HOME CREDIT DEFAULT RISK

Minh-Chiên DUONG, Thi-Thanh-Binh NGUYEN, Taiyun YU

M1 2IS, Université Toulouse 1 Capitole

Advisor: Dennis WILSON

June 16, 2019

1 Introduction of Home Credit

Founded in 1997, Home Credit Group is an international consumer finance provider with operations in 10 countries. They focus on responsible lending primarily to people with little or no credit history. Their services are simple, easy and fast. Home Credit aims to:

- Provide innovative retail financial services with a focus on mass-retail lending
- Help clients realise their dreams and ambitions in a financially responsible way
- Offer long-term, stable and interesting employment to our employees
- Encourage economic development through supporting domestic consumption, thereby improving living standards

The Point-of-sale loan is the important product in Home Credit's business model because for most of their customers it is the point of entry into Home Credit. Having acquired a customer through POS loan for durable goods, they start the long-term relationship and do repeat business with them. Other products of Home Credit include credit cards, revolving loans, car loans, current accounts, insurance...

1.1 Home Credit's Competition on Kaggle

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data–including telco and transactional information–to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

1.2 Data description

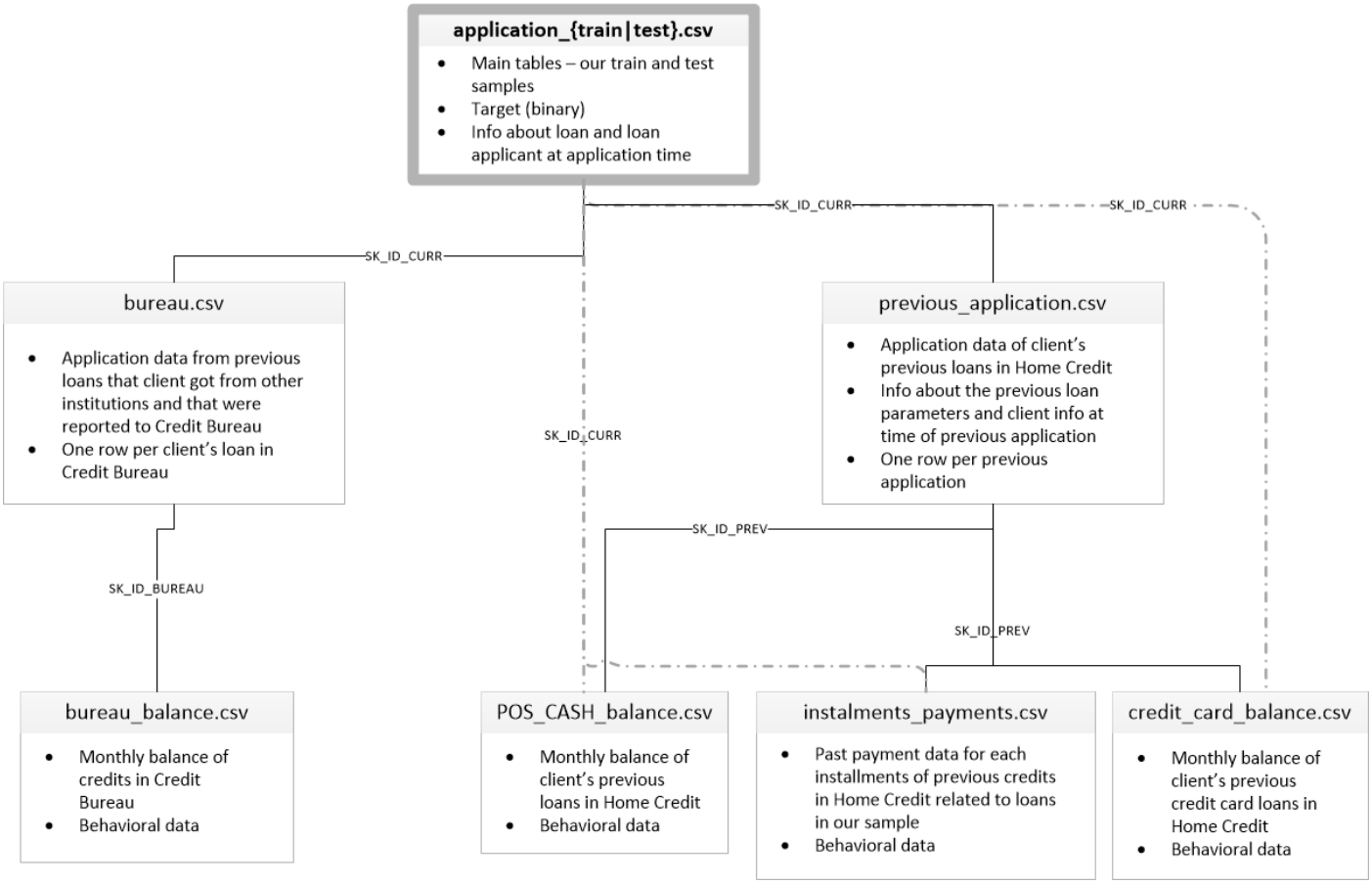

Figure 1: Data Relationship

The data of this competition includes 7 files. Among them, some are very large such as buerau_alance 27.3 millions rows, installments_payment 13.6 millions rows, POS_CASH 10 millions rows. Similarly, application_train has a huge number of features up to 121.

- application_train/application_test: the main training and testing data with information about each loan application at Home Credit. Every loan has its own row and is identified by the feature SK_ID_CURR. The training application data comes with the TARGET indicating 0: the loan was repaid or 1: the loan was not repaid.
- bureau: data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.
- bureau_balance: monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.
- previous_application: previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.
- POS_CASH_BALANCE: monthly data about previous point of sale or cash loans clients have had with Home Credit. Each row is one month of a previous point of sale or cash loan, and a single previous loan can have many rows.

- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.
- installments_payment: payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment.

