Homework 2 Report: Predicting Term Deposit Subscription - Decision Tree and Ensembles

Honglin (Leo) Wang
UNI: hw3124
The Fu Foundation SEAS, Columbia University, New York, NY 10034 USA
hw3124@columbia.edu

Abstract

In modern banking, effective marketing strategies are essential for customer acquisition and retention. A key objective is to predict which customers are most likely to subscribe to term deposit products, enabling more targeted and cost-efficient campaigns. This study aims to make better prediction in such scenarios using the Bank Marketing dataset from the UCI Machine Learning Repository. The task is to predict whether a customer will subscribe to a fixed-term deposit. Several models were implemented, including a Decision Tree, Random Forest, Gradient Boosting, XG Boost, and Light GBM. Performance was evaluated on this imbalanced dataset using weighted metrics including Accuracy, Precision, Recall, F1-score, AUROC, and AUC-PR. Results show that ensemble methods outperform the baseline Decision Tree. Confusion matrices, PR curves, feature importance barplot and learning curve (train & val loss) are illustrated, with bias-variance trade-off of boosting models discussed.

1 Introduction

In modern banking, effective marketing strategies are crucial for customer acquisition and retention. A common objective is predicting which customer segments are most likely to subscribe to time deposit products, thereby helping banks allocate resources efficiently and design more targeted marketing campaigns. This report addresses this challenge using a banking marketing dataset provided by the UCI Machine Learning Repository. The dataset includes customer demographics, socioeconomic attributes, and detailed information on past marketing campaigns conducted by a Portuguese banking institution. The target variable is simply to predict whether a customer subscribes to a fixed-term deposit: yes or no.

The core objective of this report is to build and evaluate predictive models for precise classification based on customer subscription propensity. We employed multiple supervised learning methods, including decision trees, ensemble algorithms such as Random Forest and boosting. By comparing these approaches, we aim to identify the model most effective at enhancing marketing campaign outcomes while analyzing the key feature contributions influencing customer decisions.

This report aims to demonstrate the application of machine learning techniques in real-world marketing scenarios, highlighting the trade-offs between model interpretability, predictive capability, and computational efficiency. The findings provide actionable insights for optimizing banking marketing strategies and enhancing customer engagement.

2 Data Preparation and Metrics

Metrics used to evaluate model performance are Accurary, Precision, Recall, F1 score, AUC (area under curve) for ROC and PR curve. Our evaluation focuses on AUC PR.

2.1 Dataset Information

UCI Bank Marketing Dataset is related with direct marketing campaigns of a Portuguese banking institution and was introduced in a paper by Moro. S. *et al*. It contains 2 main sub-datasets, in bank-full.csv (original) there are 45,211 cases and 17 attributes and in bank-additional-full.csv there are 41,188 cases and 21 attributes. The original one contains details collected from the clients and campaign such as the exact day last contact was made. The additional one inferred weekdays and assigned domestic trend indicators, such as *emp.var.rate* is the quarterly employment variation rate of Portugal, to the recorded days which may contribute as useful information in the prediction task. In **Table 1** we listed the 21 attributes used to predict bank term deposit subscription.

From *age* to *loan*, these first 7 input variables captures the bank client's features. The next 4 variables, *contact* to *duration*, describes the last contact of the current campaign. The 4 following variables, *campaign* through *poutcome*, describes other features of the campaign. Then the 5 following variables are social and economic context attributes. Lastly the *y* is our desired target. The underline indicatesthere are values set to "unknown" in the corresponding column.

Table 1 Attributes used to Predict Term Deposit Subscription

Column	Type	Description
age	numeric	Age in years (rounded).
job	category	Type of job (administrator, unemployed, management, student, etc.)
marital	category	Marital status: married, divorced (or widowed) or single.
education	category	Educational status: illiterate, basic 4/6/9y, high school, professional course, university degree.
default	binary	Has credit in default: yes/no.
housing	binary	Has housing loan: yes/no.
loan	binary	Has personal loan: yes/no.
contact	category	Contact communication type: telephone or cellular.
month	category	Last contact month of year: Mar - Dec (no inspection of Jan nor Feb).
day_of_week	category	Last contact day of the week: Mon - Fri (no record of Sat nor Sun).
duration	numeric	Last contact duration in seconds.
campaign	numeric	Number of contacts performed during this campaign and for this client (including last contact).
pdays	numeric	Number of days that passed by after the client was last contacted from a previous campaign, 999 means client was not previously contacted.
previous	numeric	Number of contacts performed before this campaign and for this client.
poutcome	binary	Outcome of the previous marketing campaign: success, failure or nonexistant.
emp.var.rate	numeric	Employment variation rate - quarterly indicator.
cons.price.idx	numeric	Consumer price index - monthly indicator.
cons.conf.idx	numeric	Consumer confidence index - monthly indicator.
euribor3m	numeric	Euribor 3 month rate - daily indicator.
nr.employed	numeric	Number of employees - quarterly indicator.
y	binary	Has the client subscribed a term deposit: yes/no.

2.2 Data Preprocessing

The dataset is well-formed. No noisy (inconsistent) value and missing value in the form of NA was found. Albeit, there are `unknown` values in the binary attributes, which will be treated as a separate category.

