HOME CREDIT DEFAULT RISK

Cheng-Wei Huang & Arnovi Moinuddin *
H.Milton Stewart School of Industrial and Systems Engineering
Atlanta, GA
{wsncwhuang, amoinuddin6}@gatech.edu

Kavya Navaneetha Krishnan
Guggenheim School of Aerospace Engineering
Atlanta, GA
knk7@gatech.edu

Akpevwe Ojameruaye
Colleges of Computing, Business, and Engineering
Atlanta, GA
aojameruaye3@gatech.edu

ABSTRACT

Loans are important to lenders and borrowers. It is important to identify the risky behaviors of clients and make educated decision. This paper is built on an end-to-end machine learning case study for predicting the defaulting risk associated with a borrower. The paper highlights on the exploratory data analysis of the datasets, feature engineering and machine learning modeling techniques and assess the results of these models.

1. Introduction

There has been a significant increase in the banking industry recently in which the unbanked population who are not privy to financial literacy are discriminated against when applying to loans. This means there's an inequity in the field of loan granting that needs to be remedied. Our objective is to predict the likelihood of loan repayment to better distribute loans. We came across a dataset on Kaggle from HomeCredit whose mission is to broaden financial inclusion for unbanked populations.

1.1 Background

The dataset consisted of seven different sheets: current application, bureau balance, credit card balance, installments, cash balance and previous installments. The main objective is to identify the potential Defaulters based on the given data about the applicants. The probability of classification is essential because we want to be very sure when we classify someone as a Non-Defaulter, as the cost of making a mistake can be very high to the company. Our goal was to utilize important features from these sources to identify which would be key in the resultant machine learning model. Our methodology included cleansing the data, visualizing various elements to better understand it, feature selection, model fitting, and derive our final conclusions.

Since the dataset was created for a datathon, the testing data did not include the labeled response. Therefore, we decided to create a 70/30 split on the training set in order to test the accuracy of the model.

2. Methods

2.1 Data Processing

Data preparation is integral to deriving clear and accurate results from a machine learning model. There are specific properties we noted when taking a deep dive into the data. It was incredibly dense, spanning across ~450 columns in total and ~307,000 rows. There were inconsistencies in the data types within the columns as well meaning it was heterogeneous in nature. It was also time-varying and there were unbalances in some of the responses that required cleansing. The following flow was utilized to understand the scope of the data and determine which predictive model would be best.

Figure 1: Data Preparation Methodology

2.1.1 Data Exploration

We first hypothesized which features may be most important in determining the likelihood of loan repayment. Working with a few, we derived the following:

```
Percentage of people who will not repay the loan on time:  8.072881945686495 %
Percentage of people who will repay the loan on time:  91.92711805431351 %
```

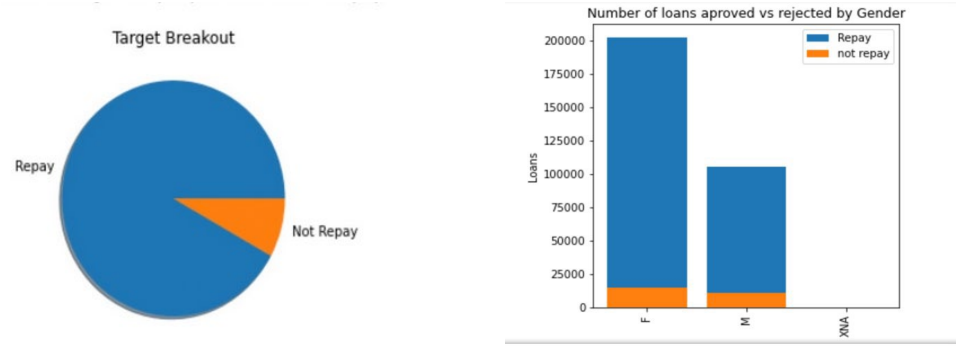


Figure 2: Number of loans approved vs rejected based on gender of the applicant

We began by looking at a high-level scale of metrics to gauge what our overall responses might look like. As you can see, the percentage of people who don't pay back their loan was very small, so it was important to identify the correct model and execute accurate feature selection to isolate the variables most impact likelihood of repayment. Here are some breakdowns of main factors we initially thought would strongly influence the model. Again, we see an even dispersion of the target across all groups of the factors observed. However, we believed that car ownership may be a strong indicator that an ability to pay off a car loan might signal an ability to pay off other types of loan.