2 Visualization

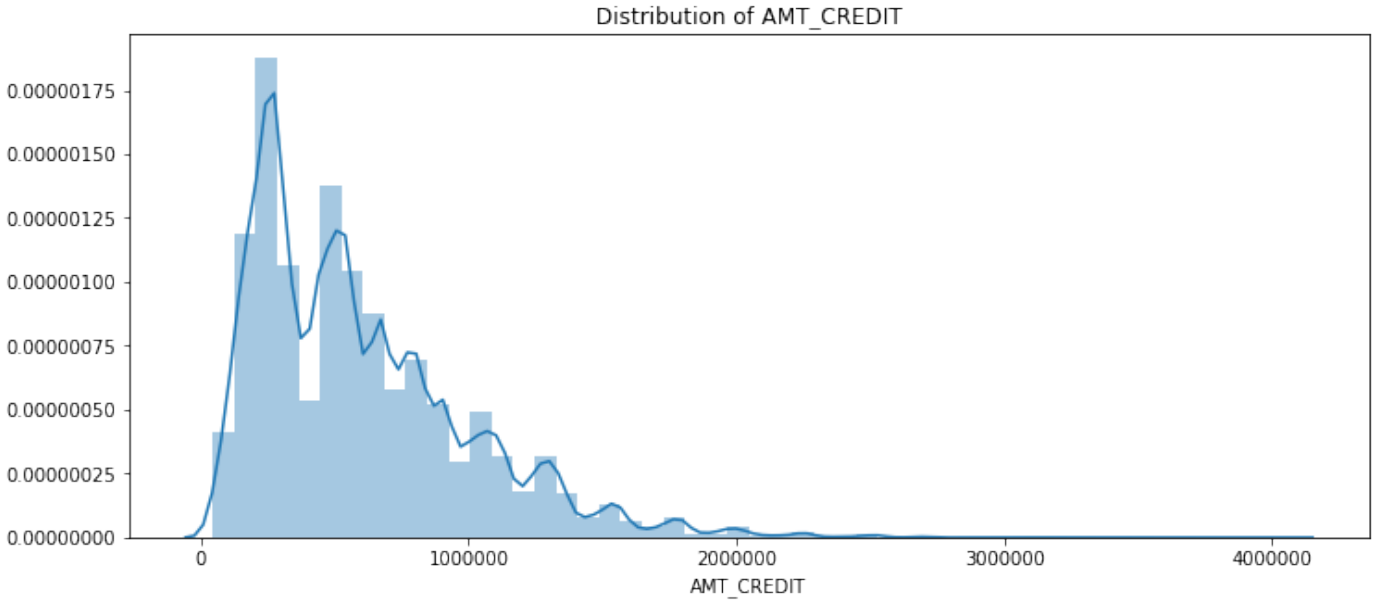

Figure 2: Distribution of AMT_CREDIT

aaaaaaaaaaaaaaa

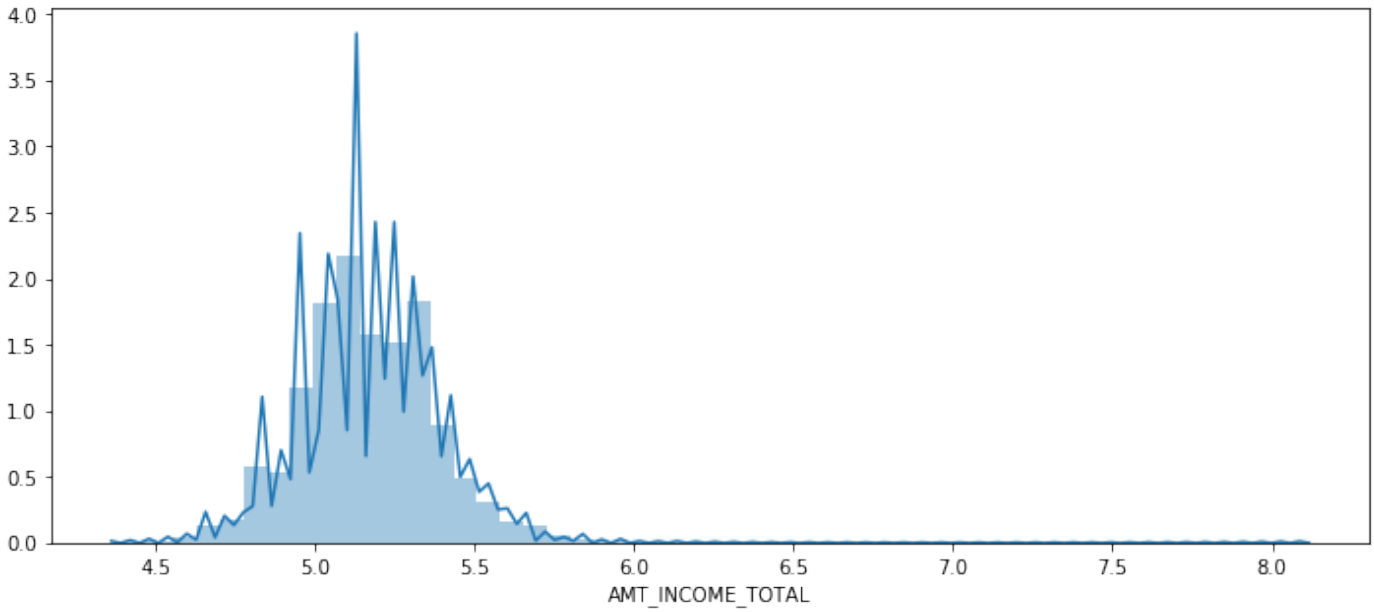

Figure 3: Distribution of AMT_INCOME_TOTAL

eeeeeee

bbbbbbbbbbbbbb

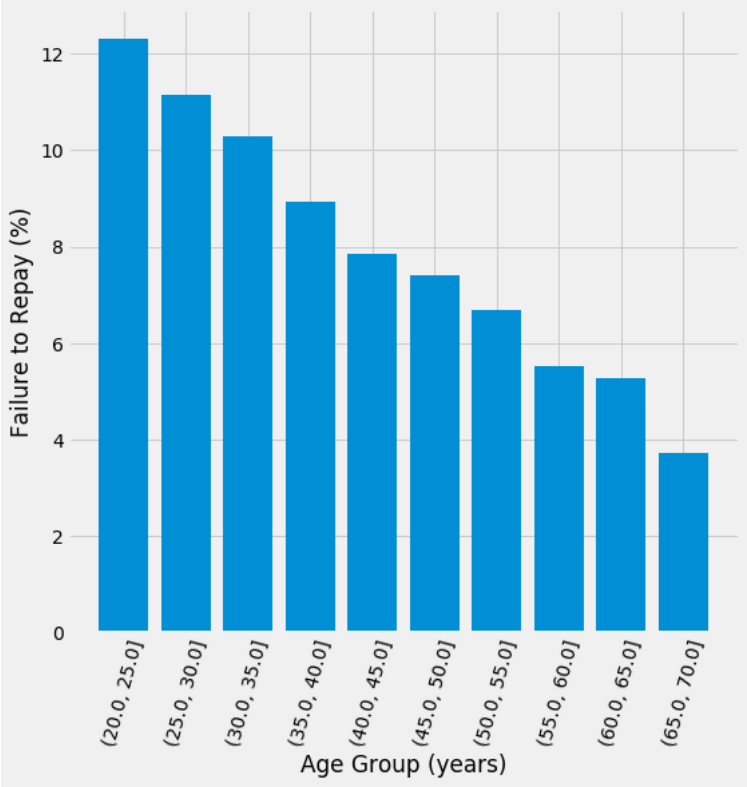

Figure 4: Failure to Repay by Age Group

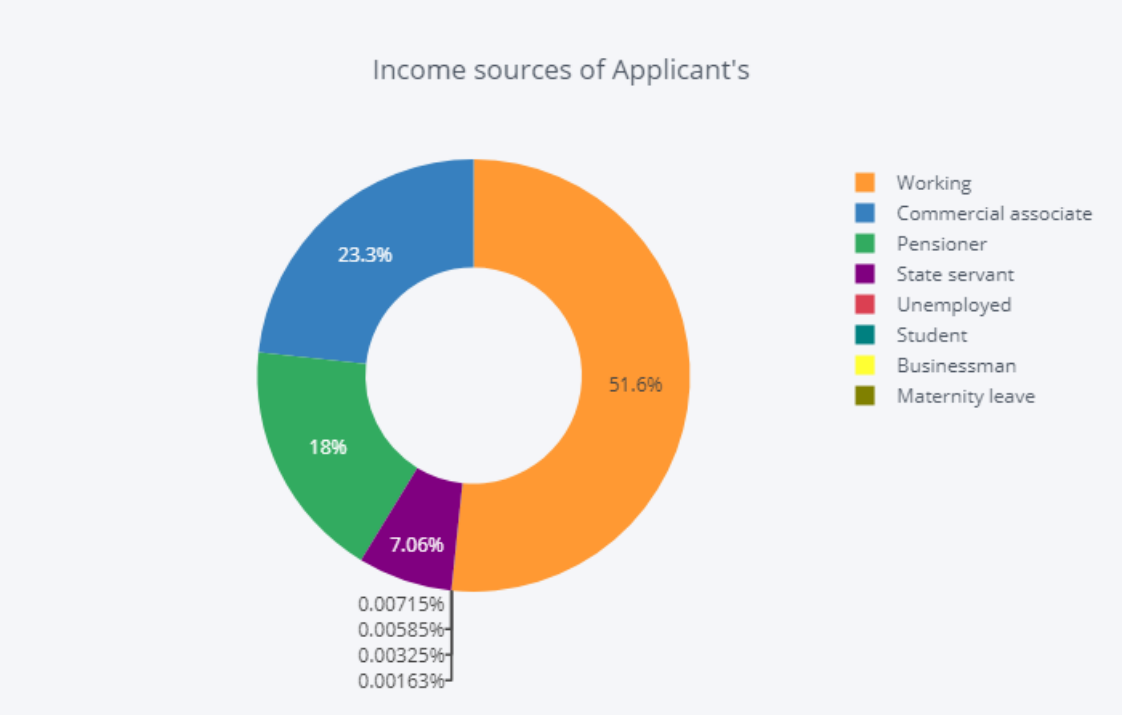