For *campaign*, *previous*, *pdays*, there are limited but many unique values, and we see from exploratory data analysis that the proportion of *yes* in *y* has some correlation with these features, though the linearity could be piecewise. Thus, we decide to divide them into several bins or apply top-coding based on this knowledge, and treat them as categorical features to reduce the possible branching in decision trees.

The *DecisionTreeClassifier* from *scikit-learn* package will treat categorical features as numeric. The divided bins described above is clearly not purely ordinal, and there is exceptional value 999 in the *pdays*. Thus, we are using one-hot encoding for categorical attributes.

Flagging may not be useful to treat binary features, since we still need to impute missing values, which may cause data leakage, unwanted manipulation of impurity when splitting, aggravating imbalance in the dataset (e.g. mode imputation for *default* or *loan*). Thus we decided to use one-hot encoding for binary features as well.

For the numeric features, we decided to keep its original form without any standardization, because decision tree type models generally doesn't take the mean or standard deviation of them into consideration of prediction, so standardization is unnecessary in making split in such features.

We can see the column 'duration' has a leakage on the output target. This feature is only known after call completion, but y is then (almost) simultaneously obtained. Also, we see that this leakage may strongly affect y: for instance, amongst cases with $duration \leqslant 36$, there is no *yes* in the target variable. Therefore, we discarded or transform this feature for realistic predictive modeling.

After one-hot encoding, we have obtained duplicative features. The *previous*==0 implies *pdays*==999 and is the same as *poutcome*==*nonexistent* logically and statistically. Thus we removed the redundant column *poutcome*==*nonexistent*.

The preprocessed dataset is then split into 3 parts by proportion of 70%, 15%, 15% as train, val (validation) and test respectively. To treat the imbalance in data correctly, we used the *sample_weight* computed on these subsets separately for training and evaluation.

3 Model Training

All codes and experiments are implemented under the environments as follows: Python 3.12.7, Numpy 1.26.4, Pandas 2.2.2, Sci-kit Learn 1.5.1, Seaborn 0.13.2, Matplotlib 3.9.2, XGBoost 3.0.1 and LightGBM 4.6.0.

3.1 Baseline Model: Decision Tree

We used the decision tree as our baseline model. Three hyperparameters were chosen for tuning: *ccp_alpha*, *max_depth* and *min_samples_split* by grid search with 5-fold cross validation. Due to the imbalance (there are only 4640 subscription compared to the total 41,188 cases) in our dataset, a naïve guesser which always predicts *no* would obtained an accuracy of 88.7%. Thus, we chose to use AUC PR, which focus on the model performance on the positive samples in imbalanced dataset, to monitor and rank the hyperparameters instead of accuracy.

GridSearchCV returns the best hyperparameters as *ccp_alpha*==10^{-6}, *min_sample_split*==200 and *max_depth*==5. At this point, the decision tree reached about 85% (77.4% using balanced sample weights) accuracy for both train and val. We further discuss how *max_depth* affects the model performance on train and validation, as shown in **Fig. 1**.

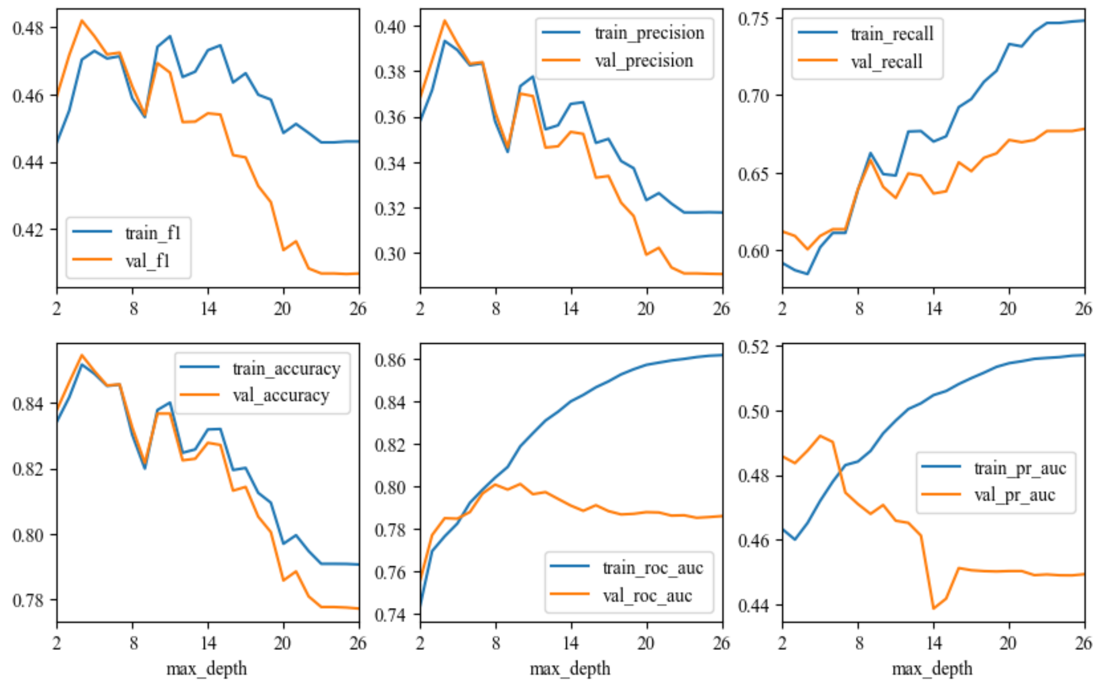

Fig. 1 Decision Tree Performance on Train and Val as Functions of Depth

From **Fig. 1** we can see that as the tree grows deeper, the AUC PR first has a slight increment, but soon dropped and leveled in deeper trees for validation, while AUC PR on the train set still increases, indicating overfitting. As trees grow, the precision, F1 score and accuracy drops but the recall improves, probably due to the model tries to predict positive samples more confidently.

