國立臺灣大學管理學院商學研究所

碩士論文

Graduate Institute of Business Administration

College of Management

National Taiwan University

Master Thesis

LightGBM與CatBoost在類別資料集下之效能探討

A Study on Performance of LightGBM and CatBoost under categorical datasets

邵立瑜

Li-Yu Shao

指導教授：蔣明晃 教授

Advisor: David Ming-Huang Chiang, Professor

中華民國109年6月

June, 2020

1. 誌謝

能完成這篇論文，我要特別感謝我的指導教授蔣明晃老師，以及Kaggle、T-brain等數據平台提供的資料集，還有父母、家人對我的支持。謝謝大家。

1. 中文摘要

對於現今中小型的資料集，梯度提升決策樹演算法(GBDT)在業界、學術界以及競賽被廣泛應用，此篇論文目的為比較目前最常使用的兩個GBDT套件，LightGBM與CatBoost，並找出兩個演算法之間效能差異的原因。為了讓比較具有公平性與一致性，我們根據一般現有真實資料集的特性設計了一個實驗，並根據此實驗的限制尋找資料集。實驗結果指出CatBoost在類別欄位較多的資料集確實預測效果更佳，而LightGBM則傾向於使用數值欄位來預測。在訓練時間上，LightGBM恆比CatBoost來的迅速。

關鍵字：梯度提升決策樹演算法、LightGBM、CatBoost

1. ABSTRACT

On medium-sized datasets, Gradient Boosting Decision Tree(GBDT) methods have been proven to be effective both academically and competitively. This paper aims to investigate and compare the efficiency of the two most used GBDT methods, LightGBM and CatBoost, and discover the reason behind the performance difference. To make a fairer comparison, we designed an experiment based on data characteristic, and found several desirable raw datasets accordingly. The implementation indicates that CatBoost tends to perform better when the dataset has indeed more categorical columns, while LightGBM incline to use numerical columns to predict. For training speed, LightGBM is always faster than CatBoost under all circumstances.

*Keywords*：Gradient Boosting, LightGBM, CatBoost.

1. CONTENTS

口試委員會審定書 #

[誌謝 i](#_Toc42863611)

[中文摘要 ii](#_Toc42863612)

[ABSTRACT iii](#_Toc42863613)

[CONTENTS iv](#_Toc42863614)

[LIST OF FIGURES vii](#_Toc42863615)

[LIST OF TABLES viii](#_Toc42863616)

[Chapter 1 Introduction 1](#_Toc42863617)

[1.1 Motivation 1](#_Toc42863618)

[1.2 Objective 2](#_Toc42863619)

[1.3 Organization of thesis 2](#_Toc42863620)

[1.4 Limitations 3](#_Toc42863621)

[Chapter 2 Related Work 4](#_Toc42863622)

[2.1 Boosting Methods 4](#_Toc42863623)

[2.2 Categorical Encoding 6](#_Toc42863624)

[Chapter 3 Research Methodology 8](#_Toc42863625)

[3.1 Research flow 8](#_Toc42863626)

[3.2 Experimental Design and Performance metrics 9](#_Toc42863627)

[3.2.1 Control Variables 9](#_Toc42863628)

[3.2.2 Evaluation Metrics 11](#_Toc42863629)

[3.3 Datasets 12](#_Toc42863630)

[3.3.1 Titanic: Machine Learning from Disaster[17] 12](#_Toc42863631)

[3.3.2 Cat in the Dat: Categorical Feature Encoding Challenge[18] 13](#_Toc42863632)

[3.3.3 Bank Marketing UCI[19] 13](#_Toc42863633)

[3.3.4 E-Sun Bank Fraud Detection[20] 16](#_Toc42863634)

[3.3.5 Data Preprocessing 17](#_Toc42863635)

[3.3.6 Hyperparameters 18](#_Toc42863636)

[Chapter 4 Results of our experimental design 20](#_Toc42863637)

[4.1 Titanic: Machine Learning from Disaster 20](#_Toc42863638)

[4.1.1 LightGBM (Self-made train(0.75)/test(0.25) on initial training sets) 20](#_Toc42863639)

[4.1.2 CatBoost (Self-made train(0.75)/test(0.25) on initial training sets) 21](#_Toc42863640)

[4.2 Cat in the Dat: Categorical Feature Encoding Challenge 22](#_Toc42863641)

[4.2.1 LightGBM(Kaggle) 22](#_Toc42863642)

[4.2.2 CatBoost(Kaggle) 23](#_Toc42863643)

[4.3 Bank Marketing UCI 23](#_Toc42863644)

[4.3.1 LightGBM(Kaggle) 24](#_Toc42863645)

[4.3.2 CatBoost(Kaggle) 24](#_Toc42863646)

[4.4 E-Sun Bank Fraud Detection 26](#_Toc42863647)

[4.4.1 LightGBM(Self-made train(0.75)/test(0.25) on initial training sets) 26](#_Toc42863648)

[4.4.2 CatBoost(Self-made train(0.75)/test(0.25) on initial training sets) 26](#_Toc42863649)

[4.5 Summary (AUC & Private Score) 28](#_Toc42863650)

[4.5.1 Training speed 28](#_Toc42863651)

[4.5.2 Performance 28](#_Toc42863652)

[Chapter 5 Conclusion 31](#_Toc42863653)

[5.1.1 Summary: 31](#_Toc42863654)

[5.1.2 Contribution: 31](#_Toc42863655)

[5.1.3 Limits: 32](#_Toc42863656)

[5.1.4 Future studies: 32](#_Toc42863657)

[REFERENCE 34](#_Toc42863658)

1. LIST OF FIGURES

[Fig. 3‑1 Research Flowchart 9](#_Toc42863659)

[Fig. 3‑2 Hyperparameter Settings 19](#_Toc42863660)

1. LIST OF TABLES

[Table 3‑1 Summary of the Experimental Design 11](#_Toc42698732)

[Table 3‑2 Titanic Dataset Description 12](#_Toc42698733)

[Table 3‑3 E-Sun Dataset Description 16](#_Toc42698734)

[Table 4‑1 Summary of the Experimental Design 28](#_Toc42698735)

[Table 4‑2 Summary of Feature Importance 29](#_Toc42698736)

# Introduction

## Motivation

Data drives the modern organizations of the world, and thus making sense of data, unraveling various patterns, and revealing unseen connections within them becomes a critical endeavor.

Long existence and development of Rational Database Management System (RDBMS) has made data comes in a tabular form. In this structured form, we can often see some columns emerge as qualitative value. It is essential that we take good care of these values for two reasons: (1)Most existing algorithms cannot handle qualitative values, we have to encode them before feeding them into our model; (2)Different encoding styles will result in rather drastic influences on model performance. Besides, Discrete categorical columns are often the bottleneck of the predictive model on tabular datasets. The way to process them will heavily influence the model performance.

In data analysis, Gradient Boosting Decision Tree(GBDT) methods and its derivatives are well known for its effective performance on medium-sized tabular datasets[1]. In most of the tabular datasets, categorical columns often appear multiple times and some of them are binary-valued. Catboost states that it outperforms LightGBM and XGBoost by introducing a new algorithm of preprocessing categorical features [2], using various datasets that is primarily based on data size and feature size.