Figure 5: Income sources of Applicant's

ccccccccc

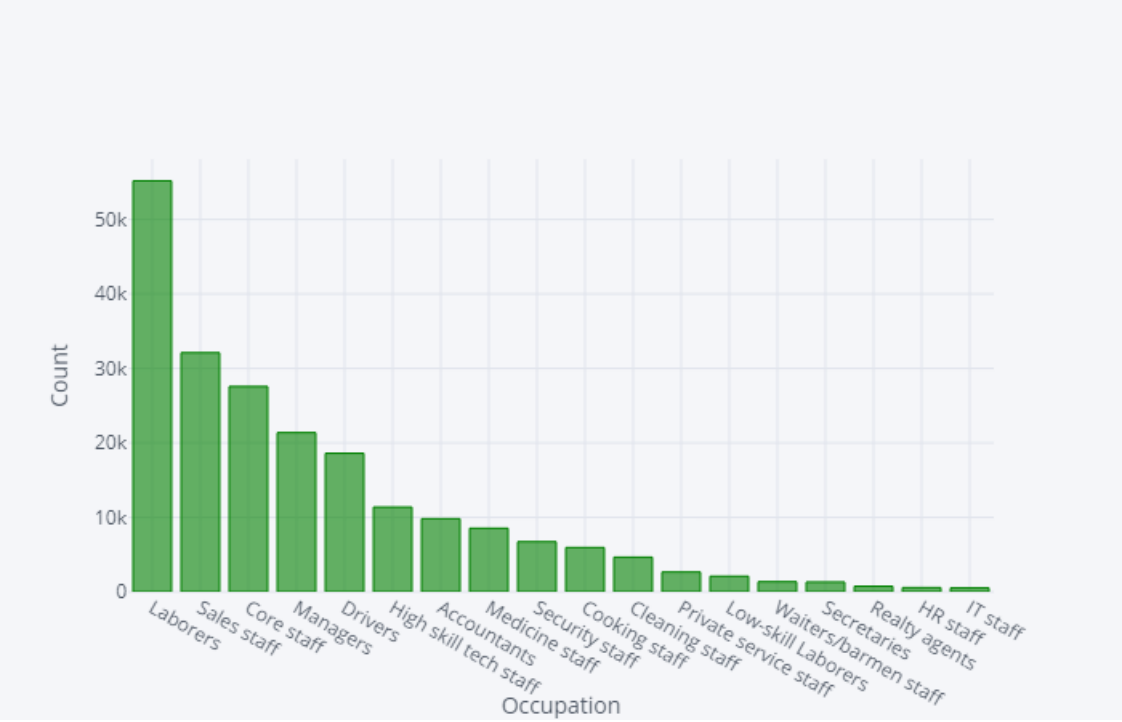

Figure 6: Occupation of Applicant's who applied for loan

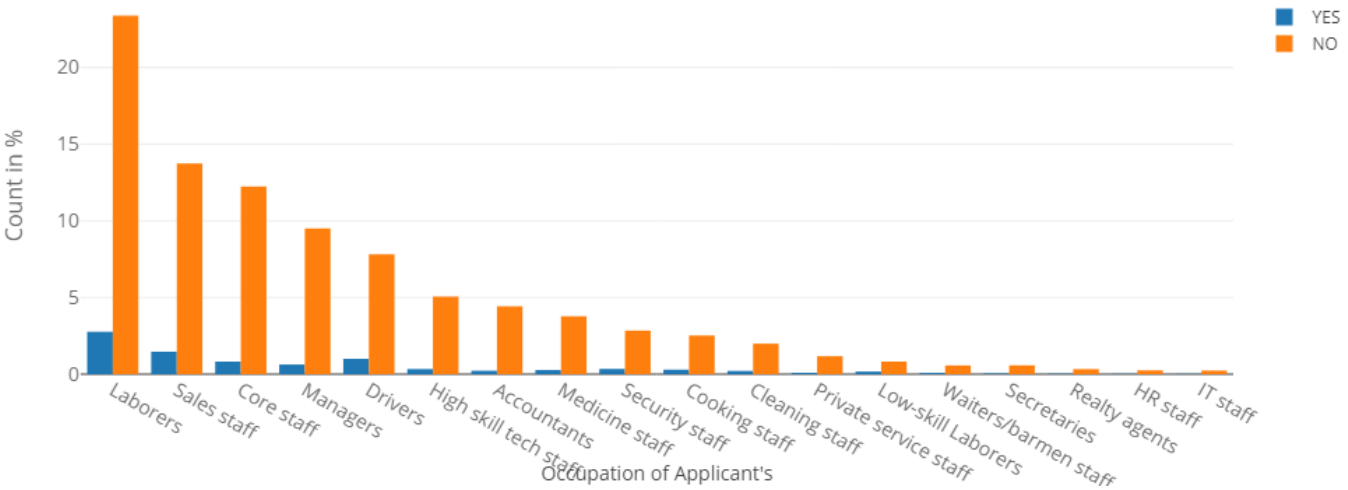

Figure 7: Occupation of Applicant's in terms of loan is repayed or not in %

gggggggggg

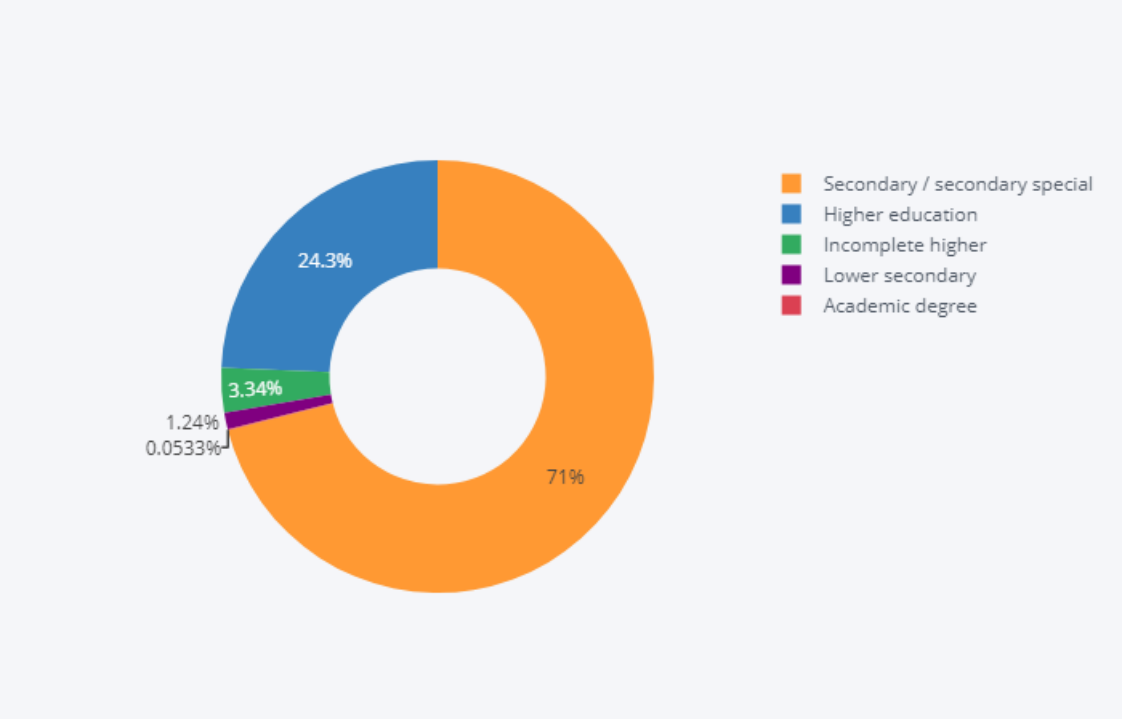

Figure 8: Occupation of Applicant's who applied for loan