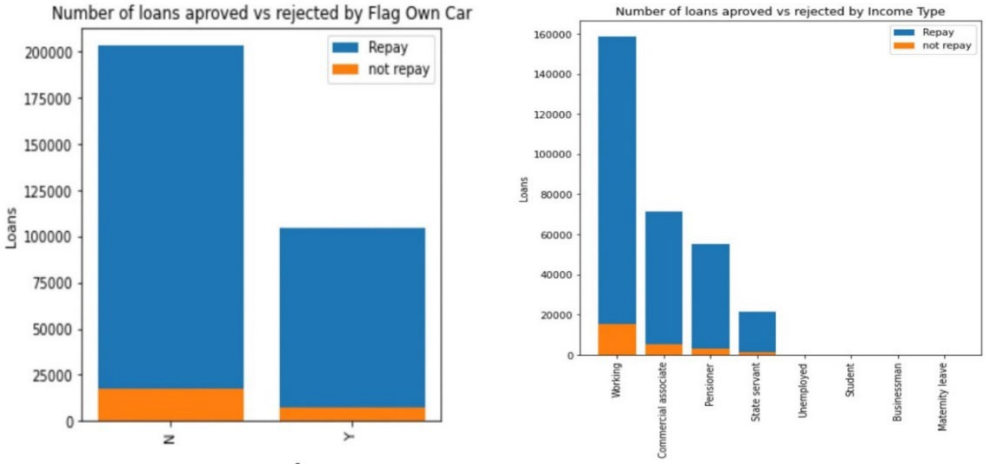


Figure 3: Data information on loans against possession of Cars and occupation status of the applicant

In order to aggregate the data appropriately, we used the following schema to determine the primary and foreign keys necessary for database integration. This allowed us to access all the attributes associated with each customer simultaneously and manipulate the data accordingly.

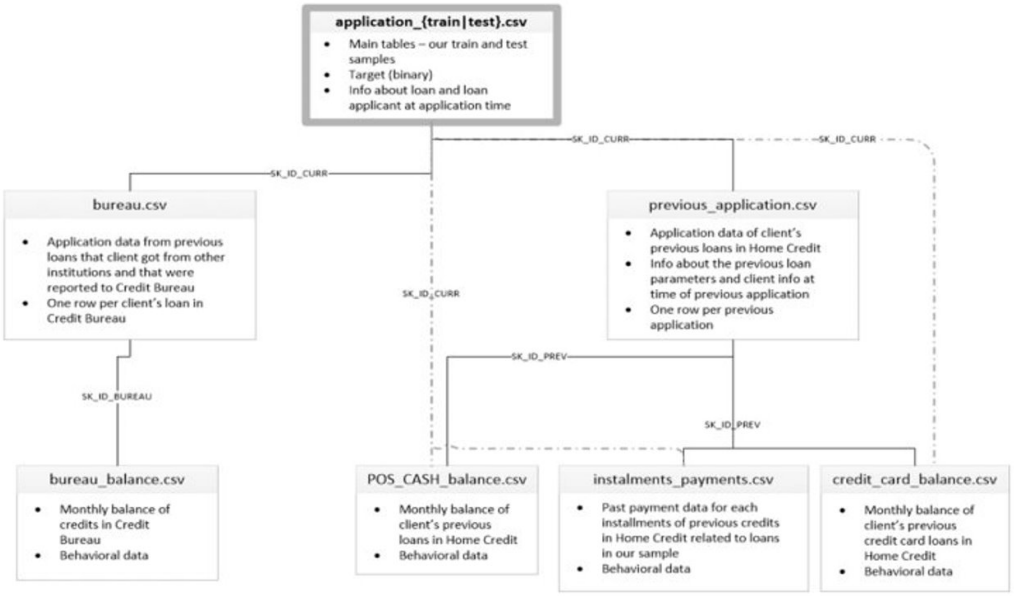


Figure 4: Information chart on the datasets available on Default Risks from Kaggle