3.2 Bagging: Random Forest

We attempted to use random forest as our baseline model and tuned its hyperparameters including *n_estimators*, *max_depth*, *min_samples_split*, *ccp_alpha* through GridSearchCV. The random forest classifier obtained an accuracy about 85% for both train and val, but a higher AUC PR ~ 0.466, larger than that 0.432 in decision tree. This shows the bagging method could reduce the variance.

3.3 Boosting: Gradient Boost, XG Boost and Light GBM

The Light GBM supports categorical features and would handle the best split on such attributes based on histograms. XGBoost supports these features in newer or experimental versions. Hence the Light GBM could produce more children nodes than 2, which adds another regularization hyperparameter (*num_leaves*) to it. For other regularization parameters such as *reg_alpha*, *reg_lambda* for regularizing splitting weights or *learning_rate*, both boosting models have similar implementation. To compare with other models, we train and evaluate all the models on the same dataset.

For boosting algorithms, we chose Gradient Boosting, XG Boost and Light GBM as our estimators, tuned their hyperparameters including *learning_rate*, *n_estimators*, *max_depth* and *subsample* through GridSearchCV and trained them based on the binary log loss. Best hyperparameters were chosen for later evaluation and best estimators are compared. Early stopping are used in all 3 of these models. We chose 3 values: 0.01, 0.1 and 0.3 for *learning_rate* in order to make comparison.

4 Model Evaluation

All of the 5 best estimators (1 for each type of models) are evaluated on the test set with 5-fold cross validation. The performance score is then taken the average of cross validation, with sample weights calculated on test set. Their performance in units of % over the 6 metrics are shown in **Table 2** with maximum's text bolded for each metric. To be noticed, the baseline for AUC PR is 11.27%, calculated as the proportion of positive samples in all cases.

Table 2 Models' Performance

Model	Accuracy	Precision	Recall	F1 score	AUROC	AUC PR
Decision Tree	77.39	29.03	**68.54**	40.70	77.45	43.21
Random Forest	**85.19**	**40.26**	64.23	**49.45**	**80.45**	46.60
Gradient Boost	84.50	38.59	62.79	47.74	79.35	46.71
XG Boost	84.43	38.66	64.08	48.17	79.73	**47.77**
Light GBM	84.08	37.62	62.22	46.83	79.56	47.29

First of all, all ensemble methods show a significant gain in accuracy and F1 score. The recall score drops because of that the proportion of recalled negative samples is decreasing.

We can see that the random forest (baseline model) unexpectedly behaves better than others in most metrics, and it has the highest accuracy. This shows the generalizability of random forest and its reductive effect of variance. However, it has second least AUC PR, which depicts how models perform under the imbalanced rare positive samples and the cost of false positives (the bank wastes resources contacting or targeting totally uninterested clients) is high. This shows the random forest is more conservative. For AUC PR, we see that all boosting models have higher scores, showing that they provide a better balance between precision and recall amongst positive samples.

Confusion Matrix The confusion matrices of all 5 models are shown in **Fig. 2**. It is noticeable that all boosting models predict a bit more false positives while having higher AUC PR.

PR Curve The PR curves of all 5 models are shown in **Fig. 3**. The horizontal dashed line shows the worst PR curve that describes an estimator always predicting positive.

