Elo Merchant Category Recommendation

Yingshuang Yang
DSCI 503

***Abstract*—This report presents a machine learning approach for predicting customer loyalty for Elo, one of the largest payment brands in the world. The objective of this project is to enhance Elo's category recommendation system by identifying loyal customers and offering targeted promotions. Using a combination of historical and new merchant transactions, the dataset was enriched and preprocessed to overcome challenges related to data size and quality. Various models, including Linear Regression, SVR, Random Forest, XGBoost, and Artificial Neural Networks, were evaluated using RMSE as the primary performance metric. Among these, XGBoost showed the best balance of accuracy and computational efficiency, achieving a competitive Test RMSE of 3.734. The report also addresses ethical considerations, such as a feedback loop in predictive task. Future work will explore advanced neural network techniques and specialized model training to further improve prediction accuracy.**

I. Introduction

Elo is one of the biggest payment brands in the world. Elo wants to optimize its category recommendation system by predicting customer loyalty [1] as studies show that the cost of appealing to new customers is 5 times higher than retaining existing customers [2]. In addition, predictive analytics using techniques such as machine learning can help companies anticipate customer needs, provide personalized services, and accurately conduct promotions at the right time [3]. This kind of analytics not only identifies customer loyalty in advance but also effectively reduces the maintenance and promotion costs of enterprises and meets Elo's current business needs. The aim of this project is to assist Elo Company in predicting customer loyalty scores. By identifying loyal customers and offering them targeted promotions, the company seeks to strengthen customer loyalty and create a virtuous cycle of engagement and retention.

II. Data Collection

In this project, the data was collected from Kaggle competition. It contains 6 files in total.

A. Data source

The data can be directly downloaded through this site. The link is Elo Merchant Category Recommendation Data.

B. Tools and Techniques

I used Python's pandas library to load and handle each dataset. Additionally, I used the gc library to clean up memory, as the historical transactions file is large, and keeping it in memory could cause a shutdown.

C. Data description

- `Data Dictionary.xlsx`: This file contains each data of csv files description.
- `train.csv and test.csv`: These files contain information for each card, including the first active month, Card IDs, and three other features. The difference between the two files is that the training data includes target columns, which the model needs to predict for the test file.
  The columns description of the train data are shown in Figure 1.

	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

Fig. 1. The columns of the train data

- `historical_transactions.csv and new_merchant_transactions.csv`: They contain information about transactions for each card. The historical data only contains transactions from the past 3 months in merchants where users have previously made purchases. In contrast, the new data contains transactions from the past 2 months at merchants where users did not make purchases in the previous 3 months or even earlier. The columns description of the transaction data are shown in Figure 2.

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

Fig. 2. The columns of the transaction data

- `merchants.csv`: This file contains data on all merchants involved in the transactions. [1]
  The columns description of the transaction data are shown in Figure 3.

Columns	Description
merchant_id	Unique merchant identifier
merchant_group_id	Merchant group (anonymized)
merchant_category_id	Unique identifier for merchant category (anonymized)
subsector_id	Merchant category group (anonymized)
numerical_1	anonymized measure
numerical_2	anonymized measure
category_1	anonymized category
most_recent_sales_range	Range of revenue (monetary units) in last active month --> A > B > C > D > E
most_recent_purchases_range	Range of quantity of transactions in last active month --> A > B > C > D > E
avg_sales_lag3	Monthly average of revenue in last 3 months divided by revenue in last active month
avg_purchases_lag3	Monthly average of transactions in last 3 months divided by transactions in last active month
active_months_lag3	Quantity of active months within last 3 months
avg_sales_lag6	Monthly average of revenue in last 6 months divided by revenue in last active month
avg_purchases_lag6	Monthly average of transactions in last 6 months divided by transactions in last active month
active_months_lag6	Quantity of active months within last 6 months
avg_sales_lag12	Monthly average of revenue in last 12 months divided by revenue in last active month
avg_purchases_lag12	Monthly average of transactions in last 12 months divided by transactions in last active month
active_months_lag12	Quantity of active months within last 12 months
category_4	anonymized category
city_id	City identifier (anonymized)
state_id	State identifier (anonymized)
category_2	anonymized category

Fig. 3. The columns of the merchant data

D. Challenges Encountered

The historical_transactions.csv file is too large, with 2.85 GB and almost 30 million rows, making it impossible to load into a dataframe using pandas directly. To handle this, I combine the use of chunksize and dtype to read the data in chunks and optimize data types for each chunk. After processing each chunk, I then combine them. This approach significantly decreases memory usage and allows efficient processing of large datasets.