ddddddddd

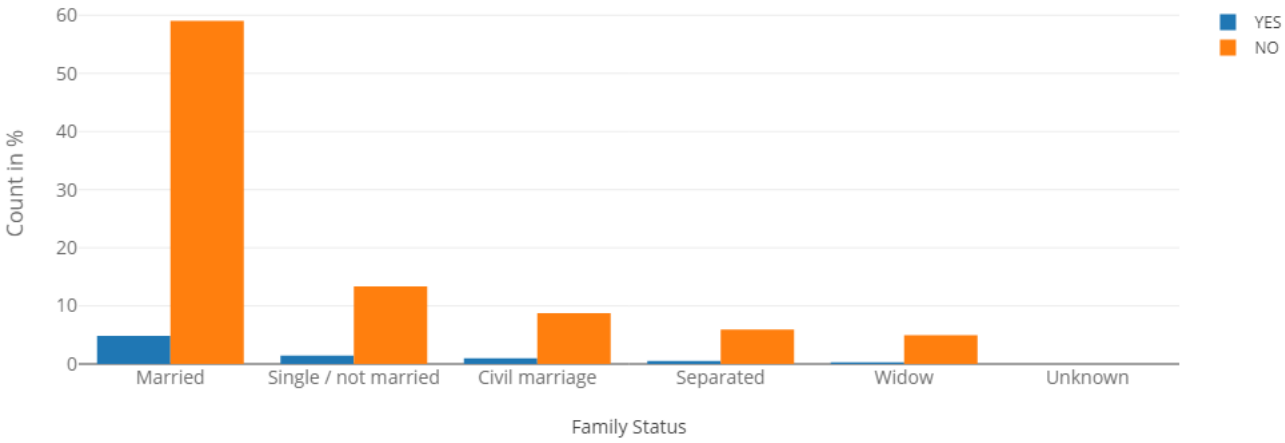

Figure 9: Family Status of Applicant's in terms of loan is repayed or not in %

3 Available kernels

3.1 First kernel: A Gentle Introduction

This kernel is published by Will Koehrsen. He gives a very basic introduction how to do this project, and we show briefly good points what we learn from it.

