Tide - Tagging Transactions to receipts

**Objective:** The objective of the analysis is to build a machine learning model to identify the most relevant transaction pertaining to the receipt scanned on the app

**Context:** The receipt details are scanned and information is extracted, the extracted data may deviate from the actual data due to the limitations of the software. Extracted receipt details are matched with the transaction data available in the tide app based on various transaction properties such as date and time of the transaction, merchant name (short/long) and transaction amount.

**EDA Insights:**

1. There are about 1,155 receipts in the dataset provide, 857 receipts have a correct transaction mapped based on (Merchant/Feature transaction id)
2. **Univariate analysis:** There are no outliers in the data or missing values in the data
3. **Bivariate analysis:**
   1. DifferentPredictedDate and DateMappingMatch are perfectly correlated
   2. DifferentPredictedTime and TimeMappingMatch are perfectly correlated
   3. We will skip this data to remove redundant variable in the model
   4. **DateMappingMatch** has the highest correlation with the correct transaction tagging classification followed by **TimeMappingMatch**
   5. Name and descriptions mapping have shown a positive correlation with correct classification
4. Model importance:
   1. Date and Description matching seems to be more important matching properties in order to identify the right transaction given a set of possible transactions

**Model training and analysis:**

1. LightGBM model is used as it is a powerful model and lightweight which can be used in production for very fast inference compared to other ensemble tree based models which provides similar level of accuracy such as XGBoost, Catboost
2. It is an imbalanced dataset with about 7.2% examples having positive class and appropriate class weights are chosen to provide highly accurate models so that model focus on positive examples classification correctly, before selecting the final weights, we have used the default option of ‘balanced’ option which lead to suboptimal performance, class weights are tuned to provide more robust model over the default option
3. Given the time constraints, more important hyperparameters are prioritized to be tuned

such as number of trees, depth of trees, class weights, learning rate,

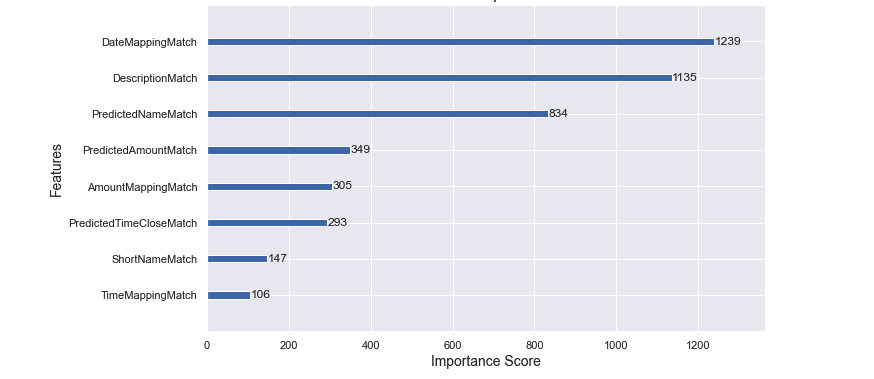
1. We have used precision recall graph to identify the trade off points to have a fairly robust model to select the right cut off threshold [0.76] to classify correct tagging, we have prioritized focusing on providing high precision model
2. We were able to build a fairly accurate model with ~90% precision, 70% F1-Score and about 56-57% Recall. AUC of the model is more than 75% and overall model accuracy is about 96%.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Train/Test Split - 70:30** | |  |  |  |  |
|  | **Precision** | **Recall** | **F1- Score** | **Accuracy** | **AUC** |
| Train | 89% | 57% | 70% | 96% | 79% |
| Test | 88% | 56% | 69% | 97% | 75% |
| \*Precision/Recall/F1-Score are reported w.r.t positive class as reference, 0.76 cutoff threshold is used to classify positive vs negative class | | | | | |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Model Insights**

1. Date and Description matching seems to be more important matching properties in order to identify the right transaction given a set of possible transactions
2. PredictedName match is also an important variable based on the trained model

**Feature importance plot based on frequency**

****

**Trained Model saving**

1. Model is saved using ‘**joblib’** module in python and shared as *Tide\_receipt\_transaction\_mapping\_model.sav* file
2. It can be loaded and be used for inference