The first order of business to find which attributes are positively and negatively correlated with our response. We found the following results.

Top 10 Negative Correlations:	
EXT_SOURCE_3	-0.178919
EXT_SOURCE_2	-0.160472
EXT_SOURCE_1	-0.155317
NAME_EDUCATION_TYPE_Higher education	-0.056593
CODE_GENDER_F	-0.054704
NAME_INCOME_TYPE_Pensioner	-0.046209
ORGANIZATION_TYPE_XNA	-0.045987
DAYS_EMPLOYED	-0.044932
FLOORSMAX_AVG	-0.044003
FLOORSMAX_MEDI	-0.043768

Top 10 Positive Correlations:	
REG_CITY_NOT_WORK_CITY	0.050994
DAYS_ID_PUBLISH	0.051457
CODE_GENDER_M	0.054713
DAYS_LAST_PHONE_CHANGE	0.055218
NAME_INCOME_TYPE_Working	0.057481
DAYS_CREDIT	0.058315
REGION_RATING_CLIENT	0.058899
NAME_CONTRACT_STATUS_Refused	0.060789
REGION_RATING_CLIENT_W_CITY	0.060893
DAYS_BIRTH	0.078239

Figure 6: Correlation results from the Dataset

As we can see, the influence of each factor seems incredibly small, but it's crucial to note the number of factors under consideration during this analysis. Therefore, relatively, this insight is useful. We've included part of the myriad of breakdown we've built, including the relationships between correlation across age groups, and the variable known as 'EXT_SOURCE_3'. Upon investigation, we discovered that a large part of the data sourcing comes from confidential financial information that has been tokenized for our use. They've been exported into these columns accordingly. This information is insightful particularly because it helps us better predict which attributes, we may observe as a result of our feature selection.