III. Exploratory Data Analysis

A. Training and testing data analysis

The training dataset consists of 201,917 records, while the testing dataset contains 123,623 records. The test dataset lacks a target column because the loyalty scores need to be predicted. Additionally, there are only 4 features available, which is clearly insufficient for building a robust model to predict loyalty accurately.The training and testing datasets exhibit similar distributions, as evidenced by comparable mean values, standard deviations, and maximum values across each column. And it is evident that the three features in both the training and testing datasets have already been processed using one-hot encoding, shown in Figure4 and 5.

	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

Fig. 4. The describe of the train data

From the first feature, it can be observed that a higher number of cards were activated during the period from 2016 to 2018 compared to earlier years, shown in Figure6

The loyalty score distribution reveals that most data points are concentrated between -10 and 10, with a few scores falling below -30(about 1%), as shown in Figure7.

But they're not outliers because they have the same relationship with target compare to the data point between -10 and 10, shown in Figure8. Moving forward, I will refer to the data

	feature_1	feature_2	feature_3
count	123623.00000	123623.000000	123623.000000
mean	3.10926	1.741796	0.564377
std	1.18911	0.749195	0.495840
min	1.00000	1.000000	0.000000
25%	2.00000	1.000000	0.000000
50%	3.00000	2.000000	1.000000
75%	4.00000	2.000000	1.000000
max	5.00000	3.000000	1.000000

Fig. 5. The describe of the test data

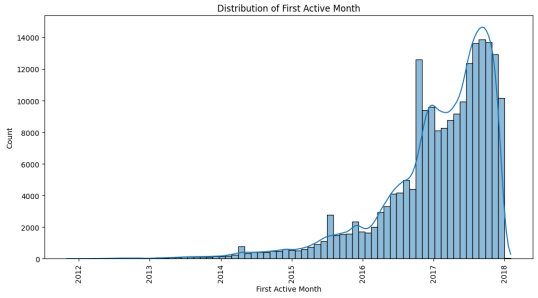

Fig. 6. Distribution of First Active Month

with scores below -30, which are similar to outliers, as the "special target." During the modeling phase, I will evaluate model performance by including and excluding the "special target" data to compare results.

B. Historical transaction data analysis

The historical transaction data file is the largest and most information-rich file among all the datasets. Due to its size, it cannot be directly loaded using pd.read_csv. To handle this, I process the data in chunks and specify the data type (dtype) for each column. This approach helps prevent program crashes caused by the inability to load the file into memory all at once. The historical transaction data has 29,112,361 records and 14 features. The following pie chart 9 illustrates that some features have imbalanced data. For instance, the majority of records are labeled as "approved." This suggests that it would be ideal to select models that are more robust to imbalanced data. Similarly, as shown in the bar plot10, certain features contain outliers, such as installments, which includes values like -1 and -25. During subsequent preprocessing, these outliers can be either removed or flagged for further handling.

C. New transaction data analysis

The new transaction data has 1,963,031 records and 14 features. The pie chart below11 shows that, similar to the historical transaction data, the features in the new transaction data also exhibit imbalance. Additionally, all transactions are

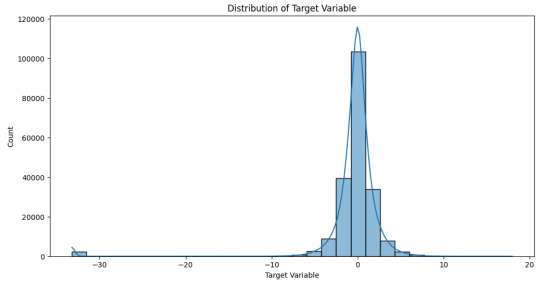

Fig. 7. Target Variable

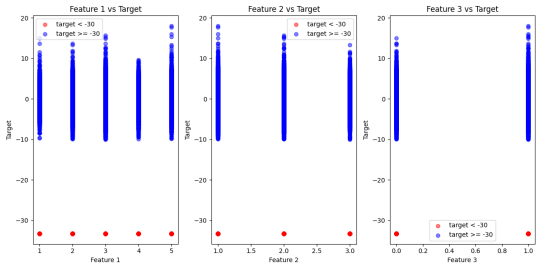

Fig. 8. The relationship between features and target

marked as "approved," making this column redundant and suitable for removal. The bar plot12 reveals that the installments feature contains outliers, such as -1 and 999. These outliers can be removed or flagged during subsequent preprocessing steps.

D. Merchant data analysis

The new transaction data has 334,696 records and 22 features.From the pie chart13 and bar plot14, it is evident that the merchant data, like the other datasets, also has a highly imbalanced distribution. It may be worth considering merging sparse categories and grouping them into an "Others" category.

IV. Data Preprocessing

Extensive preprocessing was necessary to ensure the quality and suitability of the data for analysis. Data preprocessing is a prerequisite for building a model. Raw data cannot be directly used in modeling as it may contain missing values, outliers, or excessively large data ranges that could introduce noise and negatively impact the model's performance. The preprocessing steps typically include data cleaning, data merging, feature engineering and data transformation.