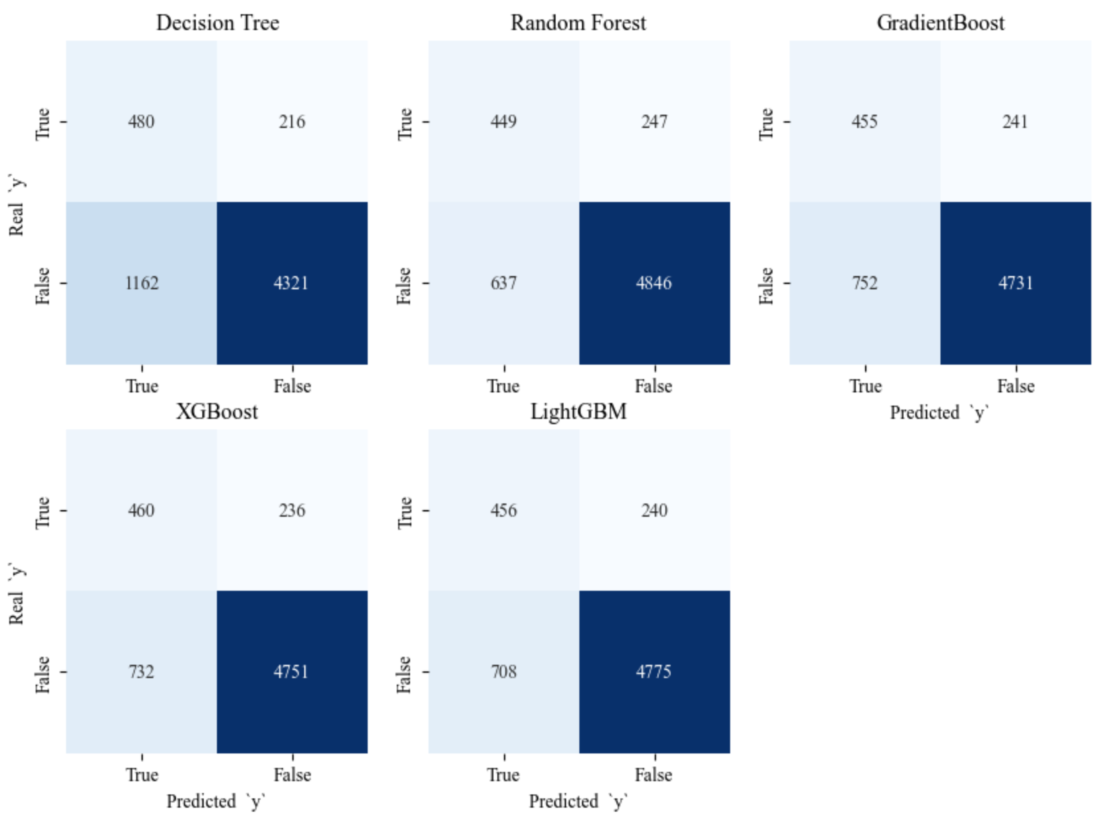

Fig. 2 Confusion Matrices of 6 Models

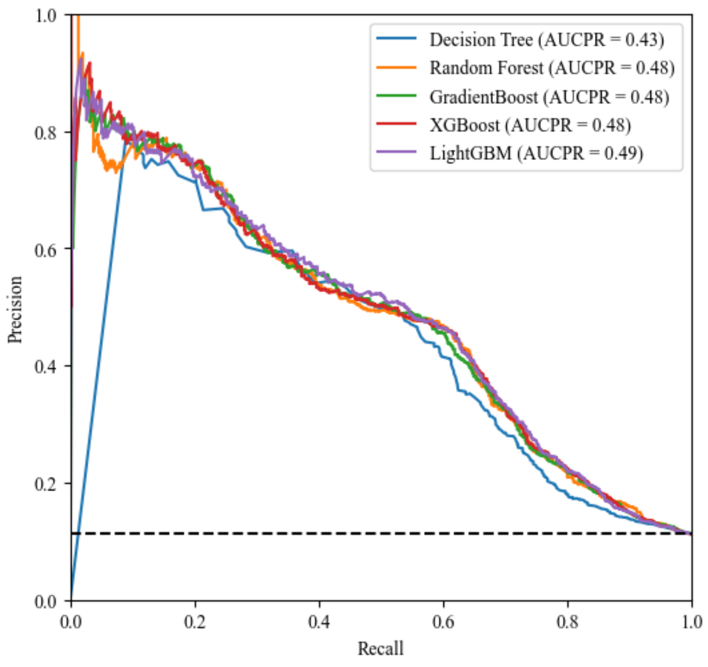

Fig. 3 ROC Curves of 6 Models

Feature Importance Barplot Feature importance barplots learnt and generated by 3 boosting models are shown in **Fig. 4** and **5**. The features are sorted by their importance in the models and only the top 30 are included. The numerical scales of importance are different from gradient boosting to the other 2 boosting models. Nevertheless, the important features coincides amongst these models. For instance, *nr.employed*, *euribor.3m*, *cons.conf.idx*, *age* are all in the top 8 features with prevailing importance values. These results aligns well with our experience and intuition: different *age* group acts differently, the deposit will is affected by the domestic employment and consumer confidence.