It is a continuously debate about whether one algorithm performs better than another algorithm. Therefore, various experiments were conducted to test their performance. Unfortunately, different conclusions are revealed. In this thesis, the two existing methods: LightGBM and Catboost, that have been proven to be successful in industry, academia, and competitive model building, will be compared. The reason that we exclude XGBoost is because LightGBM is developed to improve from XGBoost and often there is little difference between the two algorithms. We hope to explore some physical characteristics of datasets that might identify the situation that one algorithm performs preferably than another.

## Objective

The two methods mentioned previously are relatively new, although heavily used in recent data science field. There have been numerous researches of algorithms’ efficiency, but how the models work on real-life scenario datasets are seldomly discussed. This thesis aims to achieve three points below:

* Explore possible characteristics of the tabular datasets that would affect the boosting methods’ performance. Also, datasets that best suited to our experimental design are identified.
* Build up LightGBM and CatBoost models on these real datasets selected based on our criteria with common conditions(hyperparameters) and evaluate their performance accordingly.
* Conclude our results based on characteristics of the tabular datasets.

## Organization of thesis

The rest of the thesis is presented as follows: Chapter 2 summarizes some existing prevalent methods and the recent developments. Chapter 3 the experimental design framework for the datasets is built up, and selected datasets are introduced briefly. In Chapter 4, the two boosting method studied in this thesis are implemented, and comparison based on their performance under different characteristics of the tabular datasets is discussed. Chapter 5 will conclude our finding and propose some future works needed to be further studied.

## Limitations

1. We used dataset that have only numerical and categorical columns. Some of the categorical columns are string-based values. Even though sometimes using NLP (Natural Language Processing) to preprocess these columns may produce better results which may influence algorithm performance. In this thesis, we only do categorical encoding for this research.
2. Dataset volume varies between each dataset. From past studies, small datasets tend to more unstable in their performance since there is little data to learn from, and small validation/test data is susceptible to noises. Because datasets we chosen come from real life scenarios, and we have no way to control the dataset’s volume. Therefore, we do not consider dataset volumes as our selection criteria when we select datasets.
3. Because one of our characteristics of the tabular datasets is related to binary attribute, all our datasets are chosen to have binary characteristics. Our research did not use a dataset with multiclass classification problems.

# Related Work

## Boosting Methods

In Machine Learning, gradient boosting is a method that capitalizes on the combination of Weak Learners, and iteratively improves the model by learning from the error produced from previous run. To reduce error with a systematic manner, Gradient Descent is often used on differentiable loss function to find the local minimum of errors.

In Gradient Boosting method, loss function is defined on the error resulted from Boosting, and then Gradient Descent is used to find the local minimum, iteratively by adding weak learners. In order to utilize the large volume of data, researchers apply a more efficient way to find the gradient to replace the actual gradient with an estimated one, the prevalent method being Stochastic Gradient Descent(SGD). In this method, each iteration learns by fitting the negative gradients (residual errors).

Back in the days, an implementation of Gradient Boosting Decision Tree(GBDT), Gradient Boosting Machine, was a breakthrough in predictive analysis. In general, choosing a parameterized model changes the function optimization problem to one of parameter optimization

where

and then

In the case of finite data, cannot be estimated accurately. One way to solve this is to assume a parameterized form and do parameter optimization to minimize the corresponding data-based estimate of expected loss

In Gradient Boosting, it uses a greedy stagewise approach. For m = 1,2,,M

and then

More mathematical explanation and proof is on [3][4].

There have been many implementations of GBDT. As shown in [5], XGBoost outperforms the traditional tools [7][8][9]. Since LightGBM improves on XGBoost [1], we can safely assume that at least both boosting methods’ performance are similar. In [10] LightGBM even outperforms both XGBoost and CatBoost on a home credit dataset.

In LightGBM, the author proposes two new ways called “Gradient-based One-Side Sampling” and “Exclusive Feature Bundling” to effectively reduce the number of features, improving the model training speed (always significant faster than XGBoost while having similar performance) and memory consumption.

CatBoost aims to solve the problem called *prediction shifts*, caused by a special kind of target leakage which uses the “target column” to compute *target statistics*, which is a commonly used categorical encoding technique. The author purposed ordered boosting, a modification of standard gradient boosting algorithm, to avoid target leakage.

In this thesis, we will conduct both boosting algorithms on our datasets, and compare them to see if one performs better in terms of categorical-heavy datasets.

## Categorical Encoding

Since most of the questionnaires are for qualitative analysis, for example, consumer behavior, it is quite common that questionnaires produce categorical data. Sometimes, even the quantitative ones are encoded to discrete values. Income, for instance, are often recorded in ordinal ways, to either make answerer more likely to respond, or make the recorder more comfortably to record.

Taking a glance at some [Kaggle](https://www.kaggle.com/datasets) tabular datasets(e.g. recent IEEE fraud detection dataset), it is common that tabular data consists mostly of categorical columns. This is the primary reason why the experimental design of this thesis uses categorical column percentage as control variable.

In most of the machine learning settings, the model can only handle numerical columns. It is essential we encode categorical columns before we start training. Categorical variables can be divided into two kinds, based on Statistics: Nominal and Ordinal.

There have been various categorical variable encoding techniques [11], the most widely used one being One Hot Encoding. With this method, we map each category to a vector of 1 and 0, denoting the presence of the feature. This method will produce a lot of columns if the given categorical variable’s number of category is very high, causing the well-known The Curse of Dimensionality [12].

In LightGBM, it is recommended to use integer-encoded categorical features [15]. We will use Ordinal Encoding (Label Encoding) to preprocess the data for LightGBM.

Catboost doesn’t require to encode the categorical columns, as it preprocess them on its own, using a variant of Sum Encoding (also known as Target/Mean Encoding) [2].

# Research Methodology

## Research flow

In order to explore some physical characteristics of datasets that might identify the situation that one algorithm perform preferably than another, we first need to construct a framework of experimental design by determining control variables.

Next, we search for various online datasets to determine the datasets that have the attributes we want under our experimental design framework. Then we will do some data preprocessing to make the data suitable for machine learning models.

After all the preparation, we will start training the model. To simplify the interpretation, we do no hyperparameter tuning. In the end, we will choose a performance metric to evaluate our results. Figure 3.1 illustrate our research flow.

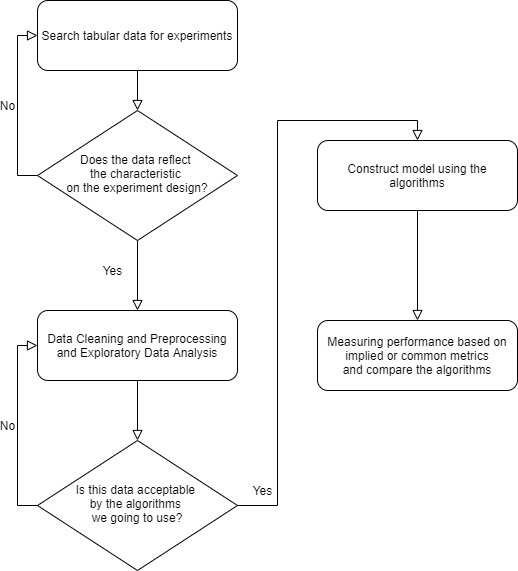


Fig. 3‑1 Research Flowchart

## Experimental Design and Performance metrics

In this section, we will define our experiment, and briefly introduce performance metrics we are going to use.

### Control Variables

The first control variable is the percentage of categorical columns out of all columns in a dataset. We want to see how the performance will be influenced based on how much of the original data are categorical.