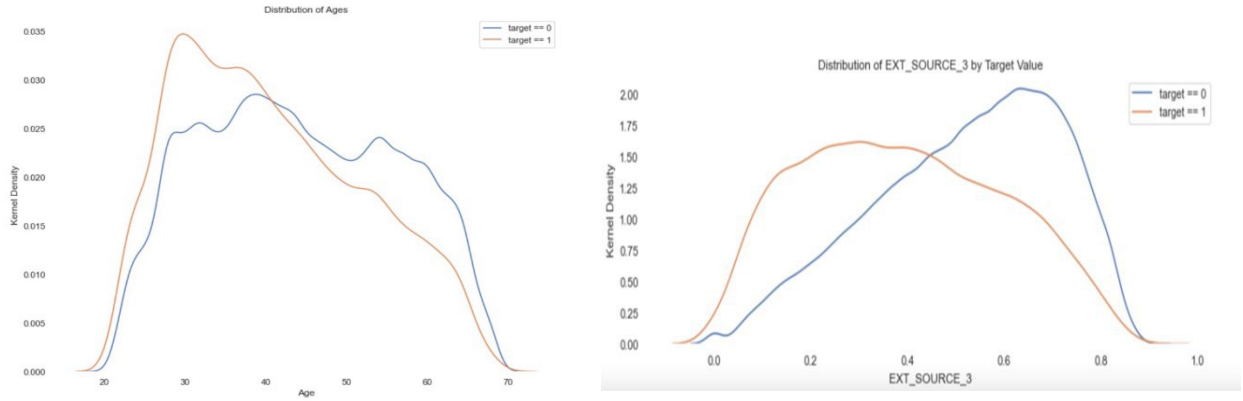


Figure 7: Attributes selection from feature engineering

We then wanted to explore whether there were significant correlations between the attributes themselves. We looked through our table with the highest correlations to the response, and then built out heat maps based on their respective values. Since we previously analyzed impact on the target value, we wanted to isolate variables' relationships to one another. This is particularly insightful to see if there was an aliasing of any kind of variable to one another. If you look at the figure associated with credit card balance, we find that there are a few extraneous highly correlated variables, but their exclusion in our final model did not impact our results, therefore we moved forward with the appropriate feature selection procedures.

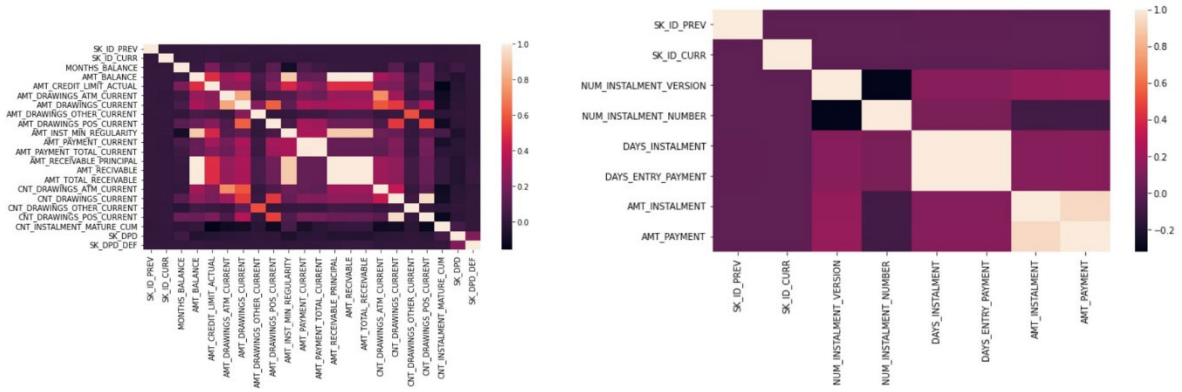


Figure 8: Correlation Heat Map: Credit Card Balance, Payment Installments

2.1.2 Data Cleansing and Feature Selection

We came across a significant number of columns that had missing values at a proportion that would be misrepresentative to keep in our model. For the columns that had over 60% missing data, we dropped the columns entirely. For those with less than 60%, but still a significant amount of missing data, we used the median of the columns in place of the null values. Here is an example of the list of columns that yielded these results.

	% Missing Values
COMMONAREA_MODE	69.872297
COMMONAREA_MEDI	69.872297
COMMONAREA_AVG	69.872297
NONLIVINGAPARTMENTS_MODE	69.432963
NONLIVINGAPARTMENTS_AVG	69.432963
NONLIVINGAPARTMENTS_MEDI	69.432963
FONDKAPREMONT_MODE	68.386172
LIVINGAPARTMENTS_MODE	68.354953
LIVINGAPARTMENTS_AVG	68.354953
LIVINGAPARTMENTS_MEDI	68.354953

Figure 9: Missing Value from the dataset

In order to prepare our data for feature selection, we needed to identify the datatypes that were utilized and the breakdown of categorical vs quantitative variables. Our analysis displayed the following results.

Column Count by Data Type

Data Type	Column Count
Float64	107
Int64	42
Object	36

Unique Value Count by Categorical Column

Column Name	Unique Value Count
Organization_Type	58
Name_Goods_Category	27
Occupation_Type	18
Product_Combination	17
Credit_Type	14

Figure 10: Categorical vs quantitative data split

To make use of the insightful categorical values, we utilized One-Hot Encoding to convert the variables into binary. An example of the conversions as a result of this method are showcased below.

