Stats 507 Project Final Report

Yifeng He Yanze Liu Wang Xiang

***Abstract*—In this project, we extracted financial data from online microlending platforms, including user's authorization information, credit records, bank account information, online shopping records, delivery address information, and user's personal information. Based on these datasets, we used random forest, XGBoost, light GBM and fully connected neural networks to predict the probability of user default. Then prediction results are shown in confusion matrices, and models' performances are evaluated by AUC score, and in the end, top 20 important features are selected to perform predictions.**

I. Introduction

Our project is to develop a machine learning data pipeline to process financial data from online microlending platforms, and help them to estimate user's loan risk, and to decide whether or not to put loans. The significance of our project is that with the development of internet finance, many people choose online loans because of its convenience and high acceptance rate. However, because it is not a face-to-face loan, many online microlending platforms find that it is difficult for them to judge whether there is a risk of default, and currently, this assessment is done manually, which is time-consuming and high cost. If we could rely on machine learning models to make judgements based on the user's past financial records, the evaluation results will be more accountable and stable.

Group 12
Apr 26, 2020

A. *Dataset Overview*

The financial dataset is from an online competition database.It includes 7 csv files, which are "authorization info", "bankcard info", "credit info", "order info", "receive address info", "target", and "user info". Details of features that are intended to be used for each file are shown below:

Auth_Info								
Loan ID	ID_Card	Authorized Time for Loan	Authorized Phone					
Credit_Info								
Loan ID	Credit Score	Quota	Overdraft					
Receive_addr_info								
Loan ID	Address ID	Receive Region	Receiver Phone	Receiver Fixed Phone				
Backcard_info								
Loan ID	Bank Name	Card Type	Bind Phone Number					
Order_info								
Loan ID	Order Amount	Type Pay	Order Status	Unit Price				
User_info								
Loan ID	Sex	Birthday	Hobby	Marriage	Income	Degree	QQ account	WeChat account	Account Level
Target								
Loan ID	Loan Application Submission Time	Target						

Fig. 1. Dataset Information

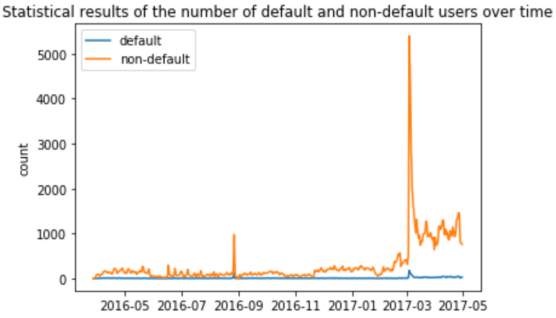

Fig. 2. Time Series Features

B. *Data Exploration*

Part 1: Understand the features.

The graph above shows that there are two spikes for the number of applicants, indicating a lot of people apply for a loan for some reason, and we can also notice that there is more data in 2017 than in 2016. The reason is that after 2017, with the development of internet banking, People gradually accepted the way of making loans online.However, this data feature also implies that it is also hard to make predictions for the recent loan applications based on the past financial data because of the uneven distribution of data in time.

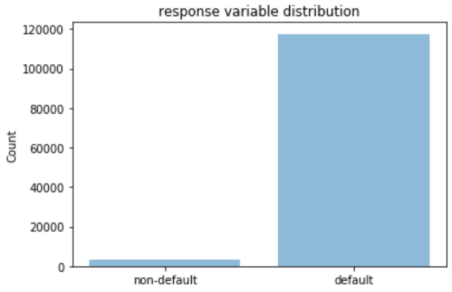

Fig. 3. Data Overview

In this dataset, there are 117,541 non-default records and 3388 default records, which is 34.69:1, and the graph above also indicates that this is a very imbalanced dataset. That is why we decided to use auc score as measurement instead of accuracy because we want to focus more on machine learning model's predictability on those default records.

Part 2: Data cleaning and Feature Preprocessing.

1. train auth info.

This data table has personal information such as id card num-

ber and phone number. The phone number cannot be directly used because only the first three and last four digits are kept. In China, different communication companies have their own three-digit starting numbers, so, based on this fact, we divided phone numbers into different communication companies. As for id card numbers, we took its first number to indicate the region. For instance, '1' stands for Northern China, '2' stands for Northeastern area, and '3' stands for Eastern area in China. Besides, we extracted the authorization time as year, month, and day as new features. To deal with missing value, we used 0/1 to indicate whether the user submits id card number and phone number. 0 means missing and 1 means existing, and we created 2 new features: 'spt_id_card' and 'phone_exist' to store them. In addition, we replaced missing values of 'auth_time' with NaT so Pandas could recognize it as missing values for the further data processing.

2. train bankcard info.