The second control variable will be the percentage of binary-valued categorical columns out of all categorical columns. In this setting, we ought to see how the cardinality of the columns affect our model, briefly separated by the cardinality of two or more.

In order to control other factors which may influence the performance of algorithms, we will apply the same preprocessing method, train/validation split style, and default hyperparameter for each dataset (learning rate = 0.1), with 1000 iterations(epochs). We will also do an early stop if the validation does not improve after 100 iterations.

After we search for various online datasets, the final data sources we going to use are from Kaggle platform, or the recent Taiwan data competition platform T-brain. Since Kaggle platform already split the test data for us, and we can submit our predictions to get a score from the platform, the performance of the 2 datasets that use AUC as evaluation from Kaggle will be evaluated from Kaggle online submission score. As for the E-Sun dataset, since T-brain prohibits submission after competition, we will mimic the train/test split style that was presented in the competition and evaluate them accordingly. Titanic’s competition is evaluated on Accuracy, so we will have to manually split Titanic data into train/test sets if we want to get the AUC score.

The following table is our summary of experimental design:

|  |  |  |
| --- | --- | --- |
|  | Percentage of categorical variables <=70% | Percentage of categorical variables >70% |
| Percentage of binary variables <= 25% | Titanic Dataset  55.56% cat, 20% bin | Cat in the dat dataset  100% cat, 17.39% bin |
| Percentage of binary variables > 25% | Bank marketing dataset  62.5% cat, 30% bin | E-Sun fraud dataset  85.71% cat, 27.78% bin |

Table 3‑1 Summary of the Experimental Design

### Evaluation Metrics

Intuitively, we often use accuracy to evaluate a model. But in real time applications, there are large amount of data generated with skewed distributions. This phenomenon is called “Class Imbalance”[13].

We will use AUC for the performance evaluation, not only it alleviate the Class Imbalance problem, but also it is widely accepted that it is an overall better metric than accuracy[14]. On Kaggle competitions that use AUC as performance metric, we will submit the predictions made from our model and compare the results with others online.

Note that by using AUC score we also eliminate the process of finding the best threshold. Since our models output probability, and accuracy (or F1-score) needs discrete value 0 or 1 to compute, we will need to find a threshold, for example, threshold = 0.3 means that if the probability is lower than 0.3 then it will be classified as negative. This will make us harder to compare algorithms since we might have to use the same threshold for algorithms, but the threshold will also influence algorithm performance, and using a certain threshold might favor an algorithm.

AUC compares probability directly since it will make several thresholds to compute the area under the ROC curve, and thus gives a fairer comparison between algorithms.

## Datasets

In the following section, we will take out ID column (since using ID to predict makes no sense in terms of modeling), and the target column (the column we want to predict), before we compute the percentage of categorical variables.

### [Titanic: Machine Learning from Disaster](https://www.kaggle.com/c/titanic)[17]

The aim of this dataset is to predict if a survivor survived the Titanic shipwreck.

Official dataset number of records: 891 train, 418 test

Self-made train/test: 668 train, 223 test (using 668+223 = 891 train dataset)

This dataset records some basic information of passengers, and a target column “survival” to indicate if a passenger survived the Titanic shipwreck.

Below is the data description directly copied from the Kaggle platform:

|  |  |  |
| --- | --- | --- |
| Variable | Definition | Key |
| survival | Survival | 0 = No, 1 = Yes |
| pclass | Ticket class | 1 = 1st, 2 = 2nd, 3 = 3rd |
| sex | Sex |  |
| Age | Age in years |  |
| sibsp | # of siblings / spouses aboard the Titanic |  |
| parch | # of parents / children aboard the Titanic |  |
| ticket | Ticket number |  |
| fare | Passenger fare |  |
| cabin | Cabin number |  |
| embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

Table 3‑2 Titanic Dataset Description

Excluding the ID and the target column, we have:

5 categorical columns out of 9 columns (55.56%)

1 binary columns out of 5 categorical columns(20%)

The original competition uses Accuracy as performance metric, so we will not use the test set from the competition, since we can’t access the true label of the test set and we want to get AUC results.

### [Cat in the Dat: Categorical Feature Encoding Challenge](https://www.kaggle.com/c/cat-in-the-dat/overview)[18]

The dataset is to predict the binary target column, which the meaning is unknown to modelers.

Official dataset number of records: 300,000 train, 200,000 test

The provider does not give much description about the data, only stating those columns characteristics. All are categorical columns, including: (copied from the Kaggle platform)

* Binary features
* Low-and high-cardinality nominal features
* Low-and high-cardinality ordinal features
* (potentially) cyclical features

Excluding the ID and the target column, we have:

23 categorical columns out of 23 columns (100%)

4 binary columns out of 23 categorical columns(17.39%)

### [Bank Marketing UCI](https://www.kaggle.com/c/bank-marketing-uci/overview)[19]

The dataset tries to predict if the client will subscribe a term deposit, which is also a binary classification problem.

Official dataset number of records: 4521 train, 427 test

The goal is to predict if the client will subscribe a term deposit.

Data description copied and reorganized from the document attached with the datasets:

* Input variables:

1 - age (numeric)

2 – job (categorical)

3 - marital : marital status (categorical)

4 - education (categorical: "unknown","secondary","primary","tertiary")

5 - default: has credit in default? (binary: "yes","no")

6 - balance: average yearly balance, in euros (numeric)

7 - housing: has housing loan? (binary: "yes","no")

8 - loan: has personal loan? (binary: "yes","no")

9 - contact: contact communication type (categorical)

10 - day: last contact day of the month (numeric)

11 - month: last contact month of year (categorical)

12 - duration: last contact duration, in seconds (numeric)

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)

15 - previous: number of contacts performed before this campaign and for this client (numeric)

16 - poutcome: outcome of the previous marketing campaign (categorical)

* Output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: "yes","no")

Excluding the ID and the target column, we have:

10 categorical columns out of 16 columns (62.5%)

3 binary columns out of 10 categorical columns(30%)

### [E-Sun Bank Fraud Detection](https://tbrain.trendmicro.com.tw/Competitions/Details/10)[20]

The dataset aims to predict if the transaction is a fraud or not, which is also a binary classification.

Official dataset number of records: 1,521,787 train, 421,665 test

Self-made train/test: 1,014,791 train, 506,996 test (using 1,014,791+506,996 = 1,521,787 train dataset)

The dataset contains every transaction record with its nature and occasion, and a target column that indicate this transaction is a fraud or not.

Description re-organized below:

|  |  |
| --- | --- |
| Variable name | Description |
| bacno | The account that makes this transaction |
| txkey | Transaction ID |
| locdt | The date of authorization |
| loctm | The time(hh/mm/ss) of authorization |
| cano | Credit card number |
| contp | The type of transaction |
| etymd | The type of transaction |
| mchno | The code name of special store |
| acqic | The code name of receiving bank |
| mcc | Mcc code |
| conam | Transaction amount |
| ecfg | Indicate if it is from online or offline(binary) |
| insfg | Indicate if it is installment(binary) |
| iterm | Installment number of periods |
| stocn | The country that transaction take placed |
| scity | The city that transaction take placed |
| stscd | Status code |
| ovrlt | Indication of above the quota amount(binary) |
| flbmk | Indication of Fallback(binary) |
| hcefg | Payment type |
| csmcu | Currency type which the transaction country uses |
| flg\_3dsmk | Indication of 3DS transaction(binary) |
| fraud\_ind | Whether this transaction is a fraud(target) |

Table 3‑3 E-Sun Dataset Description

Excluding the ID and the target column, we have:

18 categorical columns out of 21 columns (85.71%)

5 binary columns out of 18 categorical columns (27.78%)

On this particular dataset, the original competition was using 90 days as training data, and 30 days as test data (which the platform doesn’t provide the true label), totaling 120 days data. Since we are unable to access the test data evaluation from the platform anymore, we will mimic the competition’s train/test spit style by using the 90 days training data, and split them into 60 days training data and 30 days test data, using the 30 days data to evaluate our models.