A. Data Cleaning

- **Handling special data:** Processed some special data, such as Installments, which should be between 0–12 months. However, there are values like -1, -25, and 999. I removed the records with 999, but converted -1 and -25 into a new column called 'installments_flag', as these values might indicate special cases where no installment plan was selected or the data was not properly recorded during the transaction. This allows the model to retain information about these anomalies without directly influencing the Installments feature.

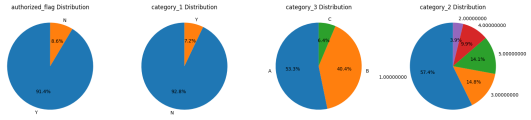

Fig. 9. The distribution of features in historical transaction data

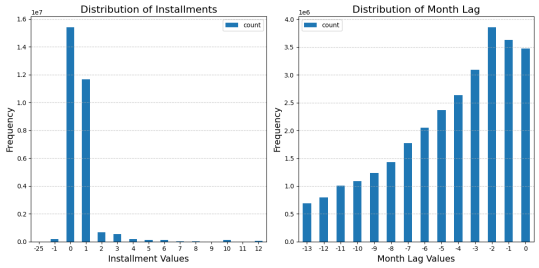

Fig. 10. The distribution of features in historical transaction data

- **Missing value:** I filled the missing values in the 'merchant_id' column with 'C_ID_272rop5ulu', which I created as a placeholder for missing entries. For the 'category_2' and 'category_3' columns, I used the values '6' and 'D', respectively, to represent new categories for these features. For the remaining numerical columns, such as 'avg_sales_lag3', I filled the missing values using the median of the respective columns. This approach ensures that the imputation strategy is appropriate for each type of data, minimizing bias while preserving data integrity.
- **Data Aggregation**The features in the training data alone are insufficient to accurately predict the labels. Since a single card can have multiple transactions, we can enhance the data by creating aggregated features from these transactions for each card. To achieve this, I enriched the transaction data by incorporating features from historical and new merchant transactions, consolidated using the 'card_id'. Additionally, I utilized merchant data, linked via 'merchant_id', to bridge the transaction and training datasets. Numerical columns were aggregated using functions such as 'min', 'max', 'mean', and 'std'(standard deviation), while categorical columns were aggregated by 'unique' counting unique values. [4]
- **Removing outlier:** To handle these outliers, I applied the Interquartile Range (IQR) method to identify and remove excessively large values. Specifically, I calculated the first quartile ($Q1$) and third quartile ($Q3$) for each feature and determined the interquartile range as

$$\text{IQR} = Q3 - Q1.$$

Outliers were defined as values lying outside the range

$$[Q1 - 1.5 \times \text{IQR}, Q3 + 1.5 \times \text{IQR}].$$

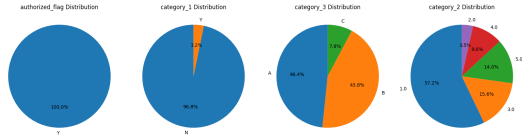

Fig. 11. The distribution of features in new transaction data

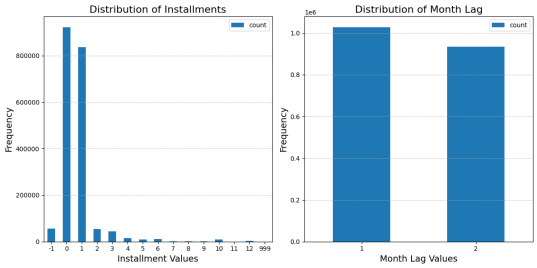

Fig. 12. The distribution of features in new transaction data

Fig. 13. The distribution of features in merchant data

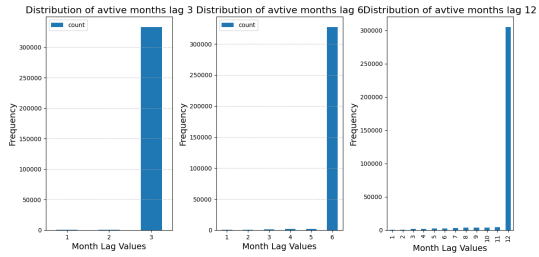

Fig. 14. The distribution of features in merchant data

Any data points exceeding these boundaries were removed to ensure that the dataset retained its consistency and robustness for analysis. To better understand the impact of outliers, I visualized the data distribution before and after removing outliers using boxplots, as shown in Figure 15.

B. Feature Engineering

New features were created from the 'data feature', such as 'first_active_month' in training and testing data, 'purchase_date' from historical and new transaction data. I extracted the year, month, and the interval in months between the last record date. To capture seasonality and behavioral trends, I extracted additional features such as the day of the week, whether the transaction occurred on a weekend, and the quarter of the year.