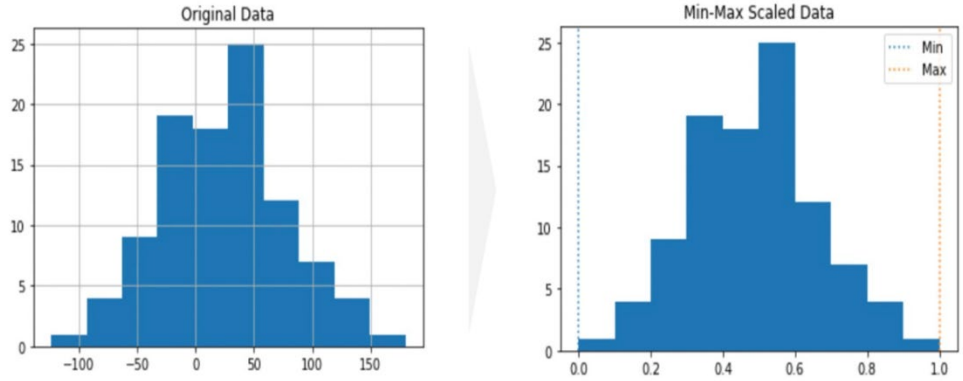
id	color
1	red
2	blue
3	green
4	blue

One Hot Encoding

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Figure 11: Illustration of One-Hot Encoding Example

To gauge the accuracy of the conversions and ensure the distribution of values was not skewed in any way, we used minimax standardization. This verifies that all the previously categorical data was on the same scale and that certain variables do not have more of an effect than others. Our results behaved as expected as can be seen below.




```
scaler = MinMaxScaler(feature_range = (0,1)).fit(train_onehot)
train_onehot = scaler.transform(train_onehot)
```

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Figure 12: Illustration of MinMax Standardization

Lastly, we used Random Forest to conduct our final form of feature selection. We finalized on this algorithm after much research for feature selection methods as Random Forest an embedded method meaning the combine the qualities of filter and wrapper method. As a result, they are highly accurate, generalize better, and are interpretable. Due to the nature of the algorithm, they are less prone to overfitting and can more accurately determine the importance of features. After running our algorithm, we were able to finalize on

~70 noteworthy columns as opposed to the previous ~400. Their respective proportion of importance is also displayed.