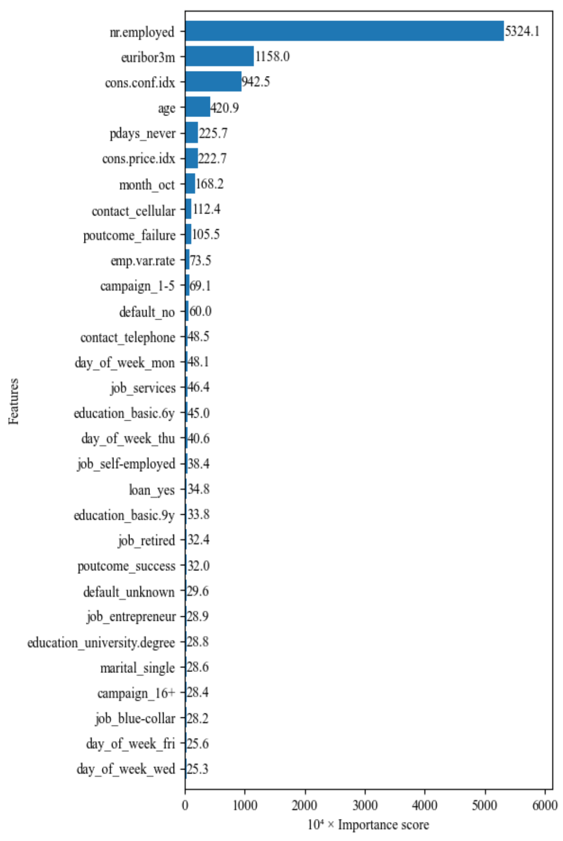

Fig. 4　Feature Importance of Gradient Boosting Model

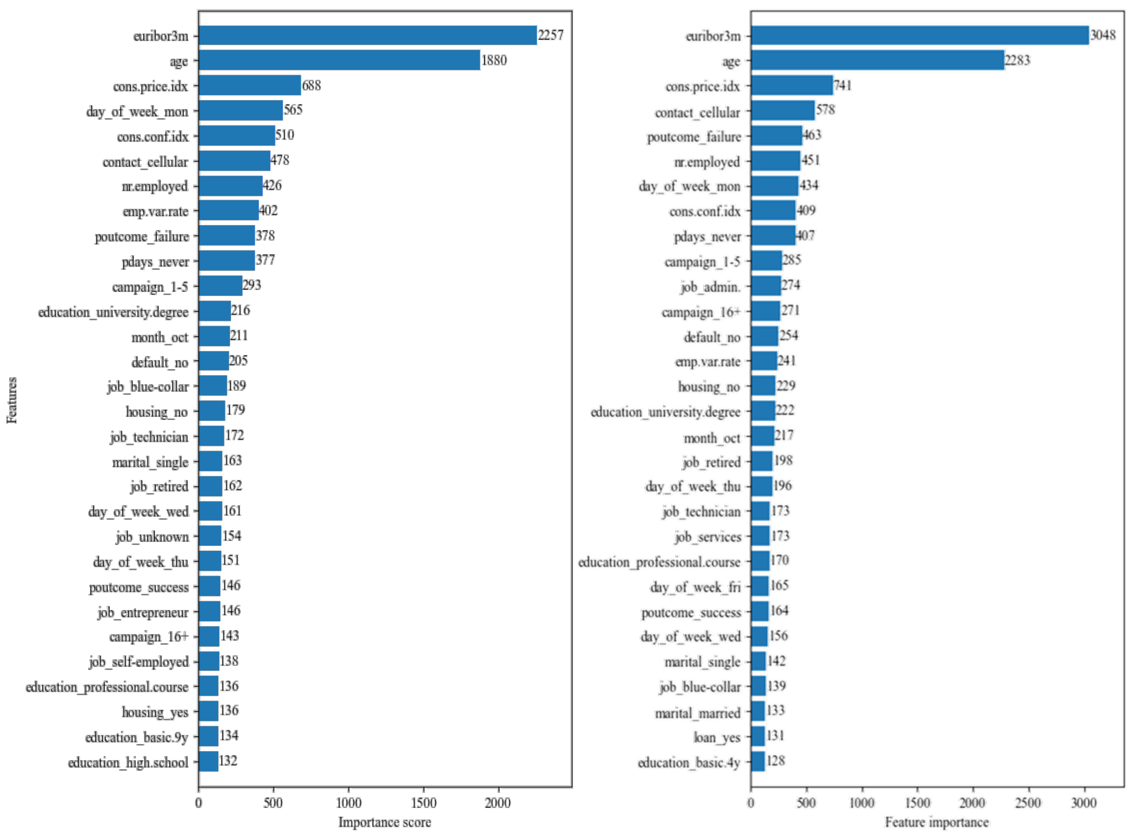

Fig. 5　Feature Importance of XG Boost (left) and Light GBM (right)

5 Discussion

Learning Curve Learning curve for all 3 boosting models are plotted as **Fig 6-8**, with various learning rate in 0.01, 0.1 and 0.3. We can see that all the models tend to reduce the train binary loss, while the validation loss levels after a certain iteration. That is where overfitting begins to take place. Before that epoch, the validation loss is not reaching its possible minimum, then the bias is large and the model underfits. The gradient boosting behaves slightly different (oscillates more drastically than) the others. Early stopping can be observed in those plots having smaller *x*-axis intervals. Note that under the same learning rate (e.g. 0.3), the Light GBM tends to learn more and stops later, XG Boost stops a little later as well, and the gradient boosting stops too early, probably learnt no patterns from the data.