C. Data Transformation

- **Label Encoding:** Applied Label Encoding to transform categorical features into numerical, such as 'id' columns, making the data compatible with machine learning algorithms.
- **Standardization:** As we can see, the scale of different features are different. To ensure all numerical features operate on the same scale and to minimize biases caused by differences in feature magnitudes, I applied standardization on all the data.
- **Log transformation**I visualized the distributions of several key features using histograms with KDE (kernel density estimation). As shown in the distribution plots 16.It exhibited significant right-skewness. This indicates that the majority of the data points are concentrated near lower values, with a long tail of higher values. Severe skewness can adversely affect the performance of machine learning algorithms. To address this issue, I applied a logarithmic transformation to these features. After the log transformation, I re-visualized the distributions (Figure 17). The transformed features showed a more symmetric and balanced distribution, which improves the stability and performance of subsequent modeling.

After completing all the preprocessing steps, I created a new training dataset consisting of 201,917 samples, each with 180 features and 1 target variable. Similarly, the testing dataset comprises 123,623 samples, each with 180 features. To better understand the relationships among the 180 features in the dataset, I visualized their correlations using a heatmap (Figure 18). Each cell in the heatmap represents the correlation between a pair of features, with values ranging from 0 (no correlation) to 1 (strong positive correlation). The majority of the features exhibit low correlations with each other (values close to 0), indicating minimal redundancy. Similarly, the correlation between the target variable and most features is also low, which is favorable as it reduces the risk of redundant information negatively impacting model training.

V. Models

A. Evaluation Metrics

I use RMSE as an evaluation metric. RMSE represents the average error between the predicted loyalty score and the true loyalty score, where $\hat{y}_i$ is the predicted loyalty score for each `card_id`, and y_i is the actual loyalty score assigned to a `card_id`. [1]

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

B. Model Choose

- **Linear Regression:** Linear Regression is an efficient baseline model for regression, providing a fundamental performance benchmark for comparison with more complex models.
- **SVR(Support Vector Regression:)** SVR is well-suited for high-dimensional data as it leverages kernel functions to handle nonlinear relationships and uses regularization parameters to prevent overfitting. For data where features have low correlation with the target, SVR can effectively