This table has the information of users' bank cards, bank name and reserved phone numbers. First, we converted the bank name and bank card type from Chinese to English. Then, we counted the number of different types of bank cards, the number of cards of different types and the number of reserved phones for each user. Then we saved the result to new features as 'bankcard_card_count', 'bankcard_bank_count' and 'bankcard_phone_count'. Besides, we divided phone numbers into communication companies as well. After dealing with all features, we deduplicated this table.

3. train credit info.

This table has the information of users' credit score, quota and overdraft amount. We calculated unused amount and amount usage rate, then saved them in new features as 'remaining' and 'using_rate'. Moreover, we created another new feature called 'inverse_credit_score', which is subtracting existing credit scores from full credit scores so people with high credit scores will have lower weight in our model for predicting the probability of default.

4. train order info.

This table has the information about users' online shopping behavior. One feature of this dataset is that each user has several shopping records so we need to aggregate these features to represent the shopping behavior of individual users. Firstly, we formulated the 'time_order'(the time of order) as standard datetime type, and obtained each user's oldest and latest transaction dates. In addition, we also calculated the sum, mean, std, skewness of amt_order(the amount of order) and saved them in new columns as 'order_sum', 'order_mean', 'order_std' and 'order_skew'. Besides, the number of phone numbers each user has was also calculated and was saved as a new feature 'pho_num'. Moreover, we counted the number of people that each user ordered for as a new feature 'rec_num' because we think this can reflect the economic condition of users to a certain extent. Furthermore, We counted the number of transactions completed as 'sts_order_finished' and counted the number of orders as 'amt_order_num', and the payment method is divided into 'online_pay', 'mixed_pay', 'cash_on_delivery' and 'loans'. Then, the number of 'online pay', and 'cash on delivery' each user has was counted, and was saved as new features called 'train_order_online_pay' and 'train_order_cash_on_deliv'. After dealing with all features, we deduplicated this table.

5. train recieve info.

This table has the receiver's address information, such as address, region and phone number. First we converted the region name from Chinese to English, and one-hot encoded it based on different provinces. Then we counted the number of addresses each user has and saved it in a new feature called 'addr_num'. We think a large number of delivery addresses may indicate that the user lives in an unstable condition. Besides, we also counted the users' number of phone numbers and the rate of submitting their phone numbers. These are saved in new features called 'fxpho_num' and 'fxpho_rate'.

6. train user info.

This table has users' personal information such as birthday, marital status, income and so on. In this dataframe, we mapped whether a user has indicated their two social software(QQ and Wechat) accounts to numbers. 'Bounded' to 1 and 'Not Bounded' to 0. We also mapped the account grade to numbers. 'Initial Level':1, '3rd Level':2, '2nd Level':3, '1st Level':4, 'Premier Level':5. In addition, we also converted the user's birthday to datetime type, and extracted different forms of birthday as new features, such as '0-0-0', '1-1-1', '–', and '0000-00-00' types. Based on the user's birthday, and loan application submission time, we also calculated the user's age as a new feature. Further, in order to process categorical data, which are 'sex', 'marriage', 'income', 'qq_bound', 'degree', 'wechat_bound', and 'account_grade', we appllied label-encoding for these features in lightgbm model, and used one-hot-encoding for other mathine learning models. The reason using different encoding here is that the lightgbm model could process categorical data by itself.

After processing each file, we merged these all features together, and created a whole new dataset. In this new dataset, we also created many new features. For example, we extracted 'hour', 'month', and 'year' from user application submission time, and calculated the difference between application submission time and the latest shopping order, saved as the new feature called 'day_order_max'. Similarly, we also calculated the difference between application submission time and the oldest shopping order, saved as the new feature called 'day_order_min'. After creating new features, we also deduplicated this dataset.

Part 3: New Dataset Overview.

First, we calculated the missing rate for each feature, and the results are shown below.

	feature	missing rate
22	is_0001_1_1	1.000000
19	is_double	1.000000
21	is_0000_00_00	1.000000
20	is_0_0_0	1.000000
59	degree_PhD	0.999868
58	degree_Others	0.999760
56	degree_Master	0.999496
41	merriage_No Disclosure	0.999214
47	income_Below 2000	0.998652
57	degree_Middle School	0.995882
60	degree_Special Secondary School	0.989333
55	degree_High School	0.988770
53	degree_Bachelor	0.987025
46	income_6000-7999	0.986306
44	income_2000-3999	0.985471
48	income_above 8000	0.979310
45	income_4000-5999	0.973819
69	account_grade_Premier Level	0.973141
39	sex_unknown	0.971769
54	degree_College	0.970627
40	merriage_Married	0.969470
64	wechat_bound_unknown	0.965385
52	qq_bound_unknown	0.965385
42	merriage_Single	0.957430
65	account_grade_1st Level	0.936756
36	sex_Female	0.932018
27	is_industry	0.931414
28	is_idcard	0.925502
66	account_grade_2nd Level	0.901289
26	is_hobby	0.874579
32	unit_price_mean	0.871148