3.1.1 Strength

We first made sure to understand the data, our task, and the metric by which our submissions will be judged. Then, we performed a fairly simple EDA to try and identify relationships, trends, or anomalies that may help our modeling. Along the way, we performed necessary preprocessing steps such as encoding categorical variables, imputing missing values, and scaling features to a range. Then, we constructed new features out of the existing data to see if doing so could help our model.

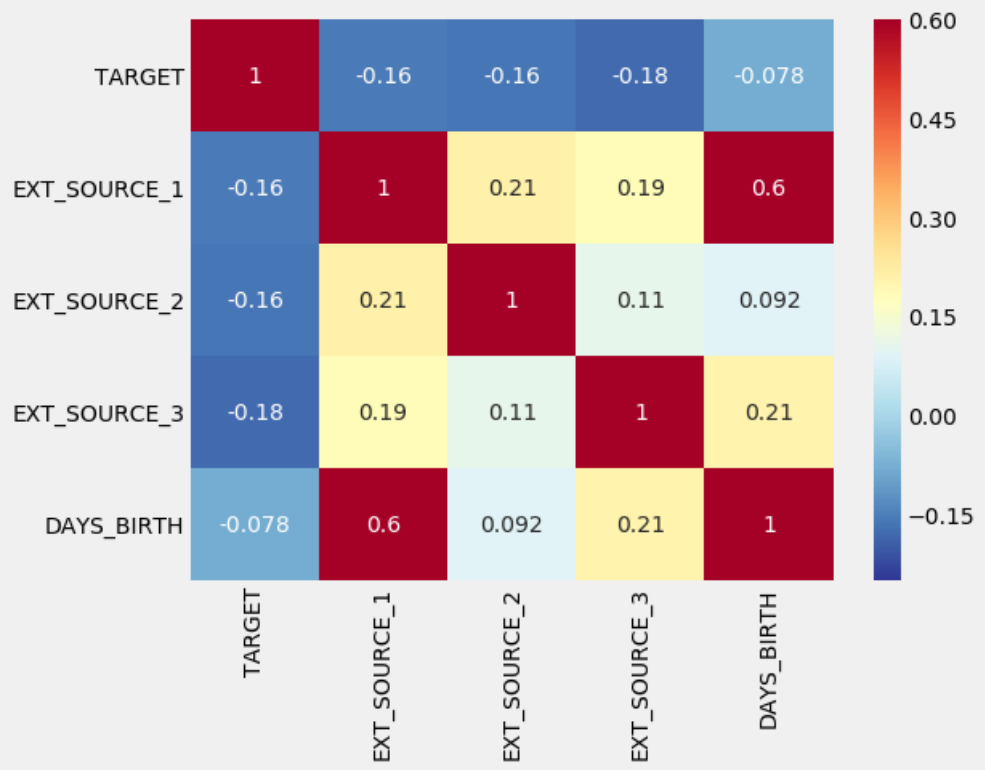

Figure 10: Top correlated features with the target

Once the data exploration, data preparation, and feature engineering was complete, we implemented a baseline model upon which we hope to improve. Then we built a second slightly more complicated model to beat our first score. We also carried out an experiment to determine the effect of adding the engineering variables.

We followed the general outline of a machine learning project:

- Understand the problem and the data
- Data cleaning and formatting
- Exploratory Data Analysis
- Baseline model
- Improved model
- Model interpretation

3.1.2 Weakness

The score of this kernel is very low. Firstly, the author just uses only application files (train and test) to build his model. In other words, he ignores the rest data. Secondly, he uses the random forest to

build his model. Random forest is not good enough for this kind of task. That is why this model scores 0.679 when submitted.

- Use only application files (ignores the rest data)
- Implement Random forest.

3.2 Second kernel: LightGBM with selected features

3.2.1 Strength

The seconde kernel gives a higher result, which is 0.79063, because of some improvements:

- Preprocess all data files, and then merge them together
- Create new features computed from original features of each file
- Drop redundant features
- Implement LightGBM algorithm

This kernel takes into account all data for its model. Secondly, we create many new features from original features, then use them to predict the target. For example, in the file Installments payments, we can find two new features which represent Percentage and Difference paid in each installment:

ins['PAYMENT_PERC'] = ins['AMT_PAYMENT'] / ins['AMT_INSTALMENT']
ins['PAYMENT_DIFF'] = ins['AMT_INSTALMENT'] - ins['AMT_PAYMENT']

3.2.2 Weakness

Although the second kernel gives a better result then the firs kernel's, it still has some things needed to be improved.

- New features are created from each file, not combine features from different files.
- Parameters of LightGBM could be updated with better values.

4 Implementing

4.1 Data preprocessing

In this part, we describe how we preprocessing data. We have 7 different input files, the intuition of handling data is quite similar. We preprocess data in several steps. After those steps, data are eligible for training models.

- Preprocess application files

- Preprocess bureau and bureau_balance, merging them together
- Preprocess previous_application
- Preprocess POS_CASH_balance
- Preprocess instalments_payments
- Preprocess credit_card_balance
- Merge all data frames together
- Adding new features

4.1.1 Utility functions

One-hot encoder

We use get_dummies fucntion of panda to convert categorical variable into dummy/indicator variables. Whenever we encode a table, we use a variable *categorical_columns* which contains all the columns created by the One-hot encoder.

For example, the original bureau balance has only two features: MONTHS_BALANCE and STATUS.

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C

Figure 11: Original Bureau Balance data

We use One-hot encoder to handle the feature STATUS, and group by SK_ID_BUREAU.