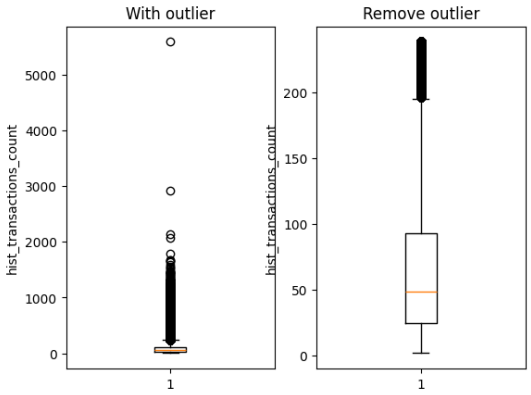

Fig. 15. Before and After removing outlier

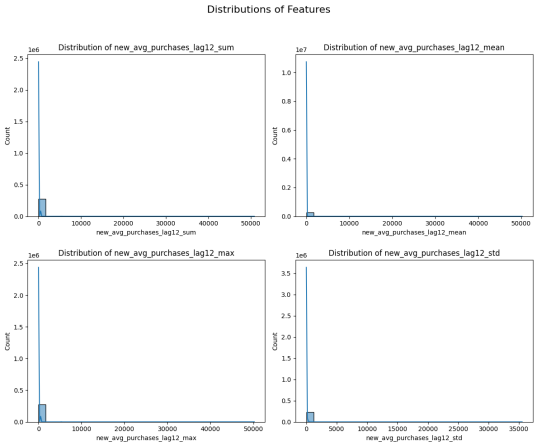

Fig. 16. Before log transformation

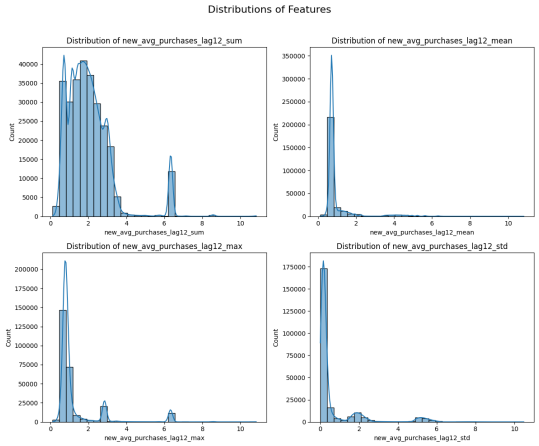

Fig. 17. After log transformation

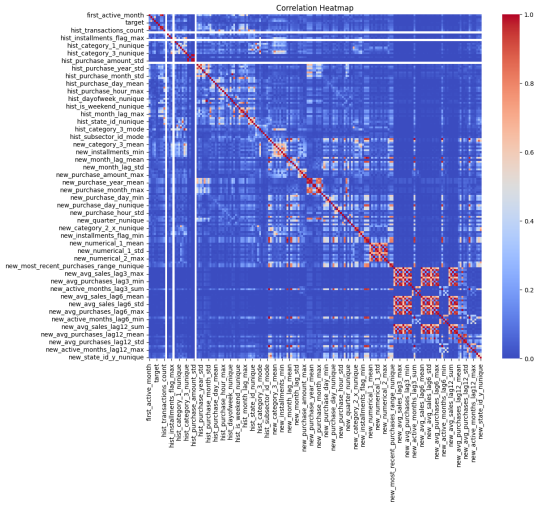

Fig. 18. Relationship between features and target

capture complex interaction patterns through its kernel approach, making it an ideal choice.

- **Random Forest:** Random Forest make predictions using multiple decision trees, effectively reducing overfitting in single models, especially in scenarios with imbalanced feature distributions, like my data. Additionally, tree models excel at capturing nonlinear relationships.Furthermore, it is simple to implement, requires minimal parameter tuning, and is insensitive to feature scaling.
- **XGBoost:** XGBoost is also a tree-based model, and it outperforms Random Forest in capturing more complex relationships. Leveraging gradient boosting, XGBoost incrementally optimizes prediction accuracy and incorporates L1/L2 regularization to prevent overfitting. It also automatically selects the most important features for the target variable, making it well-suited for high-dimensional data and complex nonlinear relationships.
- **ANN(Artificial neuron network)**Customer loyalty may be influenced by various interacting factors, some of which likely involve nonlinear relationships. ANN is well-suited for capturing this complexity through multiple hidden layers with nonlinear activations, allowing it to model intricate patterns in the data. Additionally, ANN can handle large datasets and high-dimensional feature spaces.

C. Model Compare

Due to the large dataset, using all of the data for the parameter tuning would result in higher computational costs. Therefore, I decided to randomly sample 10% of the data to perform the parameterization. To ensure the representativeness of the sampled data, I used simple random sampling and made sure that the distribution of the sampled data was similar to the original data(Figure 19). In this way, I can effectively reduce the amount of calculations.

- **Linear Regression**Linear Regression is an efficient baseline model for regression, providing a fundamental performance benchmark for comparison with more complex models. While it has fewer hyperparameters, I incorporated regularization techniques, such as Lasso, to reduce noise from less important features. To further improve the

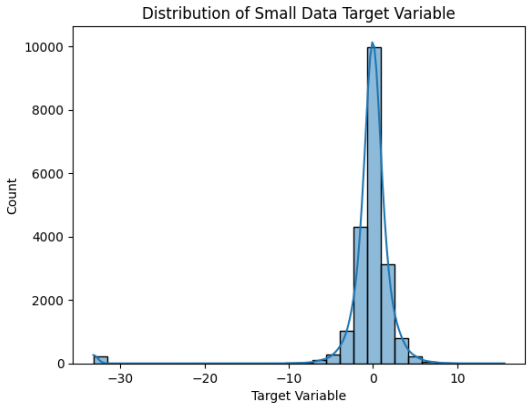

Fig. 19. Distribution of Small Data Target Variable

model, I explored dimensionality reduction using Principal Component Analysis (PCA)(Figure 20). Specifically, I applied PCA to reduce the feature set to 4 principal components, aiming to simplify the data structure and improve the model's performance. However, the results

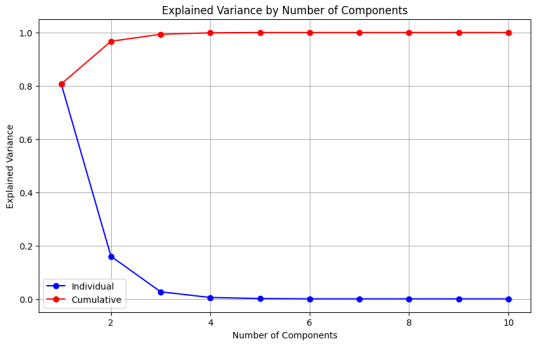

Fig. 20. PCA

showed that using PCA led to slightly worse performance compared to using the full feature set. Without PCA, the model achieved a Train RMSE of 3.7894 and a Test RMSE of 3.8214. In contrast, with PCA (n=4), the Train RMSE increased to 3.8440, and the Test RMSE increased to 3.8761. This indicates that PCA reduced the performance of the model. This is because the most of the features in the dataset are weakly correlated, applying PCA led to a reduction in the data's informational content.