Fig. 4. Feature Missing Rate

From the graph above, we could see that many one-hot features have a high missing rate, which makes sense. One side-effect of one-hot encoding is that it will create a sparse matrix. Thus, we deleted features with high missing rates except features from one-hot encoding, like 'is_industry', 'is_idcard', 'is_hobby' and 'unit_price_mean'. Although we kept most one-hot features, 'is_0001_1_1', 'is_double', 'is_0000_00_00', and 'is_0_0_0' are still deleted because there are no records for these features at all.
Then, we also used heatmap to explore the correlations among variables.

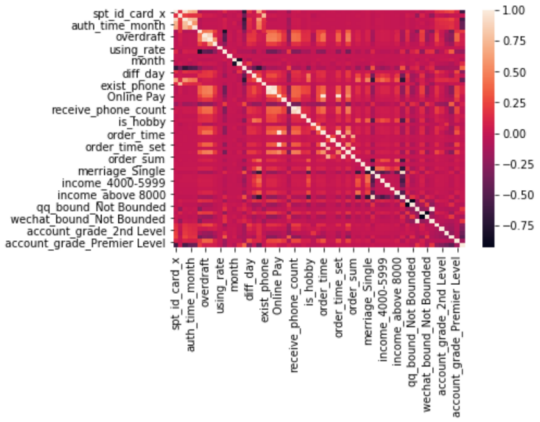

Fig. 5. Correlation Exploration

The plot above shows that many features are not correlated, so we can use this dataset directly in our ML models. After doing data preprocessing, our new dataset has 114 features in the end.

C. Model Training and Result Evaluation

Part 1: Split Dataset.

The provided dataset has user's financial records from Mar, 2016 to May, 2017, and has 120,929 observations in total. Considering the time-series feature of this model, we split the whole dataset into training set, and test set based on the time. Training set includes data from Mar, 2016 to May, 2017, and has 87174 observations in total. Test set includes financial records on Apr, 2017, and has 33755 observations. The ratio of training set and test set is 2.58:1.

Part 2: Model Training and Selection.

In this part, we applied XGBoost, Random Forest, light GBM, and Fully connected Neural Network to do predictions. We will firstly give a brief introduction of each method. Then, we evaluated each model's predictability based on AUC score, and a confusion matrix is used to show predicted results.

XGBoost.

The XGBoost library implements the gradient boosting decision tree algorithm[1]. This algorithm is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees[2]. It builds the model in a stage-wise fashion like other boosting methods do, and generalizes them by allowing optimization of an arbitrary differentiable loss function[3]. We think XGBoost can effectively help us to process the original data with a large number of variables, and do feature selection by selecting the variables with higher importance. Since our dataset is imbalanced, we need to assign "scale_pos_weight' to make our models focus more on the minor data. After applying XGBoost to our dataset, the accuracy score for the XGBoost model is 89.62%, and the valid auc score is 0.620. The plot below shows the ROC-AUC curve for the XGBoost model.

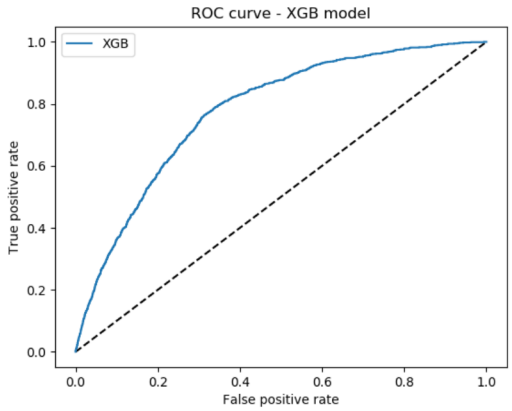

Fig. 6. XGB ROC curve

The plot above shows that the ROC curve for XGBoost is better than the baseline, so XGBoost model does have the ability to make predictions based on the dataset, but the auc

score 0.62 indicates it may not be a good option to make predictions here. Further, the smoothness of our curve implies that there is no overfitting here.