	SK_ID_BUREAU	STATUS_0	STATUS_1	STATUS_2	STATUS_3	STATUS_4	STATUS_5	STATUS_C	STATUS_X
0	5001709	0.000000	0.0	0.0	0.0	0.0	0.0	0.886598	0.113402
1	5001710	0.060241	0.0	0.0	0.0	0.0	0.0	0.578313	0.361446
2	5001711	0.750000	0.0	0.0	0.0	0.0	0.0	0.000000	0.250000
3	5001712	0.526316	0.0	0.0	0.0	0.0	0.0	0.473684	0.000000
4	5001713	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	1.000000

Figure 12: Bureau Balance after implementing One-hot encoder

Creating new aggregation features

In each data file, we usually transform interest features into new aggregate ones, which are useful to predict the target. The simplest one of them looks like this:

agg = 'MONTHS_BALANCE': ['min', 'max', 'mean', 'size']

Come back to the example in Bureau Balance, we compute Min, Max, Size and Mean duration of payments in months, and also group by SK_ID_BUREAU. Now we see clearly that the new features created by One-hot encoder and Aggregate function represent much more information of the payment ability of borrowers.

	SK_ID_BUREAU	MONTHS_BALANCE_MIN	MONTHS_BALANCE_MAX	MONTHS_BALANCE_MEAN	MONTHS_BALANCE_SIZE
0	5001709	-96	0	-48.0	97
1	5001710	-82	0	-41.0	83
2	5001711	-3	0	-1.5	4
3	5001712	-18	0	-9.0	19
4	5001713	-21	0	-10.5	22

Figure 13: Bureau Balance after handling MONTHS_BALANCE

Creating new numeric features

For numerical features, we tend to combine them to make new ones. The new features usually have financial meanings and larger correlation with the target.

Continuing with the Bureau Balance, we create some new features as following:

- CREDIT_DURATION = DAYS_CREDIT_ENDDATE - DAYS_CREDIT
- DEBT_PERCENTAGE = AMT_CREDIT_SUM / AMT_CREDIT_SUM_DEBT

In bureau, DAYS_CREDIT represents how many days before current application did client apply for Credit Bureau credit. And DAYS_CREDIT_ENDDATE shows the remaining duration of CB credit at the time of application in Home Credit. We can get the new feature [CREDIT_DURATION] which reveals the accumulated payment duration of previous loans by subtracting DAYS_CREDIT from DAYS_CREDIT_ENDDATE.

AMT_CREDIT_SUM is current credit amount for the Credit Bureau credit. AMT_CREDIT_SUM_DEBT is current debt on Credit Bureau credit. We create the new features: [DEBT_PERCENTAGE], which represents current debts percentages.

Label encoder

Besides One-hot encoder, we use Label encoder to encode categorical values as integers (0,1,2,3...) with pandas.factorize.

Group and merge

In each dataset, one borrower can have many loan records in the past. For example, one value of SK_ID_CURR in Bureau corresponds to multiple values of SK_ID_BUREAU. Therefore, we use the group function to solve this kind of data.

After preprocessing each file, we merge all them together. In that case, we call for the group-and-merge function. When we merge Bureau Balance to Bureau, the key is SK_ID_BUREAU. For other files, the key is SK_ID_CURR.

Aggregate functions

As part of preprocessing data, we write some aggregate functions: do_mean, do_median and do_std, which are used to create new features.

Handling missing data

We see that there are a huge amount of missing data in each file. We can fill them by average values according to each feature.

Table 1: Top 5 missing features

	Total	**Percent**
COMMONAREA_MEDI	214865	69.9
COMMONAREA_AVG	214865	69.9
COMMONAREA_MODE	214865	69.9
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_MODE	213514	69.4

Handling outliers

If the number of outliers is small, for exam it is, we could remove them. otherwise, we treat those outliers as a new type of category.

4.1.2 Important features

For each data, we do many steps to preprocess it. Because of the limitation of this report, we just introduce some useful features in our opinion to predict the target. In our programme, the number of new features created is much larger than the ones introduced in this part.

Application files

In Application files, we remove 4 people with XNA code gender, CODE_GENDER. And we get the value of the total income, AMT_INCOME_TOTAL, less than 20 million $. By doing this, we consider the clients whose incomes may be up to 117 millions $ are outliers.

Next, we see that there are 20 features, named from FLAG_DOCUMENT_2 to FLAG_DOCUMENT_21, those features represent the documents provided by clients. We aggregate those features to compute their sum, kurtosis. Furthermore, we combine some features to create many new ones.

$$\text{EXT\_SOURCES\_PROD} = \text{EXT\_SOURCE\_1} * \text{EXT\_SOURCE\_2} * \text{EXT\_SOURCE\_3}$$

EXT sources are important

Since we know that, DAYS_BIRTH has a strong correlation with the target, we label them into 6 groups: 1 (<27 years old), 2 (<40 years old), 3 (<50 years old), 4 (<65 years old), 5 (<99 years old), and 6 for the rest.

Last but not least, we drop redundant features. For those features which we use to create new ones, we should drop them to improve the training model. For instance, as mentioned above, we have 20 features from FLAG_DOCUMENT_2 to FLAG_DOCUMENT_21, then we drop all of them but only FLAG_DOCUMENT_3.

Bureau and Bureau Balance

In Bureau, we particularly interest in CREDIT_ACTIVE feature, which reveals whether a credit is closed or still active. Another important feature we take into account is CREDIT_TYPE, which contains Consumer credit, Credit card, Mortgage, Car loan, Microloan.

Previous file

In Previous data, NAME_CONTRACT_STATUS is a good feature. Through it, we could know an application of loan is approved or refused.

Next, to know whether a payment is late or not, we subtract DAYS_ENTRY_PAYMENT by DAYS_INSTALMENT.

To handle outliers, we change 365.243 values DAYS_FIRST_DRAWING, DAYS_FIRST_DUE, DAYS_LAST_
DAYS_LAST_DUE, and DAYS_TERMINATION to nan.

Posh-cash

We have interest on CNT_INSTALMENT and CNT_INSTALMENT_FUTURE. First, CNT_INSTALMENT implies whether a payment was completed before initial term. And CNT_INSTALMENT_FUTURE shows installments left to pay on the previous credit.

Installments

We can calculate Days past due and Days before due from two features: DAYS_ENTRY_PAYMENT and DAYS_INSTALMENT.