- **SVR(Support Vector Regression)** I used the Support Vector Regression (SVR) model with the RBF kernel to make predictions on a random sample. The other parameters are C=100, which controls the model's complexity and tolerance for errors. epsilon=0.1,which defines the tolerance margin. The Train RMSE is 2.31 and the Test RMSE is 3.70. Compared to the benchmark model (linear regression), SVR achieved a improvement in prediction accuracy. However, since C=100 increased the model's complexity, the model becomes overfitting, as indicated by the large gap between the training and testing errors. Moreover, while the RBF kernel performs well for nonlinear problems, its high computational complexity resulted in a prolonged runtime (24 minutes for 10% of the data).Considering both prediction performance and computational efficiency, I decided not to select this model as the final solution.
- **ANN(Artificial neuron network)** The ANN model consists of 13 layers, including fully connected layers (Dense), batch normalization layers (Batch Normalization), and dropout layers (Dropout). Details of the architecture are shown in the figure below. The training process took 3 minutes, completing 10 epochs, with the Train RMSE of 3.84 and Test RMSE of 3.88. Compared to Random Forest and XGBoost, the training time for ANN is significantly higher. Additionally, its accuracy is not better than those two models. Due to the large number of parameters in ANN, the training process is more complex, and the cost of hyperparameter tuning is higher.Considering computational efficiency, prediction accuracy, and the cost of tuning, ANN is currently not suitable as the final solution.
- **Random Forest and XGBoost** For both Random Forest and XGBoost, I trained the models using default parameters. From the results, it is evident that both models exhibit varying degrees of overfitting. Specifically, Random Forest has a training RMSE of 1.492 and a test RMSE of 3.571, while XGBoost has a training RMSE of 1.591 and a test RMSE of 3.691. Although the test RMSE of both models is not significantly different from that of SVR, they show a clear advantage in efficiency: Random Forest took around 5 minutes to train, whereas XGBoost only took 3 seconds. Considering the performance, efficiency, and potential of both models, I will decide on the final model after completing a comprehensive hyperparameter optimization.

D. Hyperparameter Tuning

Parameter tuning is a complex process. Initially, I attempted to tune all the parameters simultaneously, which led to system resource overload and prevented the process from completing successfully. Therefore, I adopted a step-by-step tuning strategy to adjust the main parameters more efficiently. This sequential optimization approach significantly improved the tuning process, allowing the model to find optimal values for each parameter without being overwhelmed by the vast number of potential parameter combinations. In each stage of tuning, I utilized GridSearchCV combined with negative mean squared error as the evaluation metric to identify the best parameter combination. For the Random Forest model: Step 1: Tune n_estimators to determine an appropriate number of trees. Step 2: After deciding on the number of trees, adjust max_depth and min_samples_split to influence the tree's structure and overall complexity. Step 3: Tune min_samples_leaf to ensure that each leaf node has a sufficient number of samples, thereby preventing overfitting. Step 4: Finally, tune

max_features to find an optimal number of features, which is crucial for enhancing the diversity and robustness of the model. The final result of parameter tuning yielded: n_estimators = 200 max_depth = 7 min_samples_split = 2 min_samples_leaf = 3 max_features = 'sqrt' With these parameters, the model achieved a mean RMSE of 3.75, with a running time of 48 seconds. For the XGBoost model: Step 1: Start by tuning n_estimators to determine the appropriate number of boosting rounds. Step 2: Adjust max_depth and min_child_weight to control the complexity of each individual tree. max_depth limits the depth of the trees, helping to prevent overfitting, while min_child_weight ensures a minimum weight of instances in each leaf node. Step 3: Tune gamma, which controls the minimum loss reduction required to make a split. This helps in regularizing the model and reducing complexity. Step 4: Adjust subsample and colsample_bytree to add randomness and avoid overfitting. subsample represents the fraction of samples used for each tree, and colsample_bytree represents the fraction of features used in each split. Step 5: Finally, tune learning_rate (eta), which controls the contribution of each tree. A smaller learning rate generally leads to better results but requires more boosting rounds (n_estimators), which could increase training time. The final tuned parameters for XGBoost are as follows: n_estimators = 300 max_depth = 5 min_child_weight = 3 gamma = 0.2 subsample = 0.9 colsample_bytree = 0.5 learning_rate = 0.01 With these optimized parameters, the model achieved a mean RMSE of 3.73, with a running time of 32 seconds. In terms of tuning results and efficiency, the XGBoost model outperformed the Random Forest model, achieving a lower RMSE while also being more computationally efficient.