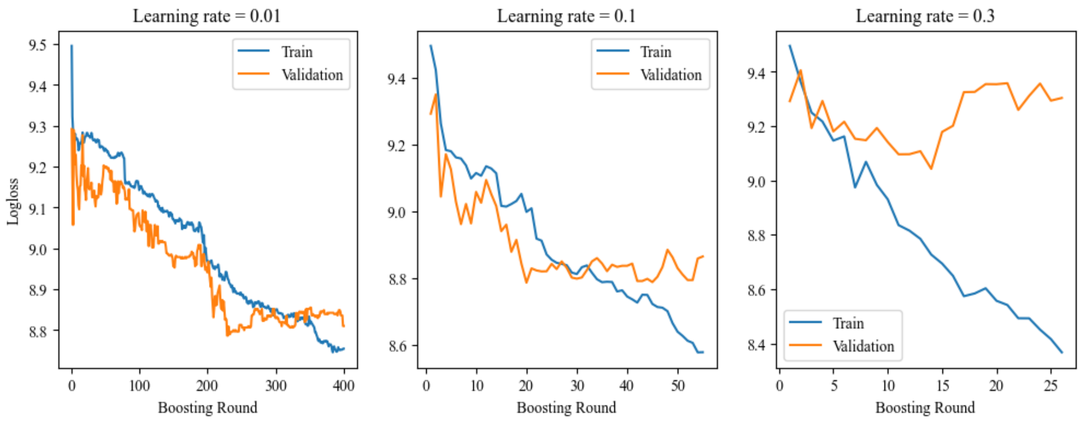

Fig. 6 Learning Curve of Gradient Boosting

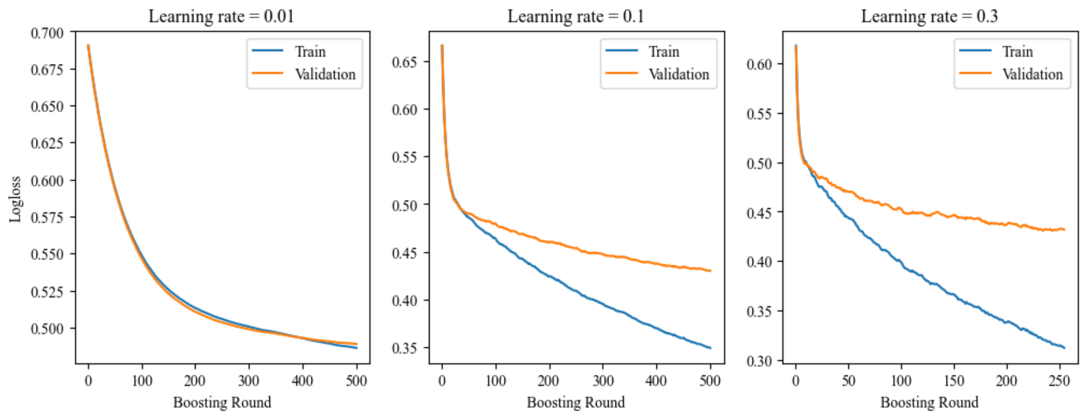

Fig. 7 Learning Curve of XG Boost

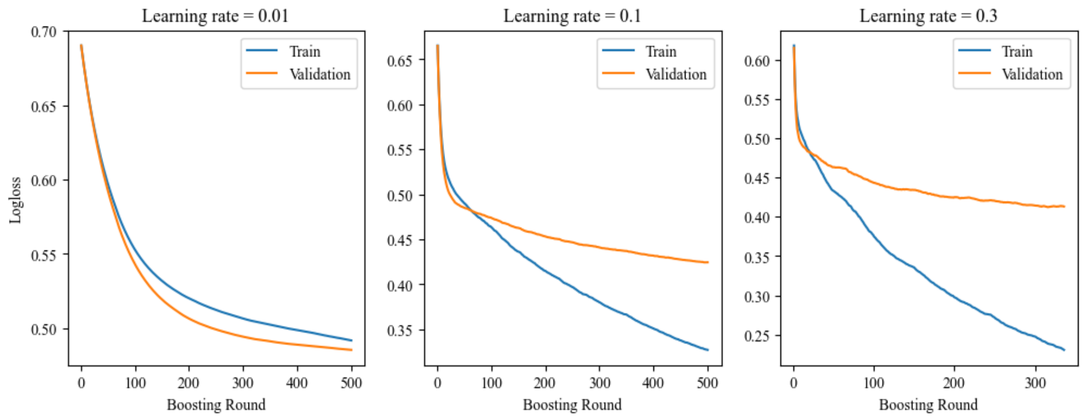

Fig. 8 Learning Curve of Light GBM

The learning rate seems to affect the models' performance in the way that it decides whether the model could learn more and reach its maximum iteration rather than early stopping. Take Light GBM as an example. When the learning rate is too small, e.g. 0.01, we could not see the full potential of the model, and when we raise the learning rate to 0.1, we see that the validation loss can be further reduced. This indicates lower learning rate makes models prone to underfitting, having higher bias. In boosting methods, this is to say that the number of minor estimators are not enough to study the mistake made by previous ones. On the other hand, higher learning rate induces mild oscillation in the learning curve. Thanks to early stopping that even with higher learning rate, the model could converge to a probable minimal validation loss. However, the gap between validation loss and training loss strongly indicates the model is overfitting at this stage, this could be explained as the mistake corrected by the new learner is almost the same to cancel out as the mistake introduced by itself. Thus it could have high variance and predict inconsistently from datasets to datasets.

Impact from Counts of Estimators We further discuss on how counts of estimator, i.e. the hyperparameter *n_estimators* impact on the models' performance, which can be deemed as a more concrete illustration of bias-variance trade-off: the counts of estimator directly reflects the model's complexity. We obtained the metric scores on train and test set by training a new model for each chosen value of *n_estimators* through the settings same to that in Section 3, **Fig 9-11** shows how the 6 metrics varied from fewer estimators to more in the 3 boosting models respectively.