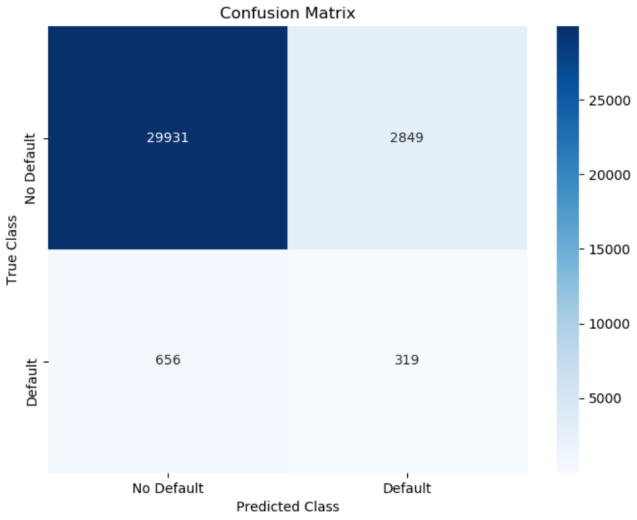

Fig. 7. XGB confusion matrix

The confusion matrix above also shows the imbalance of our data, and the model is more likely to predict '0' rather than '1'. This issue can be solved by assigning larger weight to 'default' records in our model. However, the tradeoff is that the model will also tend to attribute 'non-default' users to 'default' users.

Random Forest.

Random Forest is a good algorithm in dealing with imbalanced datasets. Besides, since the dataset contains lots of NaN values, we need to use a tree model before converting categorical variables into numerical variables. The advantage of random forest includes: it provides a reliable estimate as well as it offers efficient estimates of the test error without incurring the cost of repeated model training associated with CV. The disadvantage of this method includes: it is less interpretable, training a large mode is costly and it is difficult for us to tune hyperparameters. After applying the model, we got 0.971 oob_score (out of bag score), which means the model performs well in our dataset. Besides, the AUC score is equal to 0.71, which is better than the score of XGBoost. The corresponding ROC curve is shown below.

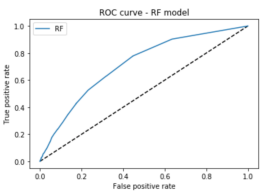

Fig. 8. Random Forest ROC curve

Again, the picture shows that the random forest model has predictability on our dataset, and the valid auc score 0.71 indicates that it is an acceptable model.

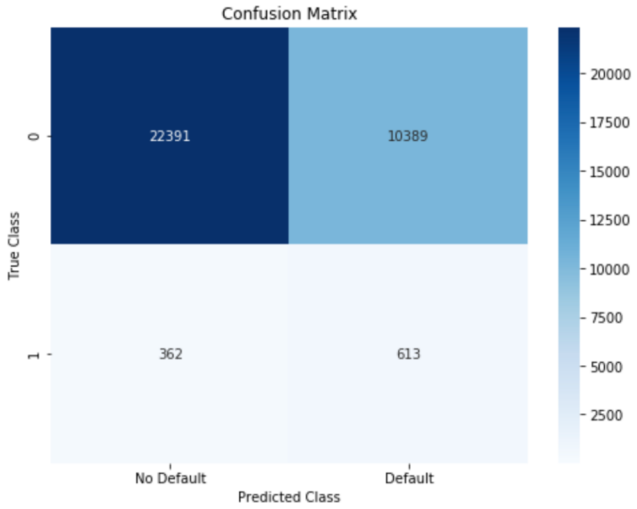

Fig. 9. Random Forest confusion matrix

The confusion matrix shows that although most 'non-default' users are correctly identified, many 'non-default' users are also labeled as 'default'. In the real application, this may help microlending platforms identify those default applicants, but it will also lead to the problem of low loan pass rate, and reduce platform earnings in the end.

Light GBM.

The reason that we applied light GBM here is that compared to other machine learning models, and boosting algorithms, it has (1) faster training speed and higher efficiency, (2) lower memory usage, (3) high accuracy, (4) support of parallel and GPU learning and (5) capable of handling large-scale and categorical data. [4] In general, light GBM is a gradient boosting framework based on decision tree algorithm, which is usually used for ranking, classification and many other machine learning tasks. [5] Unlike other boosting algorithms which split the tree depth wise or level wise, light GBM grows tree leaf-wise. When fit as a model, light GBM will choose the leaf with max delta loss to grow. Thus, when growing the same leaf, leaf-wise algorithm can reduce more loss than a level-wise algorithm. [6] One disadvantage of light GBM is that it is not good at fitting small datasets since it is sensitive to overfitting. To avoid this, it is better to use data with over 10,000+ rows. In our case, since we have a fairly large dataset, it wouldn't be an issue here. In addition, since our dataset is imbalanced here. We also assigned 'scale_pos_weight' to our model, which is the ratio of 'non-default'/'default'. The overall accuracy is 0.876. The ROC curve of the light GBM model is shown below.