### Data Preprocessing

* Missing Values

Since we compare two methods, we make sure to follow the same rule to process data for different datasets. The only problem we must deal with is that some data sources have missing values.

We approach them differently based on if the column is numerical or categorical.

1. Numerical column values will be replaced by the average of the column.   
   If a dataset has only 1 column with value {3,(missing),4,5,}, the missing value will be substituted with (3+4+5)/3 = 4 .
2. Categorical column values will be replaced by a value that represents “Missing” category.   
   For example, a dataset with only 1 column with value {male, male, (missing), female, (missing)}, the missing value will be changed to “missing” before encoding them.

* Train/Validation dataset split

The train/validation split percentage will always be 0.75/0.25 for all datasets.

To further alleviate the Class Imbalance problem, when we do the train/validation split, we will always do stratify sampling based on the target variable percentage. If we randomly split the dataset, for example a binary classification problem, the validation data might have too little positive labeled data to learn, making the model tend to always predict negative.

### Hyperparameters

We will use early stop models, meaning if the validation loss function does not improve for n rounds, choose the model that gives the best validation score.

All our datasets are binary classification problems. We will mostly use default hyperparameters of both algorithms, with some exceptions:

1. Max iterations(epochs) = 1000:

This is the maximum number of iterations the model is going to train, and we set 1000 for this. For the default hyperparameters, it normally early stops before 1000 iterations, even if not, the validation score negligibly changes.

1. Learning rate = 0.1:

Learning rate is the amount the weights are updated during training, often coined as ". CatBoost default learning rate differs on the number of iterations, while LightGBM default learning rate is 0.1. We will force both algorithms to use 0.1.

1. Objective function (Loss function) = Log Loss:

This is the function that will be calculated and optimized during training, usually using validation’s loss to indicate whether early stop or not. Since we evaluate based on AUC, it is best to use some objective functions that directly maximize the tradeoff between True Positive Rate and False Positive Rate, but since it is non-differentiable and thus cannot be used by the algorithm, it is hard to decide which objective function best suits our research needs.

To keep things simple and not dive too deep into mathematical problems, we will use binary log-loss to optimize.

1. Early stopping rounds = 100

This is the number of rounds if the validation loss doesn’t improve for 100 rounds, the model will choose the round has lowest validation loss. Most of the time when it doesn’t improve for around 50 rounds, the validation loss rarely improves again. We will use 100 for both algorithms.

To summarize, we will clip a block of code that define our hyperparameter settings, using LightGBM’s notation (CatBoost has different name, but the mechanism is essentially the same.)



Fig. 3‑2 Hyperparameter Settings

# Results of our experimental design

In this chapter, we will first report results of each datasets by implementing two boosting methods by the following sequence: missing values processing, AUC of the model, training time, trained iterations before reaching minimum log loss, feature importance, and brief comment on two boosting methods. Finally, we reveal some observations based on our experimental design when two boosting methods are compared.

## [Titanic: Machine Learning from Disaster](https://www.kaggle.com/c/titanic)

Column “Age”, “Cabin” and “Embarked” have missing values on training dataset. After the self-made train/test split, we found “Age” and “Cabin” missing values on train set, and “Cabin” and “Embarked” on test set.

“Cabin” and “Embarked” are categorical columns, so we preprocess the missing values as “Missing”, exactly as before, making it another categorical value. As for “Age” column, we use the “Age” average of the self-made train set for both missing values of train and test set.

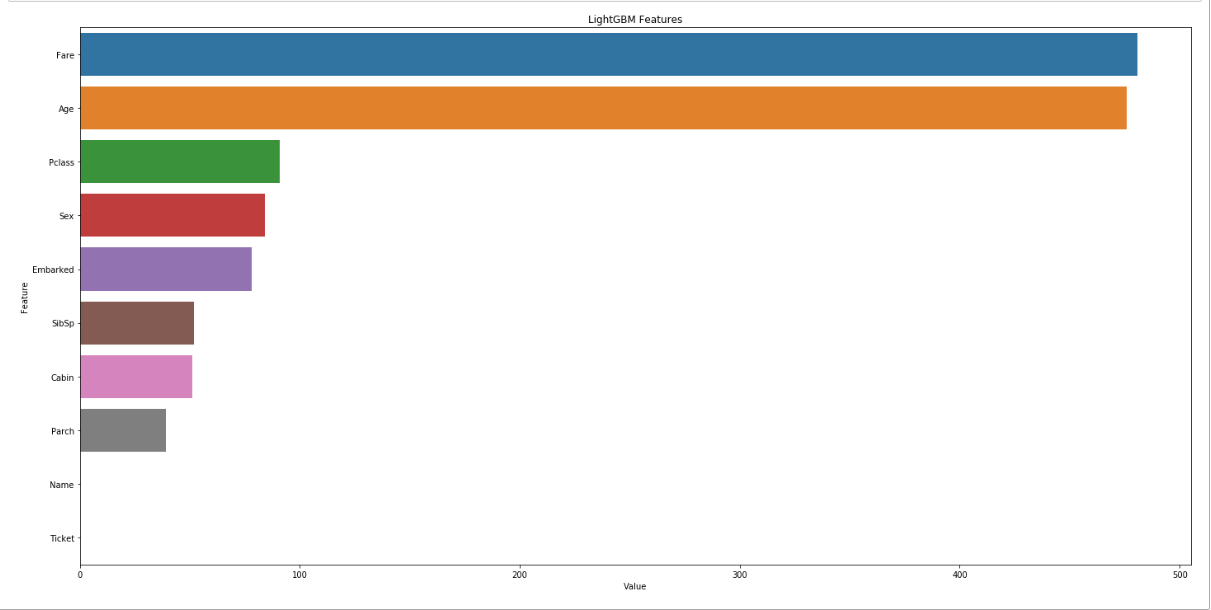
### LightGBM (Self-made train(0.75)/test(0.25) on initial training sets)

AUC = 0.85104

Training time: 0.388s

Early stop minimum log loss at iteration 95

Feature Importance:



The top 4 features are “Fare”, “Age”, “Pclass”, and “Sex”, with “Fare” and “Age” being two prominent features containing numerical values.