**Recommendations to improve model performance:**

1. **Collect more training data :** Have more training data will provide model to learn more patterns and helps improve generalization over a wide range of examples
2. **Explore other imbalance methods:**There are other data balancing methods such as over sampling, undersampling, SMOTE based methods which can lead to performance gains
3. **Experiment with other ML models :** We should experiment with other model such as XGBoost, Catboost, Linear models to compare the model overall performance [accuracy/speed] to optimize, also experiment with more sophisticated deep learning models to improve the model accuracy
4. **Get better matching input** :Current model matching variables will improve with better quality of input such as better receipt keyword extraction in terms of numbers, date, name and description match
5. **Hyperparameter tuning:** Have done some experiments with hyperparameter tuning using **optuna** module based optimization methods, didn’t contribute a lot to getting good model, we should explore other hyperparameter tuning to provide a more efficient model over more parameters to see if it helps model evaluation metrics
6. **Feature engineering :** Explore interaction variable and variable transformation, improve the matching logic criteria which is used as model input

**Next steps:**

1. **Deploy the model :** Use the saved model and integrate with the API to allow for inference on the app in real time. Select the best transaction based on the highest probability score based on the matching properties
2. **Track the model performance :** Monitor the model accuracy metrics, classification rate to observe the performance and check for data/model/concept drifts
3. **Retrain** the data with more samples as we have more samples collected
4. Use more sophisticated hyperparameter optimization,experiment with new models etc
5. Explore options provided in the recommendations to make the model more robust across all evaluation metrics

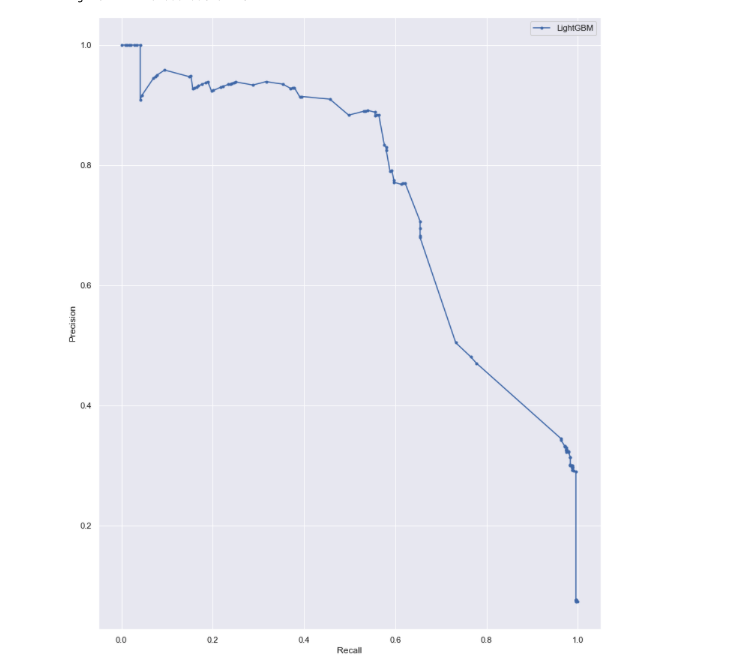
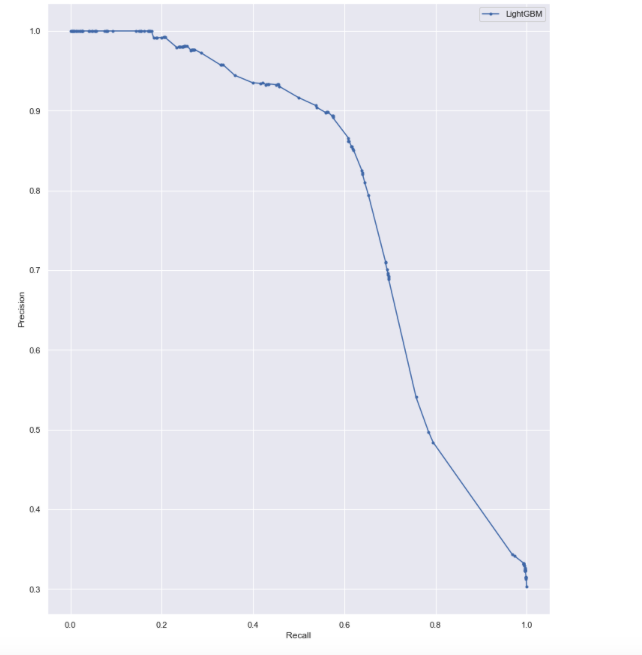
**Appendix**

1. Standard cut off basis Model evaluation [0.5]

|  | Precision | Recall | F1- Score |
| --- | --- | --- | --- |
| Train | 81% | 64% | 72% |
| Test | 77% | 60% | 67% |

Model improve with selecting the right threshold

1. Precision Recall Curves [Training and Testing data]

****