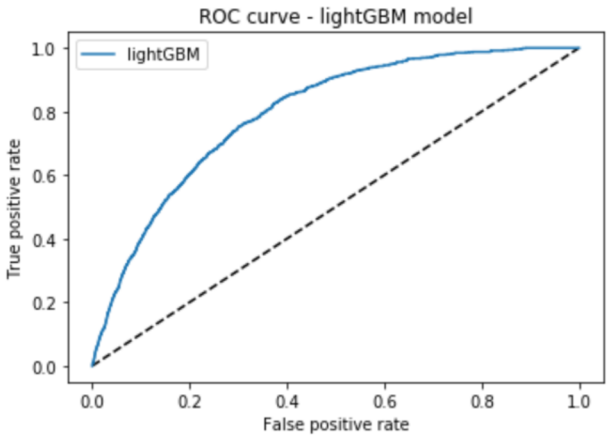

Fig. 10. LightGBM ROC curve

The valid AUC score for the Light GBM model is 0.79, which is better than the other two models, and the corresponding confusion matrix is also shown below.

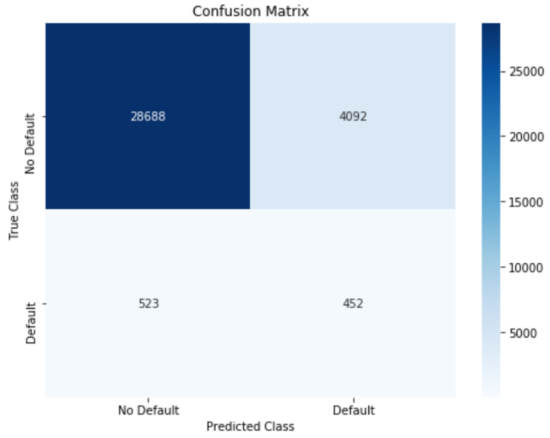

Fig. 11. LightGBM confusion matrix

The confusion matrix shows that the Light GBM identifies half of default applicants correctly, and at the same time, not many non-default applicants are wrongly labeled as 'default'. Since this mode performs better than the other two models, we decided to optimize this model by tuning the hyperparameter of Light GBM.

Seven hyperparameters are chosen to be optimized by GridSearchCV: 'max_depth' and 'num_leaves' control the tree size and complexity. [4, 6, 8] are tried for 'max_depth', and [20, 30] are attempted for 'num_leaves'. 'Feature fraction' helps the model avoid overfitting by randomly selecting features with a specific ratio to the model. [0.5, 0.7, 0.9] are tried here. 'Reg alpha' and 'reg_lambda' are parameters for L1 and L2 regularization respectively. By adjusting these two parameters can also help avoid overfitting, and [0.1, 1, 10] are tried for both parameters. 'Scale_pos_weight' controls the weight of 'default' applicants in our model. The higher the weight, the more likely the model to predict 'default' users. [15, 25, 35] are inputs for this parameter. 'Num_iterations' controls the number of trees Light GBM deployed for predictions. More num_iterations could achieve better accuracy, and may lead to overfitting as well.

After running GridSearchCV, the best parameters are 'feature_fraction': 0.5, 'max_depth': 4, 'num_iterations': 200, 'num_leaves': 20, 'reg_alpha': 10, 'reg_lambda': 10, 'scale_pos_weight': 15.

Train the Light GBM model again with these parameters, and apply it to the test data set. We get the AUC score 0.812, which is slightly better than the score of the previous Light GBM model, and the corresponding ROC curve is shown below.

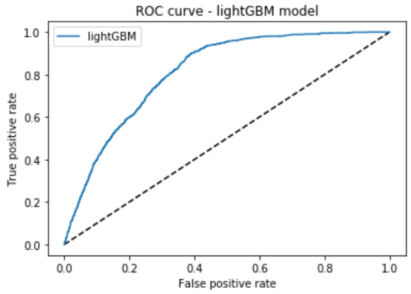

Fig. 12. LightGBM ROC curve

The curve is pretty flat when it is close to 1, indicating this Light GBM model has very good predictability here, and the corresponding confusion matrix is also shown below.

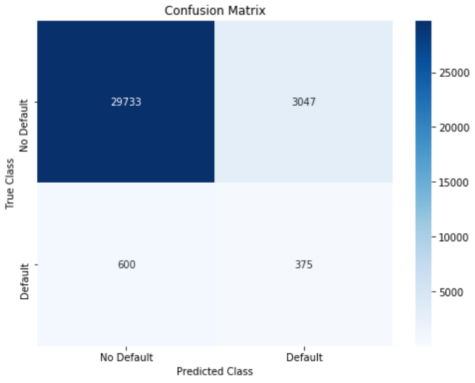

Fig. 13. LightGBM confusion matrix

There is not much information provided here. But same as before, when the model predicts more 'default' applicants as 'non-default' users, less 'non-default' applicants are labeled as 'default'. In short, Light GBM shows the best predictability in our dataset so far.

Neural Network.

Besides bagging and boosting algorithms, we also applied neural networks of fully connected layers in our project. Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to to classification.[7]

To deal with overfitting, we use dropout. Dropout is a regularization technique patented by Google for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term ”dropout” refers to dropping out units (both hidden and visible) in a neural network.[8] During training, some number of layer outputs are randomly ignored or “dropped out.” This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer.