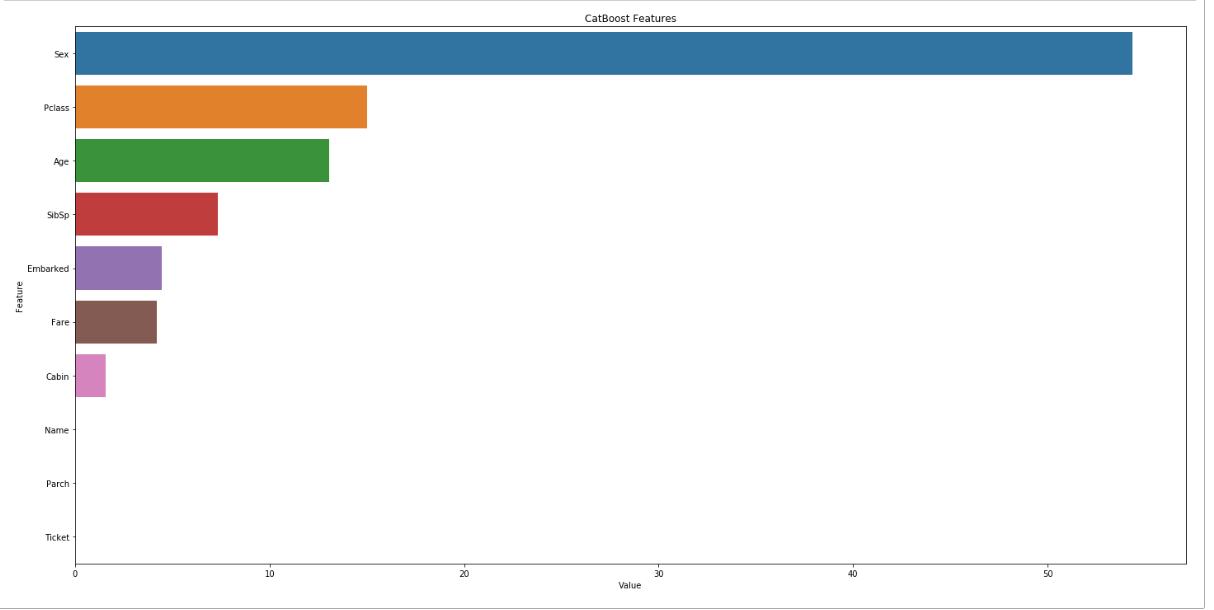
### CatBoost (Self-made train(0.75)/test(0.25) on initial training sets)

AUC = 0.86135

Training time: 3.79s

Early stop minimum log loss at iteration 113

Feature Importance:



The top 4 features are “Sex”, “Pclass”, ”Age” and “SibSp”. The most important one is “Sex”, being much more significant than any other features. “Sex” is also a categorical column.

In this dataset, based on the feature importance of two models developed by two methods, we can find that LightGBM tends to choose numerical columns as their main features, while CatBoost chooses categorical columns.

## [Cat in the Dat: Categorical Feature Encoding Challenge](https://www.kaggle.com/c/cat-in-the-dat/overview)

No missing value has to be preprocessed

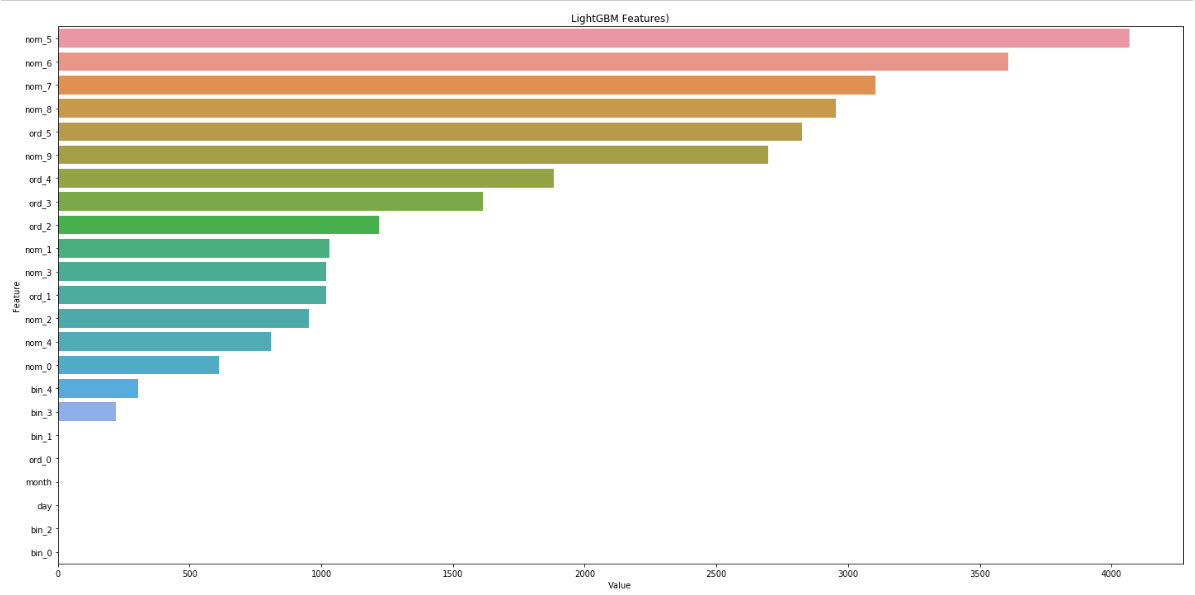
### LightGBM(Kaggle)

Public AUC = 0.75300, Private AUC = 0.75086

Training Time: 4 min 22s

No early stop, minimum log loss at iteration 998

Feature Importance:



The top 4 features are “nom\_5”, “nom\_6”, “nom\_7” and “nom\_8” with all nominal features.

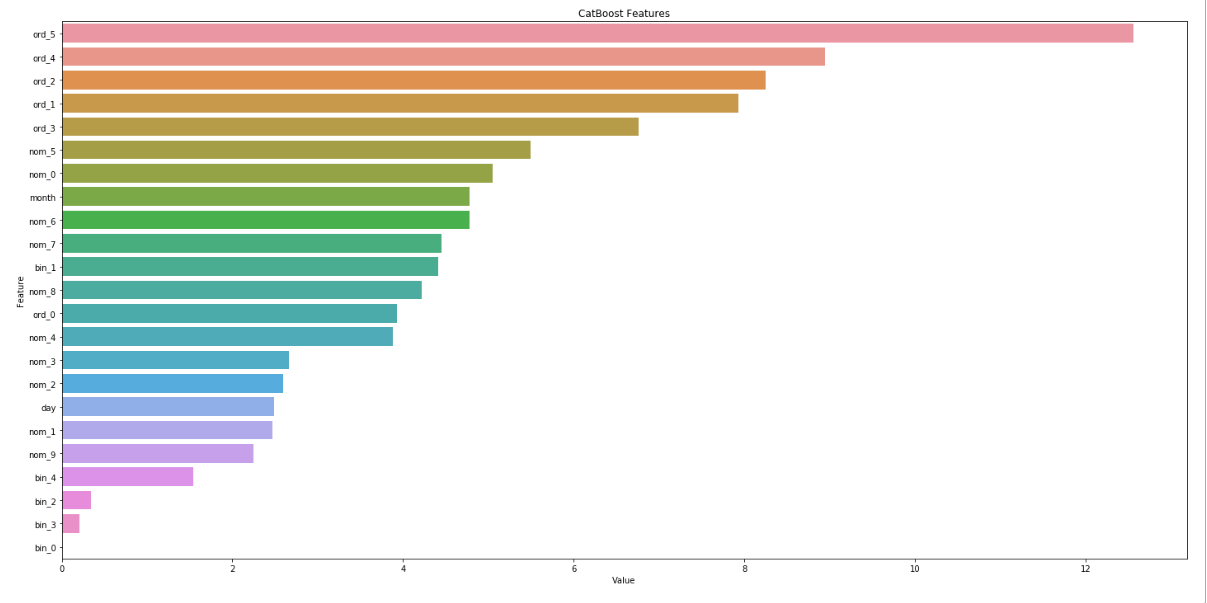
### CatBoost(Kaggle)

Public AUC = 0.80087, Private AUC = 0.79475

Training Time: 25min 17s

Early stop minimum log loss at iteration 862

Feature Importance:



The top 4 features are “ord\_5”, “ord\_4”, “ord\_2” and “ord\_1”, which are all ordinal features, instead.