- Days_past_due = DAYS_ENTRY_PAYMENT - DAYS_INSTALMENT

- Days_before_due = DAYS_INSTALMENT - DAYS_ENTRY_PAYMENT

Then we calculate flag k threshold of late payments. DPD_7 is Days past due over 7 days and DPD_15 is Days past due over 15 days.

Credit card

From this data we can extract some good features. two of them are the amount used from limit, LIMIT_USE, and the Current payment / Min payment ratio,PAYMENT_DIV_MIN.

- LIMIT_USE = AMT_BALANCE / AMT_CREDIT_LIMIT_ACTUAL

- PAYMENT_DIV_MIN = AMT_PAYMENT_CURRENT / AMT_INST_MIN_REGULARITY

Adding new features after merging

4.2 Innovation

- create new features
- normalize
- drop more features
- try with more algorithms

4.3 Home Credit Default Risk with Random Forest

4.3.1 Why dose we use Random Forest?

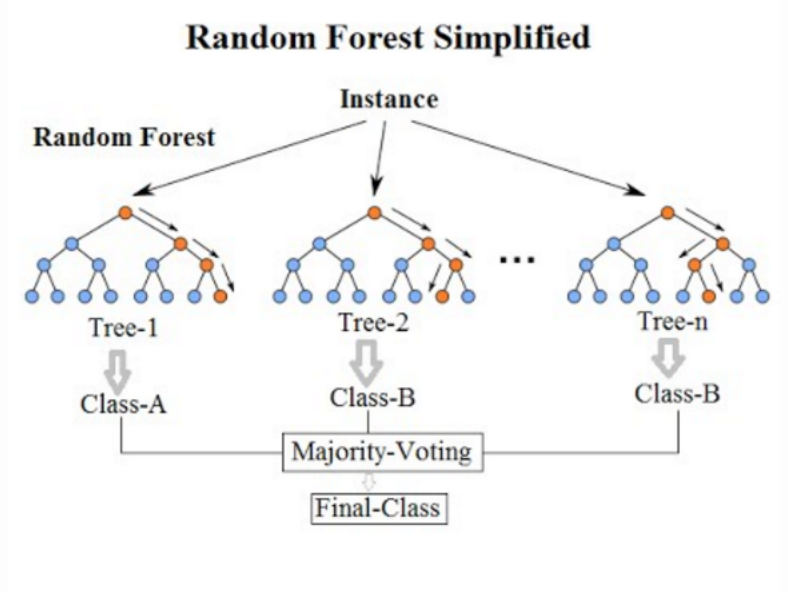

Figure 14: Explains how LightGBM works

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

4.3.2 Implementing Random Forest and result

Input data: In this model, we use all data provided by Home Credit. We preprocess all those data merge them together and use it as input data for the model.

Parameters: n_estimators = 100, random_state = 50, verbose = 1, n_jobs = -1

Result: Random Forest with merged data gives a result of 0.7123456. And the important features over Random Forest is in the following graph.

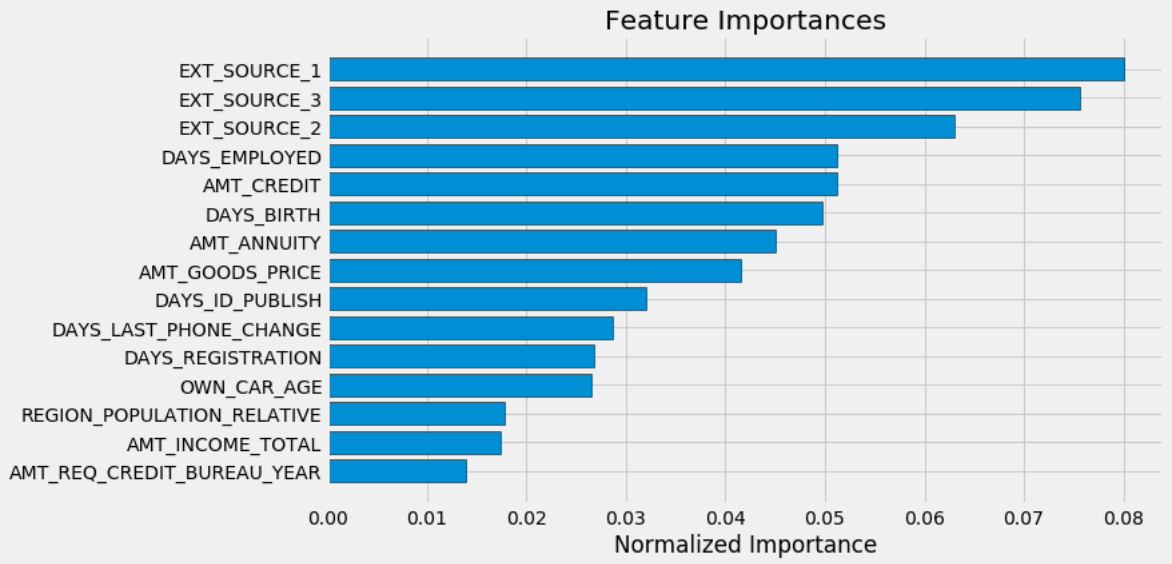

Figure 15: Explains how LightGBM works

4.4 Home Credit Default Risk with XGBoost

4.4.1 Why does we use XGBoost?

XGBoost stands for Extreme Gradient Boosting. Gradient boosting is an ensemble method that sequentially adds our trained predictors and assigns them a weight. However, instead of assigning different weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss when adding the latest prediction. So, in the end, you are updating your model using gradient descent and hence the name, gradient boosting. This is supported for both regression and classification problems. XGBoost specifically, implements this algorithm for decision tree boosting with an additional custom regularization term in the objective function.

The two reasons to use XGBoost are also the two goals of the project:

- Execution Speed
- Model Performance

Generally, XGBoost is fast. It is really fast when compared to other implementations of gradient boosting.

Secondly, XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform. For example, there is an incomplete list of first, second and third place competition winners that used titled: XGBoost: Machine Learning Challenge Winning Solutions.