Further, in order to speed up the process of training deep neural networks, we use batch normalization. Batch normalization (also known as batch norm) is a technique for improving the speed, performance, and stability of artificial neural networks.[9]Batch normalization reduces the amount by which the hidden unit values shift around (covariance shift). Besides, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.[10]

First, we tried the basic deep learning models without assigning weight, which has two layers, 64 nodes in each layer, and activation functions are all Relu. The confusion matrix is shown below.

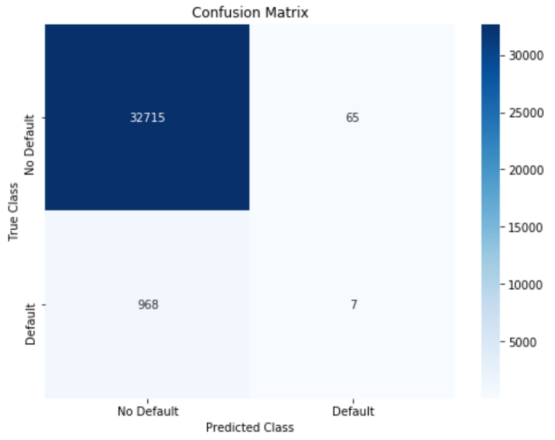

Fig. 14. NN confusion matrix

The confusion matrix shows that in order to achieve better accuracy, most applicants are labelled as ‘non-default’, and the precision and recall score for the ‘default’ users is 0.1 and 0.01 respectively, which is pretty low, so we need to assign weights to make our model focus on the ‘default’ users. In the new deep learning models, we assign the weight ‘35’ to the loss function, which is the ratio of the number of ‘non-default’ and ‘default’ applicants. After running 50 epoches, we get the valid AUC score 0.75, and precision and recall scores for ‘default’ users are 0.09 and 0.36 respectively, which is much better than the previous model. The corresponding confusion matrix is shown below.

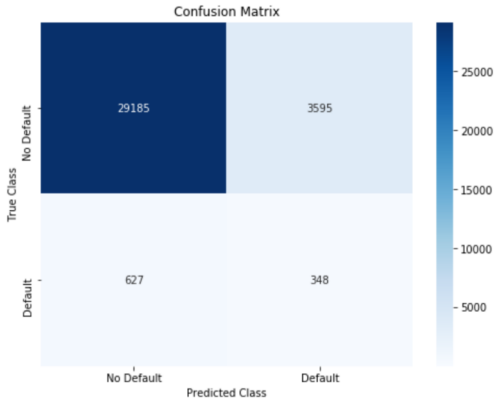

Fig. 15. NN confusion matrix

The confusion matrix agrees with the AUC score. It successfully identifies 35% ‘default’ applicants, and the corresponding ROC curve is shown below.

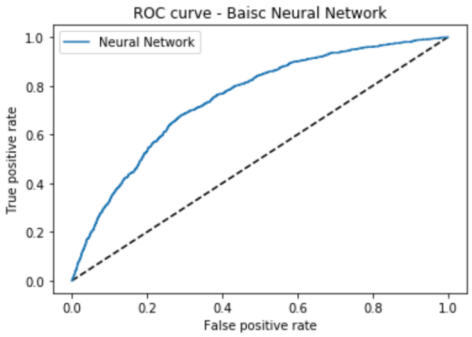

Fig. 16. NN ROC curve

The curve is not as good as the curve of Light GBM, so we decided to optimize this deep learning model by changing its hyperparameters, like number of layers, number of nodes, learning rate, and the number of epochs. After several iterations, we got the optimal model by adding one more layer, changing the number of nodes to be 256 for the first layer, 128 for the second layer, and 64 for the third layer. Further, we reduced the number of epochs to be 20 to avoid overfitting, change the batch size to be 128, and learning rate to be 0.005. The new model’s valid AUC score is 0.76, and the corresponding ROC curve is shown below.

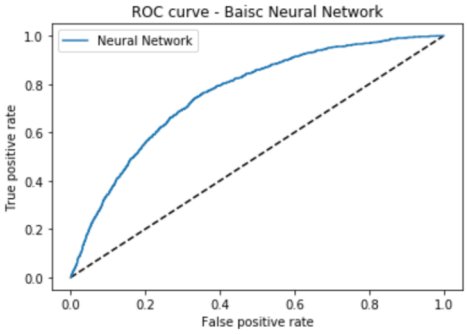

Fig. 17. NN ROC curve

The roc curve is slightly better than the previous model,indicating that if we want to get better results, we may need to try other deep learning models, like recurrent neural networks. The confusion matrix for this model is also shown below.