VI. Results and evaluation

The baseline model used was Linear Regression, which achieved a Train RMSE of 3.7894 and a Test RMSE of 3.8214. This benchmark provides a simple reference point to assess the improvement achieved by more complex models. The Support Vector Machine (SVM) model showed promising accuracy on the training set, with a Train RMSE of 2.31. However, the Test RMSE reached 3.70, indicating potential overfitting. Moreover, the training time for SVM was significantly longer, taking 24 minutes, making it a less favorable choice when computational efficiency is a key consideration. Artificial Neural Network (ANN) was then evaluated, yielding a Train RMSE of 3.84 and a Test RMSE of 3.88, with a training time of 3 minutes. The training and testing errors are quite close shows a good generalization capability. However, its overall performance slightly lags behind more advanced models, and its training time is relatively longer compared to simpler ensemble models. Random Forest (RF) produced a Train RMSE of 3.4794 and a Test RMSE of 3.4573, with a training time of just 48 seconds. The close match between the training and testing errors indicates good generalization, and the relatively short training time makes Random Forest an attractive option. Finally, XGBoost was evaluated, resulting in a Train RMSE of 3.3718 and a Test RMSE of 3.4522, with a training time of 32 seconds. XGBoost outperformed Random Forest slightly in both training and testing error, while also requiring less training time. This suggests that XGBoost effectively balances predictive performance and efficiency, making it the best performing model in this experiment. After

Model	Train RMSE	Test RMSE	Training Time
Linear Regression	3.7894	3.8214	-
ANN	3.84	3.88	3 minutes
SVM	2.31	3.70	24 minutes
Random Forest	3.4794	3.4573	48 seconds
XGBoost	3.3718	3.4522	32 seconds

Fig. 21. Summary Table of Model Performance

training with the full dataset, the XGBoost model performs as follows: the Train RMSE is 3.644 and the Test RMSE is 3.734. While the test error is still slightly higher than the training error, showing some degree of generalization error, this result is close to the RMSE of the top player on the Kaggle charts (as of January 27, 2019) of 3.637 [5]. The performance of the model after using all of the data indicates that the model fits with larger amounts of data and is able to more fully learn complex patterns in the data. Although the RMSE values are still a bit short of the top players, the results are quite competitive considering the simplified process and relatively limited tuning of this training. Here's a refined version of your statement for the report:

The feature importance was evaluated using the XGBoost model, as illustrated in Figure 22. The results indicate that most features contribute meaningfully to the model's performance, suggesting that the features are well-suited for capturing the underlying patterns in the data.

VII. Ethical Considerations

In the context of my project, defining what constitutes a "loyal" customer is essential but also challenging. Typically, loyalty models evaluate indicators such as purchase frequency and transaction amounts. However, these criteria may not apply uniformly to all customers. For instance, customers who live in remote areas or face financial constraints may not make frequent purchases, yet they could still exhibit strong loyalty to the brand. Over-relying on narrow, transactional metrics can therefore lead to misjudging loyalty and inadvertently excluding such customers from promotional efforts. To address this issue, broader criteria should be considered, including measures like customer engagement across platforms, feedback history, and brand advocacy behaviors. O'Neil, for example, illustrates a feedback loop in predictive policing, where areas labeled as "high-crime" receive intensified surveillance, generating more crime data and entrenching biases against certain neighborhoods. This cyclical reinforcement of bias highlights how relying on historical data can perpetuate existing inequalities. [6] A similar phenomenon can occur in loyalty prediction: if customers are labeled as "high loyalty" or "low loyalty," it directly impacts the company's promotional efforts towards these groups. High-loyalty customers may receive more discounts and offers, thereby increasing