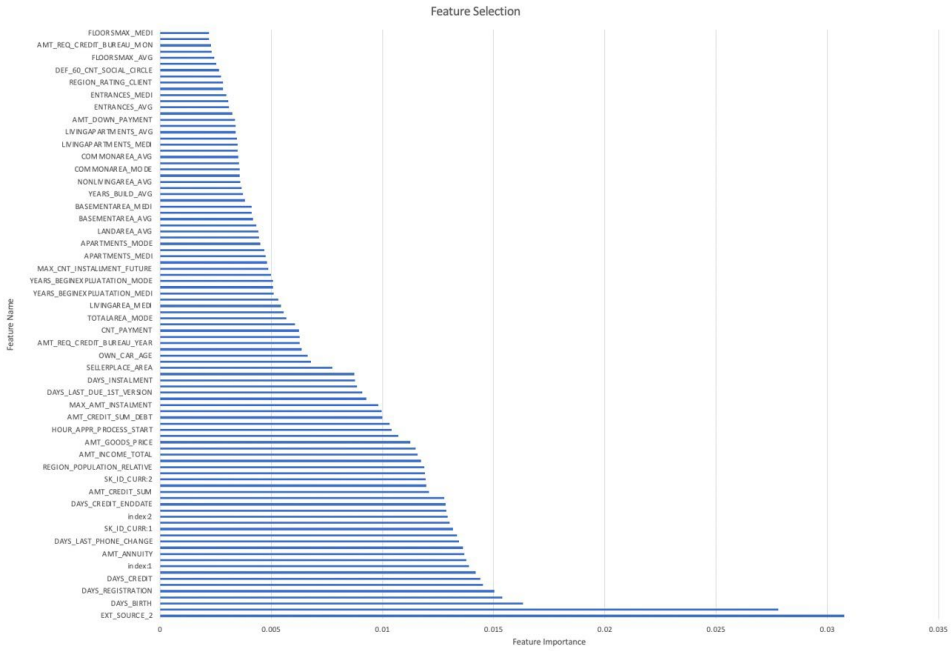


Figure 13: Feature Selection visualization from Dataset

2.2 Logistic Regression

Logistic regression was used to assess the initial classification task of the data and feature set. We noticed that this algorithm takes a longer time with the large dataset present and had high memory usage. The accuracy produced was 0.919266 with a low AUC score of 0.5026. The possible reasons for low AUC may be due to the assumption that there is a linear relationship between the features and the target variable. If this assumption is not met, the model may not be able to accurately predict the target variable, leading to a low AUC.

2.3 LightGBM

The data exploration process shows that the home credit default risk is large and complicated. Hence, we decided to use the LightGBM model to train our data. LightGBM is a gradient-boosting framework that uses tree-based learning algorithms, which can train data at a faster speed with lower memory usage and is capable of handling large-scale data.

To have a first glance at the LightGBM model, we followed the rule from LightGBM documentation and assign the parameter arbitrarily. The documentation suggests assigning larger max_bin and num_leaves parameters and a smaller training rate for gaining better accuracy. We also assign the feature selection parameter to 0.8 to avoid overfitting. The following image is the first model we made and got a rather good accuracy about 0.903527. Then we investigated the relationship between each parameter and the accuracy. In order to optimize the hyperparameter, we extract the range where the highest accuracy occurs.

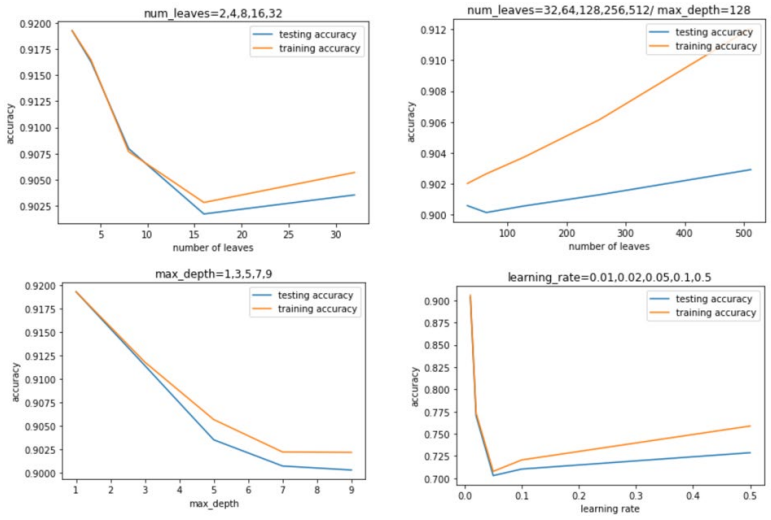


Figure 14: Accuracy plots for LightGBM

From the graph above, we can observe that the three parameters have similar patterns. We got the highest training and testing accuracy when the parameter value is small. Although both accuracies will increase again as the parameter becomes larger, the gap between training and testing accuracies raise as well, which means overfitting happens. Accordingly, the best range of the number of leaves, maximum depth, and the learning rate is 0 to 5, 0 to 5, and 0 to 0.1 respectively.

After that, we selected some values in the range to find the best combination of parameters that can build the model with the highest accuracy. We imported the GridSearchCV package to help us fulfill this work, and we can see the best combination is to set num_leaves=2, max_depth=1, and learning_rate=0.01. We then utilized these parameters to build the LightGBM again and got a higher accuracy, 0.919266.