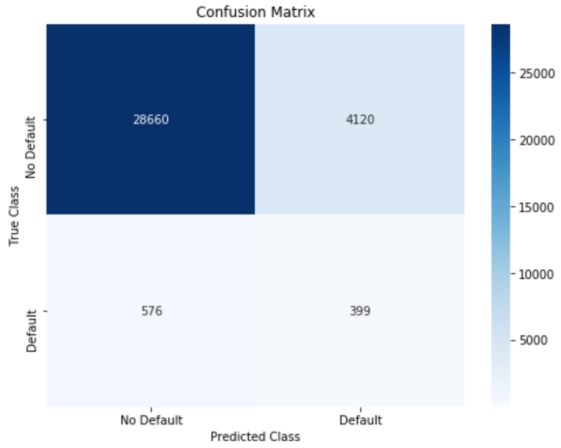

Fig. 18. NN confusion matrix

From this confusion matrix, we cannot really tell how much this model has been improved, but it gives us a visualization of our model predictability.

D. *Feature Selection*

For the feature selection part, since XGBoost and lightGBM model could rank the importance of parameters during model training, we mainly rely on these two methods to do feature selection.

XGBoost.

In the XGBoost model, the feature importance is sorted in descending order, and top20 is shown in the graph. The results are basically consistent with our judgment based on common sense. The feature 'diff_day' is the time interval between the application date and the approval date. For the fraudulent

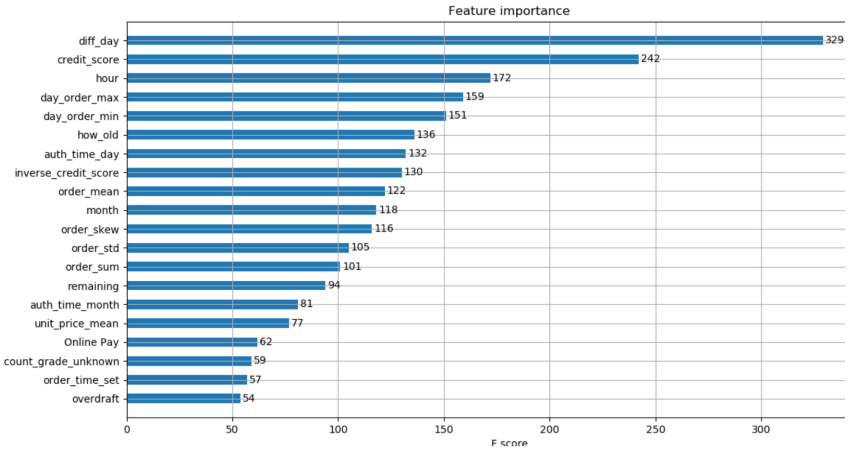

Fig. 19. Feature Selection from XGBoost

user, due to the strong purpose, the loan may be carried out immediately after the authentication; while for the general user, the loan application may be carried out after a period of authentication. 'credit_score' is an intuitive indicator. 'Hour' is the specific time of application, it means for fraudulent users, they may like to perform loan applications on specific time. The reason that 'day_order_max' and 'day_order_min' are important is that there were more defaults in 2017 than in 2016.

LightGBM.

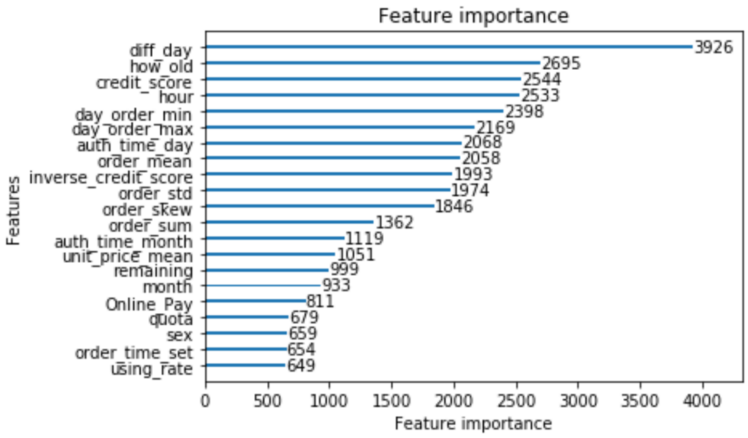

Fig. 20. Feature Selection from LightGBM

Except for some order differences, the top 20 important features selected from the light GBM model generally conforms with the XGBoost results. Based on these two results, we could roughly know which parameters are important for prediction. Then, we use these top 20 features from the Light GBM model to do predictions again. The valid AUC score for the new model is 0.73, which is worse than the our optimized model, and the corresponding confusion matrix is shown below.

