MS4610 Project Report

PayBuddy Case Study

Group 27

20th December, 2021

Contents

Introduction

An Online Payments Platform that was split from PayBay in 2015 and became an independent company. The company hosts a diverse portfolio of credit products like credit card, pay later etc. and facilitates money transfer service for free of cost between any two PayBuddy users. To ensure competitive advantage over competing brands, the company needs to build a propensity model to predict the customer propensity to apply for the new line of credit card called Evolution Card. The goal is to train a model that is able to give a benchmarking performance.

Requirement: To train a model that is able to calculate propensity score for a given customer.

The Libraries

MATPLOTLIB: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It allows us visual access to huge amounts of data in easily understandable forms with its many plotting features such as scatter plot, bar, stem, box-plot etc.

NUMPY: NumPy is a Python library used for working with arrays. Arrays are very frequently used in data science, where speed and resources are very important. It aims to provide an array object that is up to 50x faster than traditional Python lists. This allowed us to be able to manipulate and access data efficiently.

SEABORN: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It gives a more abstracted interface compared to matplotlib. Its plotting functions operating on dataframes and arrays containing whole datasets generated informative plots which helped us understand the data better.

PANDAS: Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool. We used this to load, store, manipulate, analyze and merge csv files in dataframes. It is much more optimized and varied in its uses in comparison to lists in python.

SKLEARN: It provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method. This made our analysis easier by providing various built-in models.

Preliminary Analysis and Operations on Data

Pre-processing

- **Correlation heat maps**

  A correlation heatmap is a heat map that shows a 2D correlation matrix between two discrete dimensions, using colored cells with the colors proportional to the correlation. Patterns and trends in the same dataset can be easily read and analyzed. We've plotted seaborn correlation heatmaps on all the datasets to find such patterns and made note of the highly correlated features. This allowed us to reduce the dimensionality wherever possible and run models on these reduced datasets.

- **Standard scaling**

  Standard scaling is a technique where the values are made to center around the mean with a unit standard deviation. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values. To avoid this, we used standard scaling on the features before running the models on them.

- **Principal Component Analysis**

  Principal Component Analysis (PCA) is a statistical procedure to represent a multivariate data table as a smaller set of variables (summary indices) called principal components. As PCA calculates a new projection on a data set, standard scaling is necessary to ensure that all the variables have the same weight. We used PCA on the merged dataset (containing all the different given datasets) to halve the total number of dimensions without significant loss in information. We also used PCA on the different data sets themselves and brought down the dimensions to form a different merged dataset with a lower number of features. It reduced the computational time in the model fitting.

- ## Imputation techniques

Imputation techniques are used for replacing the missing data with some substitute value to retain most of the data/information of the dataset. We have tried replacing the missing values in the dataset by calculating a statistical value for each column and replacing all missing values for that column with that statistic. The statistical values tried out were the mean, median, minimum, maximum and zero.

Based on the performance of models on each of these, median was found to be the best performing statistic followed by minimum. This outcome might be explained by the fact that outliers do not significantly affect the median of the data thus making it a reasonable statistic to use for replacement.

Algorithms tried

- ### Random Forests

  Random forests is a supervised machine learning algorithm that is constructed from decision tree algorithms. It uses bagging, i.e., it creates multiple decision trees and then selects the best performing ones out of them by voting. Two types of randomness are built into the algorithm. First, each tree is built on a random sample from the original data. Second, at each tree node, a subset of features are randomly selected to generate the best split.

  While the results obtained were better than those from simple logistic regression, we did not use these results in the final voting classifier owing to better results from the gradient boosting models.

- ### Linear Discriminant Analysis (LDA)

  Linear Discriminant Analysis (LDA) is a classifier used to find a linear combination of features, which separates two or more classes of data. The succeeding combination can be used as a linear classifier. While it can also be used in data preprocessing to reduce the number of features, we used it as a classifier on our data.

  Although running LDA on a reduced dataset seemed to be promising, predicting a much larger proportion of 1's (very close to the proportion in the training data) than any other model, it gave an ROC AUC score of 0.63298, which was quite lesser

than the scores of the gradient boosting models. Modelling it on the entire data gave an ROC AUC score of 0.71395.

- XGBoost

  XGBoost or extreme gradient boosting is a decision-tree-based ensemble Machine Learning algorithm. It is an advanced implementation of gradient boosting along with some regularization factors and is used as an alternative to bagging. When used with GPU, due to parallelization (leveraging hardware as an alternative way to speed up the algorithm), the model ran comparatively much faster.

  We used Randomized Search CV for the hyperparameter tuning on the validation dataset to get the best ROC AUC score. This XGBoost modelled on the entire data set gave an ROC AUC score of 0.75532 on the 70% of test data, one of the best performances we got among all the models. We also ran it on dimensionally reduced datasets and obtained similar but lower scores due to the loss in information.

- CatBoost

  CatBoost is a recently open-sourced machine learning algorithm from Yandex. It is known for providing powerful out-of-the-box support for the more descriptive data formats that accompany many business problems and yields state-of-the-art results without extensive data training typically required by other machine learning methods. It is based on gradient boosting. It also employs ordered boosting in order to eliminate overfitting.