3. Results and Discussions

Since the data available to us is an Imbalanced Dataset, we cannot simply use Accuracy as a metric for evaluating the performance of the model. There are some metrics that work well with imbalanced datasets, of which we will use the below-mentioned metrics. The AUC Score insensitive to class imbalance. It works by ranking the probabilities of prediction of the positive class label and calculating the Area under the ROC Curve which is plotted between True Positive Rates and False Positive Rates for each threshold value. This is because we need to minimize the False Negatives, i.e., the people who were predicted as non-Defaulters by the model but were Defaulters. We do not want to miss out on any Defaulter as being classified as non-Defaulter because the cost of making errors could be very high. LightGBM produced a higher AUC of 0.7529 against the 0.5026 of logistic regression. This shows that the boosting algorithm had better classification than binary classification, logistic regression.

4. Conclusions

The constraints from the datasets noticed were mainly on the Interpretability is partially important for classifying someone as a Defaulter or not. After identifying the business objectives and constraints, data imbalance insensitive machine learning algorithm was chosen. Feature Engineering proved to be more useful than model tuning and stacking. Using these we formulate the following conclusions:

- Some categories are discriminatory between the defaulters and non-defaulters.
- Dataset is imbalanced, some correlated features weren't affecting the performance of the model.

- Defaulters usually tend to have some behavior which deviate from the normal, and thus, we cannot remove outliers or far-off points, as they may suggest some important Defaulting tendency.
- LightGBM boosted the CV from feature selection, Usage of simple forward feature selection technique with Ridge regression and boosting algorithm like XGBoost, may have a further boost in CV and AUC.

REFERENCES

[1] Home Credit Default Risk, Kaggle Competetions, 2018, URL: https://www.kaggle.com/competitions/home-credit-default-risk

[2] First Place Solution, Home Credit Default Risk, 2018, URL: https://www.kaggle.com/c/home-credit-default-risk/discussion/64821

A. APPENDIX

1. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
clf_lr = LogisticRegression(random_state=0).fit(X_train, y_train)
prediction = clf_lr.predict(X_test)
score = clf_lr.score(X_test, y_test)
```

2. LightGBM

```
clf=lightgbm.LGBMClassifier(**parameters)
clf.fit(X_train,y_train)
```

```
[LightGBM] [Warning] boosting is set=goss, boosting_type=gbdt will be ignored. Current value: boosting=goss
[LightGBM] [Warning] feature_fraction is set=0.8, colsample_bytree=1.0 will be ignored. Current value: feature_fraction=0.8
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
```

```
LGBMClassifier
LGBMClassifier(bagging_freq=5, boosting='goss', feature_fraction=0.8,
               is_unbalance='true', learning_rate=0.01, max_depth=5,
               metric='auc', num_leaves=32, objective='binary')
```

```
#predict the value of the target
y_pred=clf.predict(X_test)
```

```
print("Accuracy:%f"%(1-(y_test.to_numpy()!=y_pred).sum()/y_test.to_numpy().shape[0]))
```

```
Accuracy:0.903527
```

```
tune_parameters={
    'objective':'binary',
    'metric':'auc',
    'is_unbalance':'true',
    'boosting':'goss',
    'num_leaves':grid_model.best_estimator_.get_params()['num_leaves'],
    'max_depth':grid_model.best_estimator_.get_params()['max_depth'],
    'feature_fraction':0.8,
    'bagging_freq':5,
    'learning_rate':grid_model.best_estimator_.get_params()['learning_rate']
}
```

```
tune_clf=lightgbm.LGBMClassifier(**tune_parameters)
tune_clf.fit(X_train,y_train)
```

```
LGBMClassifier
LGBMClassifier(bagging_freq=5, boosting='goss', feature_fraction=0.8,
               is_unbalance='true', learning_rate=0.01, max_depth=1,
               metric='auc', num_leaves=2, objective='binary')
```

```
#predict the value of the target
tune_y_pred=tune_clf.predict(X_test)
```

```
print("Accuracy:%f"%(1-(y_test.to_numpy()!=tune_y_pred).sum()/y_test.to_numpy().shape[0]))
```

```
Accuracy:0.919266
```