We can see from these plots, that for F1, Precision, Recall and Accuracy, it is not guaranteed that more estimator results in higher metric scores. In general, the recall is increasing with more estimators introduced, this is probably due to that the model is growing confident in telling positiveness with newer estimators pay more attention on previous mistake made on positive samples.

The AUC ROC and AUC PR curves follow a general pattern: the score on train set is always increasing, while that on test set increases for a while as we begin to add newer estimators, and suddenly stops growing anymore, oscillates around its expectation value. This is telling us the models are overfitting, growing its variance and no more gains generalization score after a certain amount of estimators put in. The twist of the curves almost all take place at around *n_estimators*==100, where AUC reaches its maximum. This number can be considered as a balance point made between computational cost and complexity, and model accuracy and generalizability.

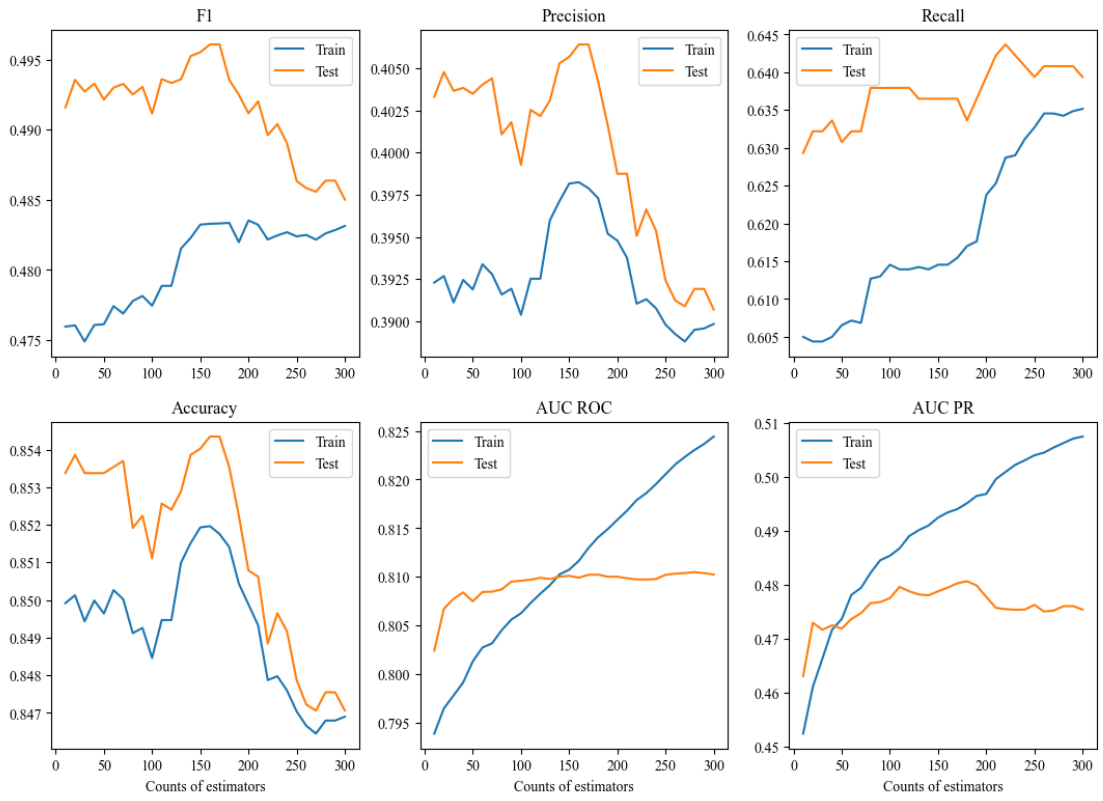

Fig. 9 Performance of Gradient Boosting as Functions in Counts of Estimators

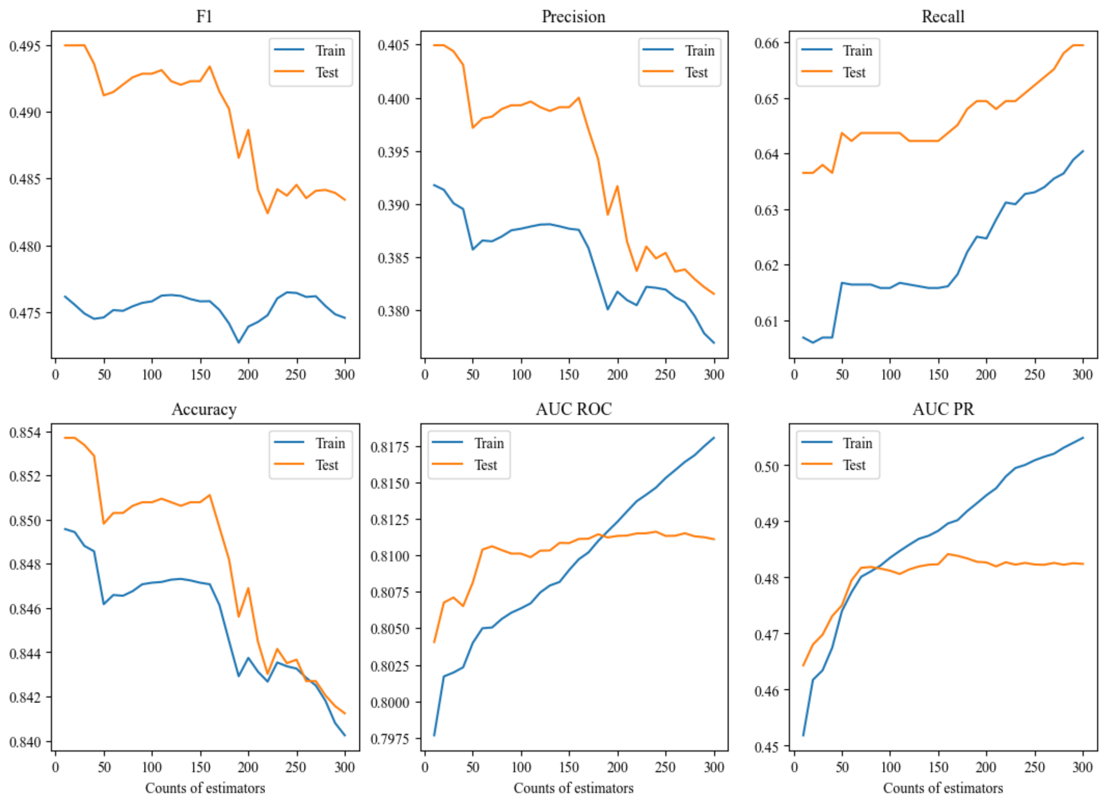

Fig. 10 Performance of XG Boost as Functions in Counts of Estimators

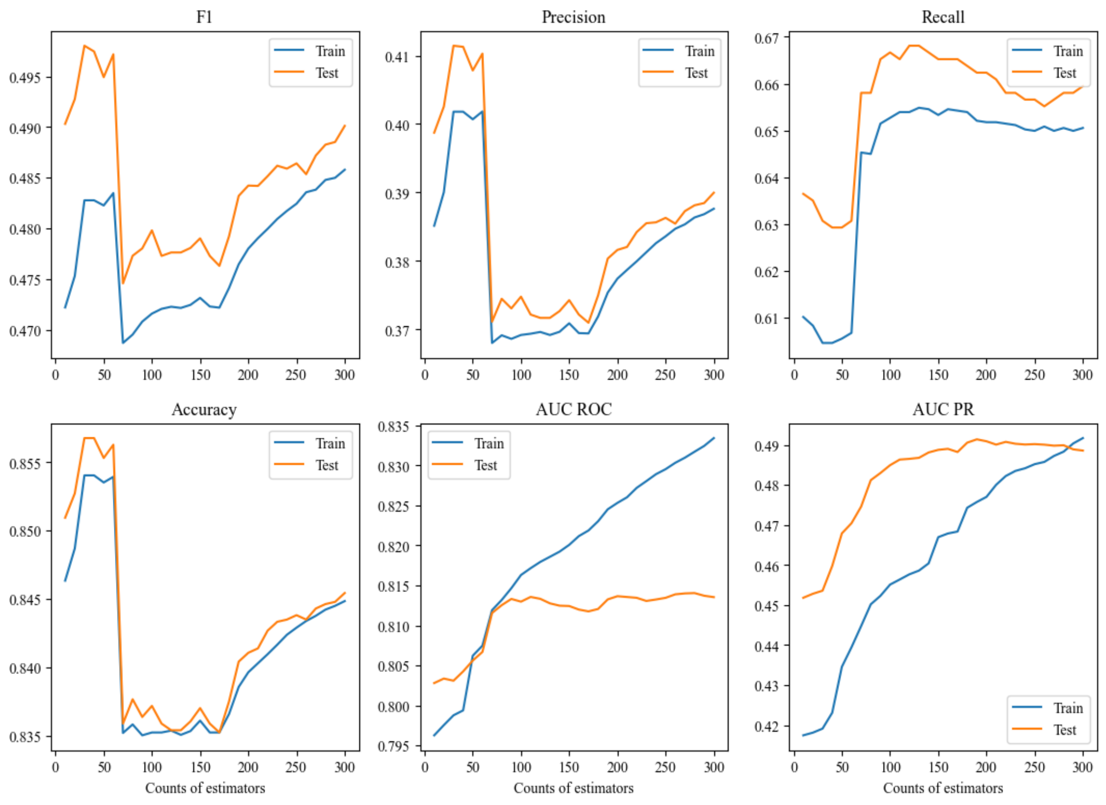

Fig. 11 Performance of Light GBM as Functions in Counts of Estimators

AI Tool Usage Disclosure

AI Tools Used ChatGPT (OpenAI, GPT-4 Configuration)

AI Contribution ChatGPT are used to assist in writing part of the introduction, abstract of this report.

Personal Contribution The author has written almost all of this report, all parts of his code on his own. No AI assistant like Copilot was used in code writing. Some parts are inspired by the online documentation and forum posts about Scikit Learn and other package libraries. Some parameters are discussed with friends of the author.