  After tuning the hyperparameters for the model with RandomizedSearchCV, the model gave an ROC AUC score of

0.73744. Combined with other models using a voting classifier, it contributed significantly to the final predictions.

- LightGBM

  LightGBM is a fast, distributed, high performance gradient boosting framework based on decision tree algorithms, used for ranking, classification and other machine learning techniques. LightGBM splits the tree leaf-wise as opposed to other boosting algorithms that grow tree level-wise. It takes less memory to run, is able to deal with large amounts of data and achieve good accuracy.

  We used Randomized Search CV for the hyperparameter tuning of the model on the validation dataset for the best ROC AUC score. This LGBM modelled on the entire data set gave an ROC AUC score of 0.75525 on the available test data, performing well while being computationally much faster compared to models like XGBoost.

- RandomizedSearchCV

  RandomizedSearchCV was used for hyperparameter tuning. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.

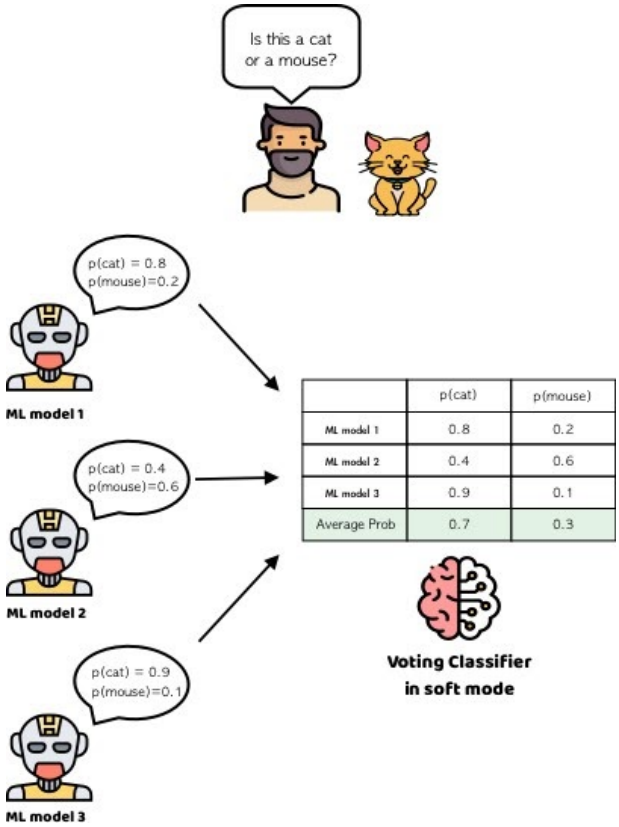
  The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings. We used an 80-20 split on the training data to obtain the validation set, which was maintained the same across all models using the same random_state for consistency in cross validation.

Conclusion

The best performance was obtained by means of a **voting classifier** in the **soft mode**.

Voting classifiers proved useful by eliminating the bias due to peculiarities pertaining to specific models. Their working is akin to that of an electoral system, where the probabilities from various models are weighted and then averaged. Soft voting is often **recommended in case of an ensemble of well-calibrated classifiers**, which is true for the models used for this project, i.e., gradient boosting models fitted after cross-validation using RandomizedSearchCV.

For this purpose, the models used were those trained on the entire data as well as those trained on the data after dimensionality reduction. Various combinations of the models were tried out, combining models on data both before and after PCA. The weights to be assigned to the models were determined by checking the ROC AUC scores on the validation set (after an 80-20 split of the provided training data).



- Accuracy metric

The assignment of weights was dependent on the **ROC AUC (Area Under Receiver Operating Characteristic Curve) score** of the models.

It is a **measure of the separability/intermingling of the predictions** of the two classes, 0 and 1 (here, 'will buy' and 'will not buy'), as opposed to the measure of the distance of predictions from their respective classes, which is commonly manifested in other accuracy metrics like the log-loss.

- Performance Achieved

The final model, chosen based on the best assignment of weights, achieved an **ROC AUC score of 0.75725** on the public leaderboard, which was judged on the basis of approximately 70% of the test data, i.e., 126,000 people.

The ROC AUC scores achieved on the validation set by various models were slightly lower, on average by 0.015, most likely owing to the smaller training set available for training the models to be tested on the validation set.

References

1. **Introduction to Principal Component Analysis:** https://www.kaggle.com/ryanholbrook/principal-component-analysis#Introduction
2. **Heatmap Basics with Seaborn:** https://towardsdatascience.com/heatmap-basics-with-pythons-seaborn-fb92ea280a6c
3. **Gradient Boosting:** https://bradleyboehmke.github.io/HOML/gbm.html
4. **Extreme Gradient Boosting Ensemble in Python:** https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/
5. **LightGBM Classifier in Python:** https://www.kaggle.com/prashant111/lightgbm-classifier-in-python -
6. **Ensemble Learning: Stacking, Blending & Voting:** https://towardsdatascience.com/ensemble-learning-stacking-blending-voting-b37737c4f483
7. **Intuition behind ROC-AUC score:** https://towardsdatascience.com/intuition-behind-roc-auc-score-1456439d1f30
8. **KS Score with Python:** https://www.listendata.com/2019/07/KS-Statistics-Python.html

Project Members

- Abdullah Mohammed - CS20B001
- Balakrishnan Aravindakshan - CS20B012
- Nalli Sai Soumya - CS20B053
- Trambadia Mantra Niteshbhai - CS20B083
- Bharat Meena - MM18B103
- Muzammil Aizaz - MM18B106