DAT 554 Final Project: Customer Loyalty Analyzer

Sheraz Ahmed

***Abstract*—The purpose of this project is to describe the process and outcome of developing algorithms to identify and serve the most relevant opportunities to individuals, by uncovering signal in customer loyalty. The outcome of this project will improve customers' lives and help reduce unwanted campaigns, to create the right experience for customers. Based on the outcome, Individuals will get recommendations served up, based on their personal preferences. The recommendation will come with an attached discount from their credit card provider for a local place around the corner. We obtained the dataset from Kaggle provided by the Elo. We have tried to solve the problem using various Machine Learning models and according to the results obtained, LGBM and polynomial regression are the best performing models on this data.**

I. INTRODUCTION

1) Background:

This paper is going to focus on helping understand customer loyalty using machine learning. Elo, a large Brazilian payment brand (focused on debit and credit cards), has built machine learning models to understand the most important aspects in their customers' lifecycle. However, there is a major limitation to their existing models. So far none of their models is specifically tailored individually. That means that Elo cannot deliver fully personalized brand recommendations to its customers, nor can it filter unwanted ones.
Our goal is to develop ML (machine learning) models that identify and prescribe the most relevant opportunities for individuals, by predicting customer loyalty based on various data features arising from customer data. This can improve Customer's payment experience and help reduce unwanted campaigns, to create the right experience for the customers.

2) The Problem Statement:

The Problem Statement for this project would be like: *How can machine learning be used to best predict customer loyalty using Elo's dataset from the "Merchant Category Recommendations"*

We will follow the below mentioned steps to solve the problem in this paper:

- Analyze, test and choose various feature selection and feature engineering possibilities for the given dataset.
- Create and test various ML models that could serve as predictors of customer loyalty based on inputted features.
- Analyze which ML models work best for the given dataset.

II. DATASET

METHODOLOGY

A. Given data

Data set provided by Elo consists of several .csv files: a data dictionary that describes the other files provided on Kaggle, a training set, a test set, historical transactions of each credit card, transactions at new merchants over a period of two months, and information on Elo's merchants. Mind that Elo mentions that the data is simulated and fictitious, but it could be the case that this simulation is based on real consumer data. The training and test set were provided with three

anonymized categorical features: features_1, feature_2 and feature_3 (see figure 1 for an example of three training data entry from the original csv file). These three categorical features were transformed into one-hot-encoded features.
The result of this is that for each category of a feature, there will be a new feature that is 1 if the feature is equal that category and otherwise 0 (figure 2). New features were made to better predict the loyalty score, which is given for the training set, but hidden for the test set. Since a couple of features, categorical and numerical are anonymized, it was hard to interpret why certain features are more important to the loyalty score.

first active month	card id	features 1, 2, 3			Target (loyalty score)
2017-06	C_ID_92a2005557	5	2	1	-0.82028
2017-01	C_ID_3d0044924f	4	1	0	0.392913
2016-08	C_ID_d639edf6cd	2	2	0	0.688056

Figure 1

card_id	feature1 is 1	feature 1 is 2	feature 1 is 3	feature 1 is 4	feature 1 is 5
C_ID_92a2005557	0	0	0	0	1
C_ID_3d0044924f	0	0	0	1	0
C_ID_d639edf6cd	0	1	0	0	0

Figure 2

B. Data preparation, feature engineering and feature Selection

It was quickly seen that new features should be made from the historical transactions. Most cariads have multiple transactions, which means that features can be created by transforming each column of historical transactions into a metric. The historical transactions were split up by authorized and unauthorized transactions, the historical transactions can be compared to the transactions made at new merchants. This is all done to create as much features as possible.
To combat the problem of the large historical transaction's dataset, the first and last occurrence of a certain current card_id were saved, since the transactions were grouped per card_id. This results in that a card_id's transactions could be more quickly located in the historical transactions, which cuts computation time from Pandas significantly.
There are different types of info given in the historical transactions, which are handled in a different way. Nominal categorical info with few categories was handled using softmax normalization, so for each category there will be a feature that can be expressed in the amount of times that category occurs divided by the total amount of transactions. Also, the amount of times a category occurred in the transactions of card_id was transformed into a feature. We only counted the number of unique categories over all the transactions of a specific card_id for nominal categorical info with a lot of categories. Ordinal categorical info on transactions was simply transformed into similarly ordered numbers for it to be useful for the machine learning models.
Numerical info on transactions was transformed into features by taking the following statistics: minimum, maximum, standard deviation, mean, and sum. This was also done for the ordinal categorical info that was transformed into numerical info.

All important features are shown in the figure 3. The 30 most important features are visible, and the importance is based on the information gain obtained from Light.
The polynomial regression model was trained on the 30 most important features, because otherwise memory errors would occur due to the large training set.

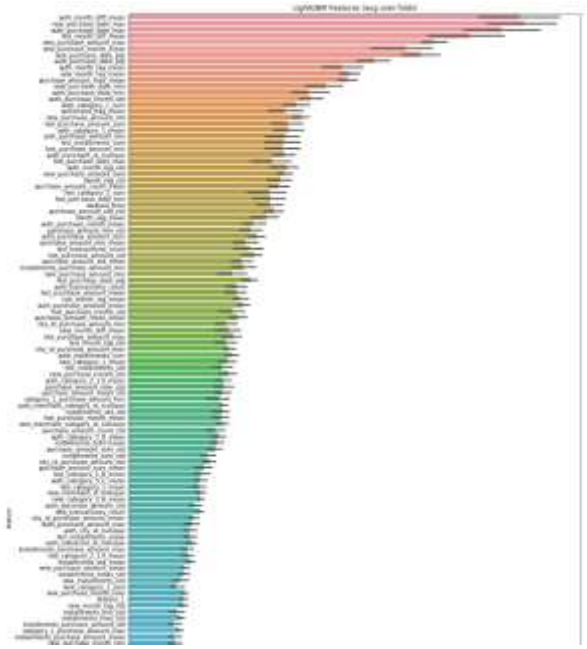

Figure 3

III. SOLUTION

In this section, we will implement various ML (Machine Learning) models. Our implementations used the Python programming language with Pandas, NumPy, Sclera, Keras, XGBoost and LightGBM libraries.