For this dataset, CatBoost tends to utilize ordinal features, while LightGBM uses nominal ones, instead.

## [Bank Marketing UCI](https://www.kaggle.com/c/bank-marketing-uci/overview)

The missing value in this dataset is valued as “?” and scattered around all categorical columns, so we do not have to preprocess them since “?” will be regarded as a category in our encoding style.

For this particular dataset, since the Kaggle competition requires output to be only [0,1], we have to set a threshold in which we translate the output probabilities into 0 and 1. We simply set threshold to be the positive label ratio of the training dataset.

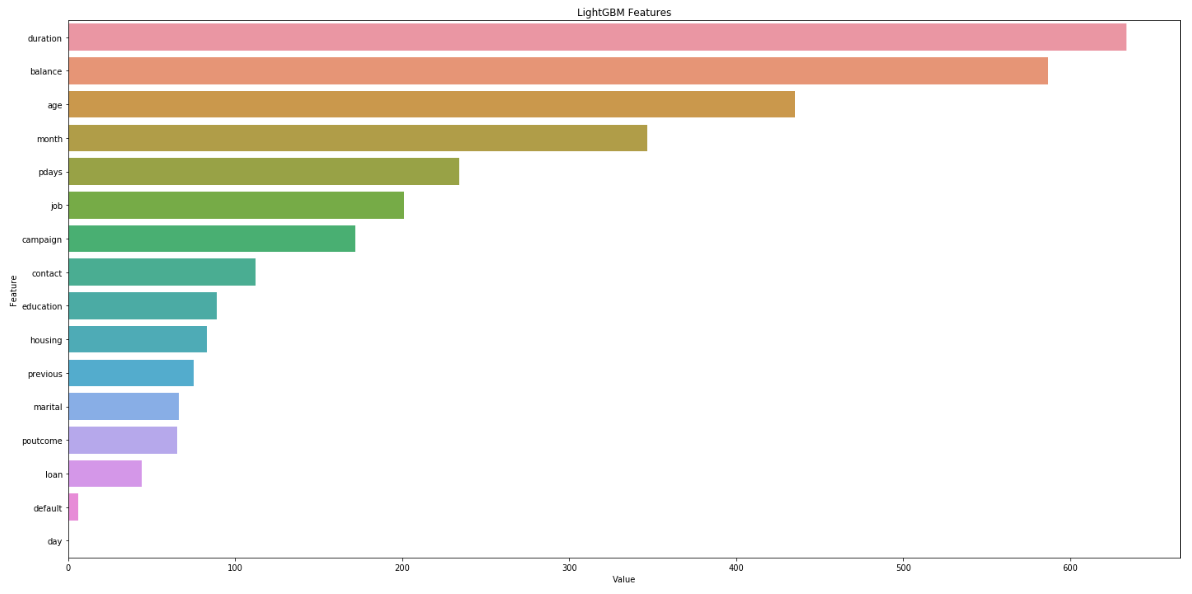
### LightGBM(Kaggle)

Public AUC = 0.81010, Private AUC = 0.87091

Training Time: 0.68 s

Early stop minimum log loss at iteration 105

Feature Importance:



The top 4 features are “duration”, “balance”, “age” and “month”, with “duration” and “balance” being a bit more prominent than the other two. The top 3 are also all numerical columns.

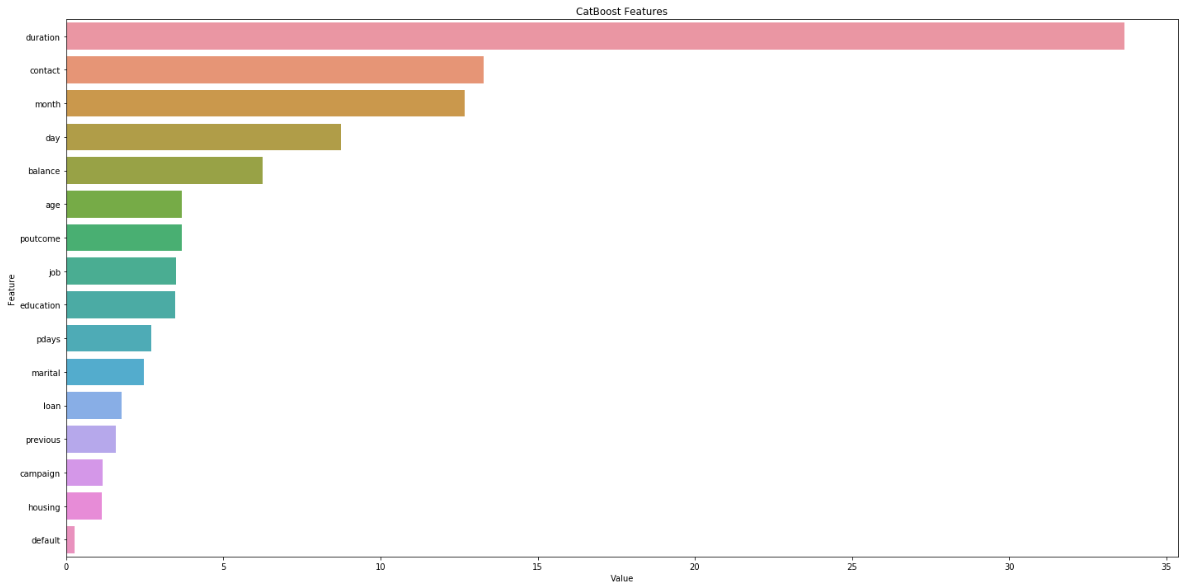
### CatBoost(Kaggle)

Public AUC = 0.78405, Private AUC = 0.78767

Training Time: 15.5s

Early stop minimum log loss at iteration 165

Feature Importance:



The top 4 are “duration”, “contact”, “month” and “day”, with “duration” being much more prominent than others. Note that “contact”, “month” and “day” are all categorical columns.

Column “duration” in data description describes as *last contact duration*, and it is straightforward that if the contact time is long, the person is more likely to subscribe the term deposit, hence the high feature importance of both models.

We can see again that LightGBM inclines to choose numerical columns to predict, while CatBoost uses categorical features. Since “duration” is an important feature in both models, and excluding that, we can still see the models’ tendency based on column’s nature.

## [E-Sun Bank Fraud Detection](https://tbrain.trendmicro.com.tw/Competitions/Details/10)

Binary column ‘flbmk’ and ‘flg\_3dsmk’ have some missing values (less than 10%), therefore, we preprocess them by assigning them value “Missing”, in effect, making it the third category value.

