

    agg_func = {
        'authorized_flag': ['sum', 'mean'],
        'merchant_id': ['nunique'],
        'city_id': ['nunique'],
        'purchase_amount': ['sum', 'median', 'max', 'min', 'std'],
        'installments': ['sum', 'median', 'max', 'min', 'std'],
        'purchase_date': [np.ptp],
        'month_lag': ['min', 'max'],
        'category_1': ['sum', 'mean'],
        'category_2': ['sum', 'mean'],
        'category_3': ['sum', 'mean']

        }

    agg_history = history.groupby(['card_id']).agg(agg_func)
    agg_history.columns = ['hist_' + '_'.join(col).strip()
                           for col in agg_history.columns.values]
    agg_history.reset_index(inplace=True)

    df = (history.groupby('card_id')
          .size()
          .reset_index(name='hist_transactions_count'))

    agg_history = pd.merge(df, agg_history, on='card_id', how='left')

    return agg_history
```

Featexp is a module I discovered that helps accelerate the EDA and feature eng process. Using this package I can efficiently observe a histogram of each feature, and how the change of value in each feature affects the target value. Along with this, it gives a trend correlation between the

training set, and a validation set. This had the goal of helping to find which features impact on the target value are holding true even on new data, and which ones are inconsistent.

The validation set consists of observations where the first active month were users activated for less than 100 days, and more than 800 days. The remaining training set were those in between those values. The validation set was approximately 20% of the training with 39,837 rows.





Here for example I can easily see the distribution of feature 1 in both the training and validation set, their impact on the target value, and how that that impact differed on the validation set.

```
stats = stats.sort_values(by='Trend_correlation', ascending=False)
stats
```

	Feature	Trend_changes	Trend_changes_test	Trend_correlation
2	feature_3	0	0	1.000000
3	year	0	0	1.000000
1	feature_2	1	1	0.996557
23	hist_month_lag_max	1	1	0.974305
8	hist_authorized_flag_mean	4	4	0.914882
22	hist_month_lag_min	1	3	0.882759
27	hist_category_2_mean	4	3	0.862556
28	hist_category_3_sum	3	4	0.860611

Using the stats feature of featexp, I can easily observe this information in a pandas dataframe.

Same as my previous capstone I plan to use variance inflation factor to eliminate features that are causing multicollinearity. Through iteration I end up with these remaining features because they have a VIF score of 5 or below

	VIF Factor	features
0	2.2	feature_3
1	4.3	feature_2
2	2.6	hist_month_lag_max
3	3.1	hist_category_2_mean
4	3.2	hist_merchant_id_nunique
5	2.3	hist_installments_sum
6	2.1	hist_installments_min
7	1.0	hist_purchase_amount_std

Prediction with XGB

Now that I have the data in a clean format and have conducted the necessary feature engineering, I'm able to push the data through an Extreme Gradient Boosting model. I've chosen XGB because it has become a standard used in data science as well as Kaggle competitions due to its boosting algorithm giving consistently superior results.

I created a function that sets up parameters, converts the features and target data into a Dmatrix, improving the performance and efficiency of the model. Using XGB's cross validation method on the data and gather the final result after 70 rounds and observe the root mean squared error on the test set as the metric for comparison.

```
def get_cv_rsme(Features, Target):

    params = {"objective":"reg:linear",'colsample_bytree': 0.3,'learning_rate': 0.1,
                'max_depth': 5, 'alpha': 10}

    data_dmatrix = xgb.DMatrix(data=Features,label=Target) #Improves the performance and efficiency of the model

    cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
                        num_boost_round=70,early_stopping_rounds=20,metrics="rmse", as_pandas=True, seed=123)

    return(cv_results.tail(1))
```

After 5 varying attemps using XGB and different methods to select features I compared my models.

1. Attempt 1
    a. Limit the features to those with the top 15 trend correlation
    b. Limit those remaining to those with low VIF scores
    c. CV RMSE: 3.81
    d. RMSE test: 3.91
2. Attempt 2
    a. Use all applicable features available
    b. CV RMSE: 3.75
    c. RSME test: N/A (Model overfit due to too many features)
3. Attempt 3
    a. Solely using features with VIF score of 5 or below
    b. CV RMSE : 3.80
    c. RMSE Test : 3.91
4. Attempt 4
    a. Scale the data
    b. Apply PCA
    c. Determine Intrinsic Value (# of pca features needed to approximate data). Result: 4
    d. Fit and transform the training and test data to 4 components (adding more components overfits the model)
    e. CV RMSE: 3.81
    f. RSME Test: 4.22
5. Attempt 5:
    a. Simply tuning the model from attempt 2 by experimenting with the hyperparameters.
    b. First attempt was using Gridsearchcv, but took too long.
    c. Used trial and error instead, Randomsearchcv could have also been an option.
    d. CV RMSE : 3.74
    e. RMSE Test: N/A (Model overfit)

Conclusion

Including all the features created by the groupby function, aggregrate_historical_transactions, did improve the RMSE score on the test set. Interestingly limiting the features through various feature engineering techniques resulted in a worsened RMSE score on the cross validation set, but indefinitely had caused overfitting, as proved when I submitted these model predictions to the kaggle competition and I received a low score. So they wouldn't be a good method to use for real world applications. A more accurate model during training is not always the better model for production.

In my analysis I've come to the conclusion that the data available predicting the loyalty score for the Brazilian payment company is insufficient to make any accurate predictions for their customer base. My RMSE score of 3.745 on the test set, or 3.921 on the unseen test target values, would not be helpful due to how inaccurate the model is with the available features. The several attempts and optimization of hyperparameters only helped to boost my placement on Kaggle, but the range of my score, as well as others that can be seen on the competition leaderboard would most likely be inapplicable to a business setting.

The majority of the target values lie between -1 and 1, so if the prediction is off by 3.75 more or less, the prediction is not something that can be utilized.

I assumed throughout the analysis that the -33 loyalty score was based off a high amount of flagged authorizations. The feature importance graph on attempt 2 shows the most important feature as hist_aurthoized_flag_mean (the mean of a users transactions that were properly authorized). This suggests the model is likely correctly predicting if a customers loyalty score is -33 or not, accurately. Hence the high RMSE score.

Future Endeavors

This analysis might not bring real value, and may be labeled not feasible, but thats okay! That is part of data science work. Sometimes you find you don't have what you need to accomplish an objective, and you need to go back to acquire more meaningful data. The hypothesis might be proved wrong, and that's part of the job in any science related field. In a business setting, domain knowledge and stakeholder input is a vital step to determining if you are asking the right questions. The data scientist needs to determine if that data is currently in our possession, and if not, investigate how it can be obtained, whether through surveys, web scraping or other means. Elo can change the question with more telling data, to still achieve the objective on segmenting the customer base.

Onward to the next challenge! :)