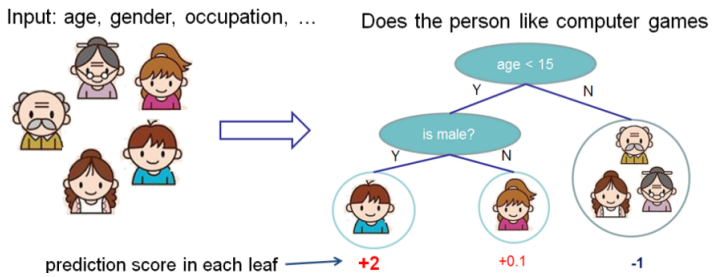

Figure 16: Explains how XGBoost works

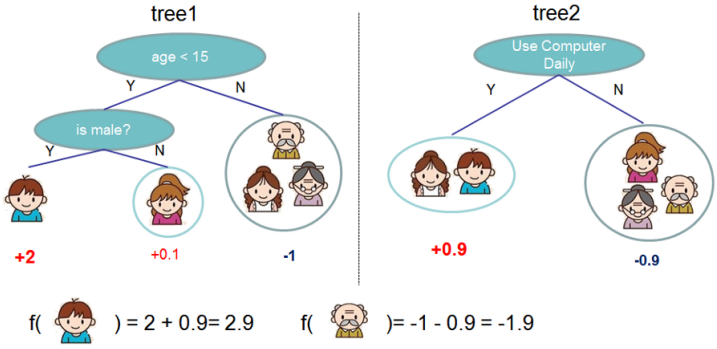

Figure 17: Tree Ensemble Model

4.4.2 Core parameters of XGBoost

- booster [default= gbtree]: which booster to use. Can be gbtree, gblinear or dart; gbtree and dart use tree based models while gblinear uses linear functions
- nthread: Number of parallel threads used to run XGBoost
- eta: step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative
- max_depth: maximum depth of a tree.
- min_child_weight: Minimum sum of instance weight (hessian) needed in a child
- subsample: subsample ratio of the training instances.
- colsample_bytree, colsample_bylevel: this is a family of parameters for subsampling of columns
- reg_alpha: L1 regularization term on weights. Increasing this value will make model more conservative.
- reg_lambda: L2 regularization term on weights. Increasing this value will make model more conservative.
- :

4.4.3 Implementing XGBoost and result

Parameters

```
XGBoost_parameters = {objective = 'binary:logistic',
          booster = 'gbtree',
          eval_metric = 'auc',
          nthread = 4,
          eta = 0.05,
          max_depth = 8,
          min_child_weight = 60,
          gamma = 0,
          subsample = 0.8715623,
          colsample_bytree = 0.7,
          colsample_bylevel = 0.632,
          reg_alpha = 0.041545473,
          reg_lambda = 0.0735294,
          nrounds = 2000}
```

Figure 18: Implementing XGBoost parameters

Result XGBoost with merged data gives a result of 0.7987654321 (edit later).

4.5 Home Credit Default Risk with LightGBM

4.5.1 Why dose we use LightGBM?

Light GBM is a gradient boosting framework that uses tree based learning algorithm. Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as 'Light' because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

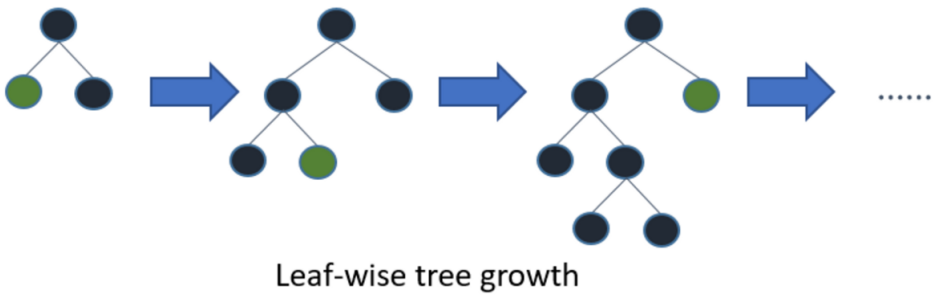

Figure 19: Explains how LightGBM works

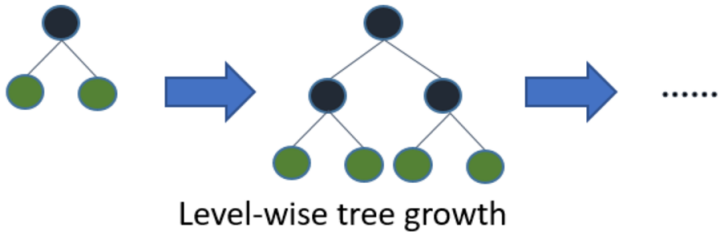

Figure 20: Explains how other boosting algorithms works

4.5.2 Core parameters of LightGBM

- boosting: defines the type of algorithm you want to run, default=gdbt
  - gbdt: traditional Gradient Boosting Decision Tree
  - rf: random forest
  - dart: Dropouts meet Multiple Additive Regression Trees
  - goss: Gradient-based One-Side Sampling
- num_boost_round: Number of boosting iterations, typically 100+
- learning_rate: This determines the impact of each tree on the final outcome.
- num_leaves: number of leaves in full tree, default: 31
- max_depth: limit the max depth for tree model
- subsample_for_bin: number of data that sampled to construct histogram bins
- reg_alpha: L1 regularization
- reg_lambda: L2 regularization
- colsample_bytree: LightGBM will randomly select part of features on each iteration if feature_fraction smaller than 1.0. For example, if you set it to 0.8, LightGBM will select 80% of features before training each tree
- min_split_gain: the minimal gain to perform split
- device: default: cpu, can also pass gpu

4.5.3 Implementing LightGBM and result

Parameters

```
LIGHTGBM_PARAMS = {
    'boosting_type': 'goss',
    'n_estimators': 10000,
    'learning_rate': 0.005134,
    'num_leaves': 54,
    'max_depth': 10,
    'subsample_for_bin': 240000,
    'reg_alpha': 0.436193,
    'reg_lambda': 0.479169,
    'colsample_bytree': 0.508716,
    'min_split_gain': 0.024766,
    'subsample': 1,
    'is_unbalance': False,
    'silent':-1,
    'verbose':-1
}
```

Figure 21: Implementing LightGBM parameters

Result LightGBM with merged data gives a result of: 0.7987654321 (edit later).

4.6 Comparison of Results

Table 2: Comparison of results

	Only application_train		*Merged data*		
Algorithms	Random Forest	LightGBM	Random Forest	XGBoost	LightGBM
Result	0.1	0.2	0.3	0.4	0.5

5 Conclusion

LightGBM gives the highest result