Support Vector Machines (SVMs)

SVMs are typically used for classification tasks, however, they can be and still are a choice for regression problems which is why they were chosen as a possible candidate for this project. They involve maximizing the minimum distance from the separating hyperplane to the nearest data point. They were also a good model candidate to try because they can have a good generalization ability and a flexible selection of kernels to allow non-linearity. We tried Support Vector Regression (SVR) with a Radial Basis Function (RBF) kernel in this project because the technique could be able to first transform the given data and then figure out how to best fit it. However, for our problem the performance of SVR was highly limited, to the degree that with some preliminary experiments, we decided that SVRs were not interesting or feasible direction to pursue.

Neural Networks

Neural networks were tried on the problem as they are not only capable of regression but can also do implicit feature selection. Neural networks have recently gained more popularity as a generic curve fitting technique that can perform reasonably well on regression problems.
Neural networks were also, but just like with SVRs; preliminary tests already showed that neural networks would not have enough performance on the given dataset and so they were not explored further.

Linear Regression

The simplest model to tackle this dataset is linear regression. Linear regression is fast, can easily be interpreted and doesn't have a lot of parameters that heavily impact its predictions. The advantage of that is that it can be used to quickly make predictions and get a feel of what the features in the training set are allowing the model to predict. A downside is that this model does not select the best features to be most important, if these are not uniformly or normally distributed, so skewed data can severely hinder linear regression in giving correct predictions. Another downside is that the model suffers from multicollinearity. Multicollinearity is the correlation between features, if this is high, the model will have worse predictions.
With the smaller feature set before the expansion to 166 features, we tried linear regression and it showed potential. The success of linear regression with early feature sets led us to add polynomial features up to degree three (all feature combinations up to cubed) and do polynomial regression. However, as the number of features increased, we began encountering memory errors and even regular linear regression model began overfitting. For example, with 27 features a degree 3 polynomial becomes over 4000 features which when coupled with the large dataset runs into memory issues. We decided to counter the memory issues by using degree 2 polynomial regression and reducing the number of features to the top 30 best performing feature by information gain identified by Light (figure 3).

Decision trees (LGBM)

Decision trees were also good candidates for our problem, because: they can handle feature interactions, they can handle linearly inseparable data and they can provide a good performance at a low computational cost. Gradient boosting can be an effective prediction approach for regression tasks, like in our project. They also show that boosting further adds to model performance by incorporating automatic feature selection during the regression task.

Therefore, we tried two popular boosting algorithms: LightGBM and XGBoost. LGBM was tried both with boosting type of 'gbdt', traditional Gradient Boosting Decision Tree and of 'rf', Random Forest. Regarding LGBM versus XGB, LGBM always outperformed XGB on the given dataset (where both were fed the same features and comparable hyperparameters). So, LGBM was the algorithm that was taken further.

We tried LGBM with traditional Gradient Boosting Decision Tree ('gbdt' boosting type) and with Random Forest ('rf' boosting type). We used 33 as the hyperparameter value for maximum tree leaves for base learners and used 7 as the maximum tree depth for base learners (to avoid overfitting).

IV. Evaluation Metrics

This project uses a Root Mean Square Error (RMSE) evaluation for its competitors. This metric is representative for about the average error between a prediction and a given label.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where ŷ is the predicted loyalty score for each card_id, and y is the actual loyalty score assigned to a card_id. Figure 4 shows the training and testing RMSE scores for the top performing models from the above-discussed methodology. The training RMSE corresponds to the models' performance on the entire training set.

Model	Features Used	Training RMSE	Test RMSE
LightGBM (with traditional Gradient Boosting Decision Tree)	All 166 features	3.663	3.710
LightGBM (with Random Forest booster)	All 166 features	3.664	3.713
Linear Regression	All 166 features (log transformed)	3.744	3.905
Polynomial Regression (degree two)	Top 30 information gain features (figure 2)	3.808	3.886

Figure 4

These RMSE values can be put into context by looking at the statistics of the dataset. The RMSE is quite close to the standard deviation, meaning that our model results are quite weak.

V. Conclusion

We have concluded that for the given dataset, two approaches seem suitable: linear regression and decision trees (specifically, LightGBM). This can at first glance be concluded from the results, where these models have a lower RMSE than other models.

Comparing Decision trees to the linear regression, decision trees might outperform linear regression, since these can better take into the complexity of the data, like its outliers. Decision trees are also fast in their predictions, while higher order linear regression poses tougher computations, especially if there are a lot of features in the training set. This also led to memory errors.

Neural networks and SVMs clearly do not perform well. They do not seem suitable for the large dataset with a lack of features for the neural network to accurately predict and too much features for SVMs to train properly.

REFERENCES

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

[2] https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

[3] https://stackoverflow.com/questions/58328708/feature-engineering-using-python

[4] https://towardsdatascience.com/machine-learning-polynomial-regression-with-python-5328e4e8a386

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

[6] Aslam, Suhaib & Heijden, Guido & Collins, Harry. (2019). Predicting Customer Loyalty Using Various Regression Models.

[7] https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

[8] https://www.kdnuggets.com/2013/03/too-slow-or-out-memory-problems-machine-learning-data-mining.html