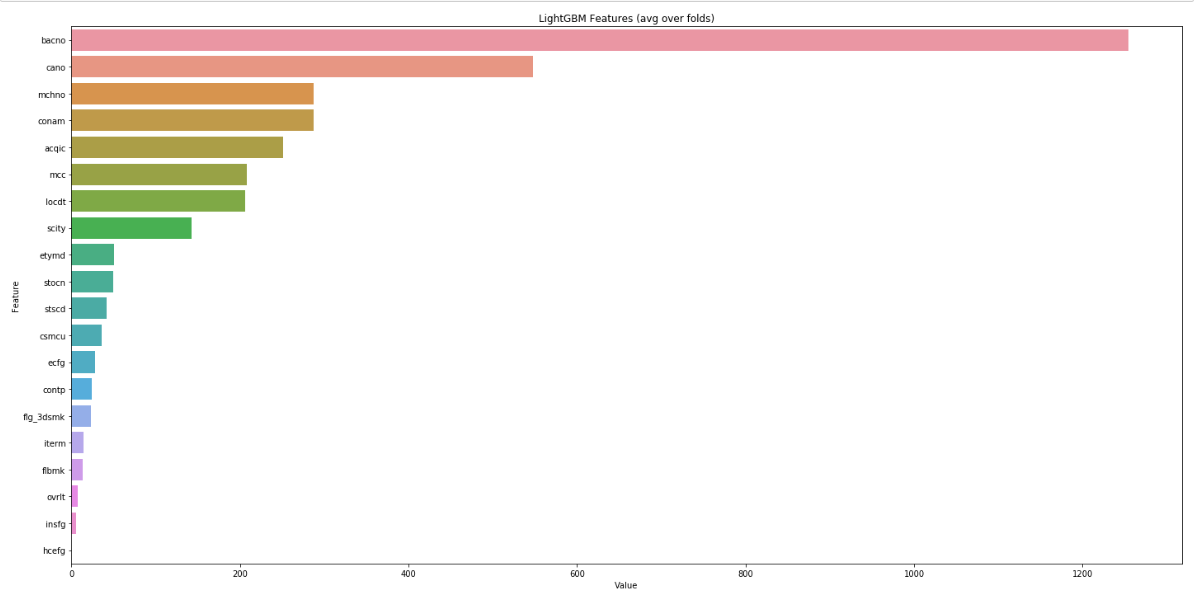
### LightGBM(Self-made train(0.75)/test(0.25) on initial training sets)

AUC = 0.96541

Training Time: 1 min 1 s

Early stop minimum log loss at iteration 116

Feature Importance:



The top 4 features being “bacno”, “cano”, “mchno” and “conam”, with “bacno” being much more prominent than others”. The top 3 are in categorical form, and “conam” is in numerical form.

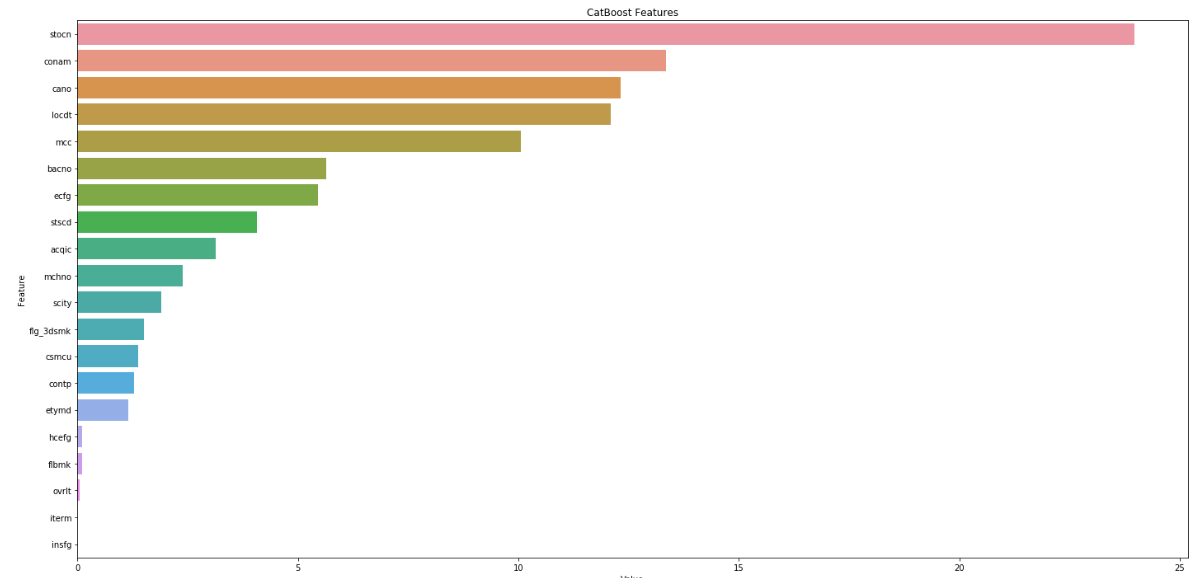
### CatBoost(Self-made train(0.75)/test(0.25) on initial training sets)

AUC = 0.97361

Training Time: 58min 14s

Early stop minimum log loss at iteration 860

Feature Importance:



The top 4 features are “stocn”, “conam”, “cano” and “locdt”, with “stocn” being much more prominent than others.

Note in this particular dataset, using default hyperparameters, CatBoost performs much more iterations to train (860 iterations) than LightGBM(116 iterations). By looking into the most prominent feature, we find that LightGBM uses the feature “bacno”, which means the bank account ID that makes the transaction while CatBoost uses the feature “stocn”, which is the country that the transaction take place. If the aim is to predict the new clients’ fraudulent transactions, CatBoost might be a better choice here, since for a new client, the bank account ID “bacno” is new and unseen, while with “stocn”, as long as the new client is from a country that the bank transactions regularly take place, the feature values almost always learned by the model.

## Summary (AUC & Private Score)

### Training speed

It is almost always the case that LightGBM is much faster than CatBoost in terms of model training.

Our guess is that because CatBoost preprocess categorical columns on its own predefined procedure, the process may be very time consuming for especially high cardinality ones. Since LightGBM requires users to manually encode categorical columns, this may be the reason why CatBoost is much slower.

Discussing why the big time difference might be a very complex issue, and in this thesis we will only can conclude that LightGBM’s training speed is always much faster than CatBoost for the four datasets.

### Performance

|  |  |  |
| --- | --- | --- |
|  | Percentage of categorical variables <=70% | Percentage of categorical variables >70% |
| Percentage of binary variables <= 25% | Titanic Dataset  LightGBM(85.10%)  CatBoost**(86.13%)** | Cat in the dat dataset  LightGBM(75.09%)  CatBoost**(79.48%)** |
| Percentage of binary variables > 25% | Bank marketing dataset  LightGBM**(87.09%)**  CatBoost(78.77%) | E-Sun fraud dataset  LightGBM(96.54%)  CatBoost**(97.36%)** |

Table 4‑1 Summary of the Experimental Design

Despite our experimental design, note that all our datasets are categorical-heavy (percentage of categorical columns are all more than 50%). Obviously, if the data comes in tabular form, it often contains a lot of categorical values.

To make more interesting and useful conclusion, we should use the feature importance, which means the features the model relies when producing predictions.