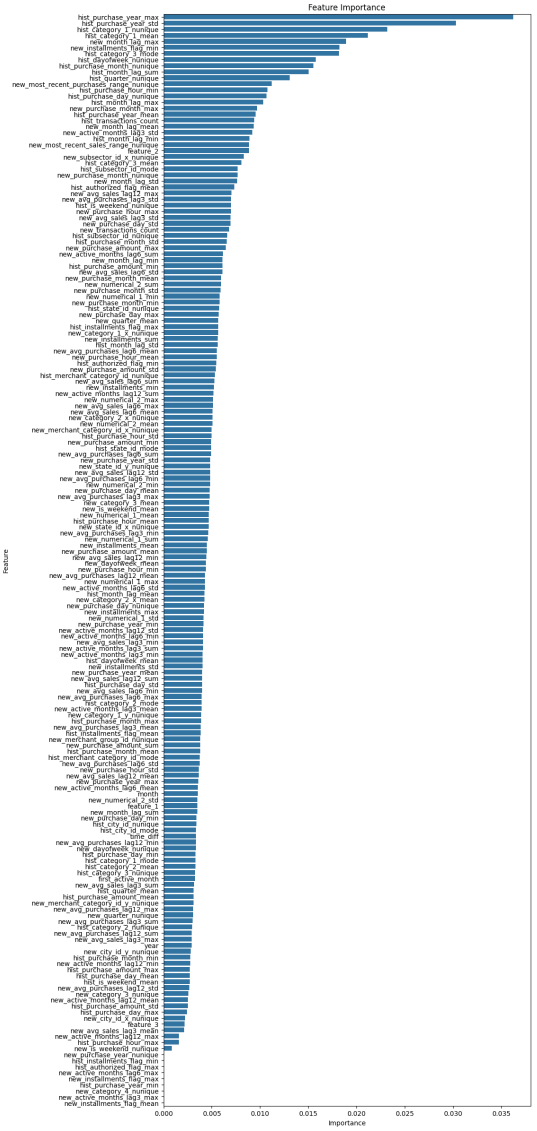

Fig. 22. Feature importance

their purchase frequency and reinforcing their loyalty status. On the other hand, customers labeled as low-loyalty may receive less attention and fewer incentives, reducing their engagement over time and solidifying their exclusion from the loyal customer segments. Such feedback loops raise ethical questions about fairness and equitable treatment in customer engagement strategies. One potential solution is to periodically reevaluate and recalibrate the loyalty model to prevent entrenched biases and make dynamic adjustments based on evolving customer behaviors. The loyalty prediction model offers significant advantages to companies, including reduced costs in acquiring new customers and the ability to anticipate and cater to customer needs through tailored, timely promotions. This approach not only helps identify loyal customers in advance but also enables cost-effective, personalized engagement. However, loyalty models also risk perpetuating privacy issues and unfair treatment of certain groups due to algorithmic biases. Addressing these concerns requires a multifaceted approach. Companies should inform users about the specific types of data being collected and the purpose of the algorithms. Additionally, allowing customers to opt out of data collection respects user autonomy and privacy. Finally, when defining loyalty, metrics beyond transaction volume, such as brand engagement, feedback contributions, and long-term loyalty trajectories, should be incorporated. Periodically communicating how loyalty scores are generated can enhance transparency, enabling users to understand and engage more meaningfully with the scoring process.

VIII. Conclusion

In conclusion, XGBoost demonstrated superior performance with a Test RMSE of 3.734 and a training time of 32 seconds. The results of this analysis suggest that XGBoost is well-suited for the problem at hand, providing an effective balance between accuracy and computational efficiency. This makes it a strong candidate for deployment in practical applications where predicting customer behavior or loyalty is essential. By improving customer targeting, this model has the potential to optimize marketing strategies and enhance customer retention rates. However, there are still some limitations. The RMSE, while competitive, remains higher than desired, and there is still a slight degree of overfitting. Addressing these issues, we could further explore different models and parameter settings.

A. Future Work

- **Regarding the ANN Model** Although the current performance of the ANN model is not as good as the tree-based models, its strong nonlinear modeling capability may enable it to perform better on more complex datasets in the future. From the RMSE curve 23, it is evident that the ANN has not fully converged after 10 epochs. This suggests that the model's performance could be further improved by increasing the number of training epochs or optimizing the learning rate. Future work could involve further exploration of the ANN model, such as adjusting its network architecture, regularization parameters, and optimization algorithms.

Fig. 23. Training and Validation RMSE in ANN

- **Regarding XGBoost Models Trained on Separate Data Segments** The data was divided into two parts: the special target segment and the non-special target segment, and

two separate XGBoost models were trained for these parts. The testing RMSE scores were non-special target segment of 1.59 and special target segment: Close to 0. These results indicate that training separate models for different data segments can significantly improve prediction performance. In future work, combining these two models to jointly predict the dataset could be explored to further enhance the overall predictive accuracy.

REFERENCES

[1] Kaggle, "Elo merchant category recommendation," https://www.kaggle.com/competitions/elo-merchant-category-recommendation, 2018, accessed: 2024-10-02.

[2] J. Latheef and S. Vineetha, "Exploring data visualization to analyze and predict customer loyalty in banking sector with ensemble learning," *International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE)*, vol. 4, no. 9, pp. 891–904, Mar. 2021. [Online]. Available: https://doi.org/10.29027/IJIRASE.v4.i7.2021.891-904

[3] C. P. Amajuoyi, L. K. Nwobodo, and A. E. Adegbola, "Utilizing predictive analytics to boost customer loyalty and drive business expansion," *GSC Advanced Research and Reviews*, vol. 19, no. 3, pp. 191–202, Jun. 2024.

[4] L. Grey, *Elo Merchant Category Recommendation*, https://www.kaggle.com/code/greylivingstone/elo-merchant-category-recommendation, 2022, accessed:2024-10-20.

[5] S. Aslam, G. van der Heijden, and H. Collins, "Predicting customer loyalty using various regression models," *Advanced Machine Learning - Group 21*, 2019.

[6] C. O'Neil, "The era of blind faith in big data must end," https://www.ted.com/talks/cathy_o_neil_the_era_of_blind_faith_in_big_data_must_end, 2017, accessed: 2024-11-09.