In the following table, we list top 4 features for each dataset that the models use most (N stands for numerical, C stands for categorical, feature that exists in both model is marked as bold, and the number shows the feature importance):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Titanic | Cat in the dat | Bank marketing | E-Sun fraud |
| LightGBM | AUC: 85.1%  Fare(N)(481)  Age(N)(476)  Pclass(C)(91)  Sex(C)(78) | AUC: 75.09%  Nom\_5(C)(4067)  Nom\_6(C)(3609)  Nom\_7(C)(3104)  Nom\_8(C)(2955) | AUC: 87.09%  Duration(N)(634)  Balance(N)(587)  Age(N)(435)  Month(C)(347) | AUC: 96.54%  Bacno(C)(1255)  Cano(C)(548)  Mchno(C)(287)  Conam(N)(287) |
| CatBoost | AUC: 86.13%  Sex(C)(54.3)  Pclass(C)(15.0)  Age(N)(13.1)  SibSp(N)(7.3) | AUC: 79.48%  Ord\_5(C)(12.6)  Ord\_4(C)(8.9)  Ord\_2(C)(8.2)  Ord\_1(C)(7.9) | AUC: 78.77%  Duration(N)(33.7)  Contact(C)(13.3)  Month(C)(12.7)  Day(N)(8.7) | AUC: 97.36%  Stocn(C)(24.0)  Conam(N)(13.3)  Cano(C)(12.3)  Locdt(C)(12.1) |

Table 4‑2 Summary of Feature Importance

From our experiment, we can see that if the dataset is categorical heavy(percentage>70%), it is always the case that CatBoost performs better in terms of prediction accuracy, albeit the difference is not that big.

We also see that if the binary categorical variables percentage is high (>25%), LightGBM performs better. But based on feature importance analysis, we assume it was not the binary variables that lead to this outcome, but merely the data has a prominent feature that is numerical.

Additionally, we can see that CatBoost tends to choose categorical features, while LightGBM uses numerical ones, with feature importance levels compared

We can conclude that when the dataset has important numerical features, LightGBM performs better than CatBoost. But if the dataset has prominent categorical features, CatBoost tend to predict better.

# Conclusion

### Summary:

As all GBDT-based algorithms, LightGBM and CatBoost iteratively train weak learners, where the weights of the data points are updated according to the results of loss function of the previous learners.

But real-life applications are not only algorithms, but also the data that is crucial to produce meaningful results.

It is said that predictive analytics are only as good as your data, and if the data does not contain the essential information that can predict the target outcome, the performance is always poor regardless of the algorithms.

In this study, we compared both boosting methods in terms of AUC performance, and found the characteristics of data that influence the performance the most.

As in Section 4.5 showed, we discovered that CatBoost tends to use the categorical columns to produce the predictions, while LightGBM is more likely to utilize numerical columns. It is likely that the *prediction shifts* problem mentioned and solved by CatBoost algorithm helped a lot in prediction performance, in terms of common tabular datasets.

Based on the above observations, we conclude that:

1. In tabular datasets, CatBoost generally performs better, and our guess is that because most real tabular datasets have more crucial categorical columns, the algorithm that utilize categorical data the most will tend to do well.
2. On datasets that have important numerical columns, LightGBM have significantly better predictions. This is supported by the results on Section 4.3.

### Contribution:

1. Past literatures seldom discuss about the characteristic of data that would influence the algorithm performance. In this study we identified the situations and reasons that one will execute better than the other.
2. This study provided a fairer comparison between algorithms by using several datasets and enact preprocessing rules when the datasets have different characteristics.

### Limits:

1. No feature engineering

The study did not do feature engineering, which is an essential practice on tabular datasets to improve prediction performance. Since artificial feature engineering varies case by case, and it is hard to find a general rule to generate features, we believe we should not make new features in order to make a comparable comparison.

1. No hyperparameter tuning

Hyperparameters will influence how the model learn. There is also different hyperparameters to set for each model. In our case, LightGBM and CatBoost have similar hyperparameters, and although one algorithm may be better with hyperparameter optimized, we did not tune in this thesis.

1. Categorical Encoding methods

There are various categorical encoding methods, and in our study we used Label Encoding for our categorical columns. Note that models may have better performance with different encoding style, and with LightGBM we merely use the one that is recommended by the author, while CatBoost doesn’t require users to encode.

### Future studies:

1. Performance Metrics

We used AUC as our evaluation metric since it was common practice and recommended on [14]. There are still some interesting metrics that might be a better indicator for performance and may produce different results.

1. Data Volume

In our study we omitted the data volume, which is sometimes a huge part of algorithm selecting process. We believe data volume would matter since the algorithms use different ways to find the best split on features, and if the algorithm is greedier when optimizing, it is likely better on larger datasets.

1. New algorithms comparison

GBDT methods are still evolving even recently. A promising newcomer NGBoost[6] may be the new go-to algorithm for tabular datasets. It is an interesting topic that whether using natural gradient to optimize would produce better results in comparison with traditional gradient descent methods.

1. REFERENCE
2. K. Guolin, M. Qi, F. Thomas, W. Taifeng, C. Wei, M. Weidong, Y. Qiwei, L. Tie-Yan, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems* vol. 30, pp. 3149-3157, 2017.
3. A. Dorogush, V. Ershov, A. Gulin "CatBoost: gradient boosting with categorical features support," *NIPS*, pp.1-7, 2017.
4. J. Friedman. "Greedy function approximation: a gradient boosting machine." *Annals of Statistics*, 29(5): pp.1189-1232, 2001.
5. J. Friedman. "Stochastic gradient boosting." *Computational Statistics & Data Analysis*, 38(4): pp. 367-378, 2002.
6. Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016.
7. Tony Duan, Anand Avati, Daisy Yi Ding, Sanjay Basu, Andrew Y Ng, and Alejandro Schuler. "Ngboost: Natural gradient boosting for probabilistic prediction." *arXiv preprint* arXiv:1910.03225*.* 2019.
8. Stephen Tyree, Kilian Q Weinberger, Kunal Agrawal, and Jennifer Paykin. "Parallel boosted regression trees for web search ranking." In *Proceedings of the 20th international conference on World wide web*, pp. 387–396. ACM, 2011.
9. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in python." *Journal of Machine Learning Research*, 12(Oct): pp. 2825–2830, 2011.
10. Ridgeway. Greg., "Generalized boosted models: A guide to the gbm package." 2007. Retrieved from <https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf>
11. Daoud. E. A., "Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset," *International Journal of Computer and Information Engineering* 13(1), pp. 6-10, 2019.
12. Potdar, K., T. Pardawala and C. Pai, "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," Article in International Journal of Computer Applications, 2017.
13. Richard Ernest Bellman, *Dynamic Programming*,Princeton University Press, 1957.
14. Longadge, R. and S. Dongre, "Class Imbalance Problem in Data Mining Review." *arXiv preprint* arXiv:1305.1707, 2013.
15. Charles X. Ling., Huang. Jin, Zhang. Harry, "AUC: a statistically consistent and more discriminating measure than accuracy." Proceedings of the Eighteenth International Joint Conference of Artificial Intelligence (IJCAI) 2003.
16. LightGBM API, Microsoft, "Advanced Topics," April 2020, <https://lightgbm.readthedocs.io/en/latest/Advanced-Topics.html>
17. Frank E. Harrell Jr., Thomas Cason (1994). Titanic: Machine Learning from Disaster. Retrieved March 20, 2020 from <https://www.kaggle.com/c/titanic/data>.
18. Kaggle (2015 April). Titanic: Machine Learning from Disaster. Retrieved March 2020 from <https://www.kaggle.com/c/titanic/data>
19. Kaggle (2019 August). Categorical Feature Encoding Challenge. Retrieved March 2020 from <https://www.kaggle.com/c/cat-in-the-dat/data>
20. Moro, S., P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62: pp. 22-31, June 2014  
    Retrieved March 2020 from <https://www.kaggle.com/c/bank-marketing-uci/data>
21. E-Sun Bank(玉山銀行). Credit Card Fraud Detection Challenge, September 2019

Retrieved Sept 2019 from <https://tbrain.trendmicro.com.tw/Competitions/Details/10>