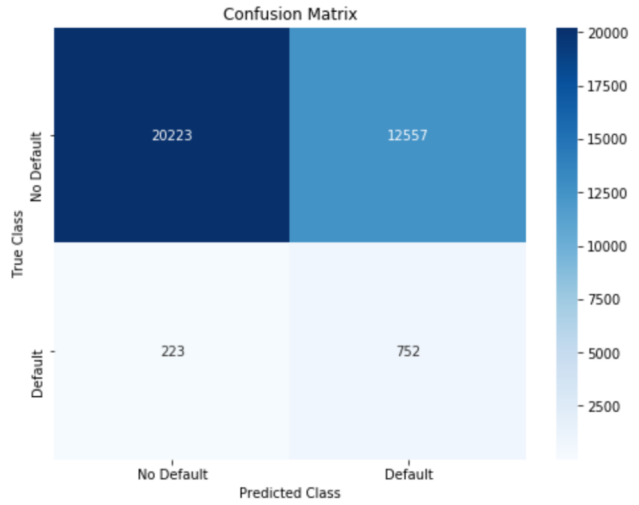

Fig. 21. LightGBM confusion matrix

The confusion matrix shows the interesting feature of this model: it identifies most 'default' users. However, at the same time, it also labels almost one third of 'non-default' applicants as 'default'.

After that, we also consider the accuracy of feature selection. Because of the imbalancement of the data set, we are not sure whether it could influence the feature selection process, so we read some papers and found a method which could eliminate the influence of imbalancement.

Firstly, we divide the entire dataset into two parts according to target (0 or 1). We could find the number of data points with target 0 is 36 times bigger than the number with target 1. So we divided data points with target 0 into 26 parts and use every part to combine with the data set with target 1 to form a new dataset. Therefore, with every new dataset we constructed a new light GBM model and found the first 20 important features, and we gave 1 point to each of them. After running 36 models, we rank features with points and choose the first 20 of them. By comparing the features selected from the previous method, we could find the two methods have the same result, so the imbalancement of the dataset has almost no influence in the feature selection process in this problem.

Feature Importance Rank(Light GBM model)

TABLE I
IMPORTANCE RANK

Feature	Meaning	Importance Rank
diff_day	Difference time between authorization time and application submission time	1
credit_score	Personal credit score	2
month	Month of application submission time	3
inverse_credit_score	The difference between credit score and the upper bound of credit score	4
day_order_max	Difference between application submission time and latest order history	5
remaining	Difference between quota and overdraft	6
account_grade	User's account grade in e-commerce website	7
day_order_min	Difference between application submission time and oldest order history	8
hour	Hour of the application submission time	9
how_old	The age of applicants	10
unit_price_mean	Average number of total online shopping consumption	11
qq_bound	Whether user linked his e-commerce account to social media account	12
sex	The sex of applicant	13
order_skew	The skewness of user shopping data distribution	14
tail_num_len	Indicator of number of bank cards	15
order_mean	The mean of order history	16
quota	Applicant's quota	17
receive_phone_count	Number of reserved phone on bank	18
degree	Education level of applicant	19
order_time	Applicant's shopping times	20

E. Model Improvement by Imputation

Firstly, we need to deal with nan value. Before that, we did not deal with it because tree models regard nan value as one label in a feature. But we need to test whether nan value might influence the result.

We choose the random forest method to fill all missing data, which is recommended by lots of papers. We try the MissForest package in R to impute missing values. The function 'missForest' in this package is used to impute missing values particularly in the case of mixed-type data. It uses a random forest trained on the observed values of a data matrix to predict the missing values. It can be used to impute continuous and/or categorical data including complex interactions and non-linear relations. It yields an out-of-bag (OOB) imputation error estimate without the need of a test set or elaborate cross-validation. It can be run in parallel to save computation time.

By imputing all missing values, we get a new dataset and we use the new dataset to build machine learning models. Due to the good performance of Light GBM, I try Light GBM and test whether the imputing method could improve the performance of Light GBM. From the result, we are able to find that the best AUC score becomes 0.83, which is a little better than the method without imputing.

II. Conclusion

TABLE II
Model Score

	XGBoost	RF	Light GBM	NN
Best AUC Score	0.62	0.71	0.812	0.76

The above table shows the best AUC score each machine learning model has. From the table, we could see that the Light GBM model has the best performance, and 0.812 indicates this model has excellent predictibility. However, for the real application, There is a trade-off between identifying more default users and loan approval rates. The microlending platform should select suitable machine learning models based on the provided confusion matrix.

In the end, since there is a huge gap between the number of observations on 2016 and on 2017, the AUC score could be improved by providing more and recent observations for training. Further, for missing values, we could also use other methods like oversampling and undersampling to deal with it, and check its effect on AUC score.

References

[1] XGBoost. (2020, March 28). Retrieved from https://en.wikipedia.org/wiki/XGBoost

[2] Brownlee, J. (2019, August 21). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[3] Gradient boosting. (2020, April 4). Retrieved from https://en.wikipedia.org/wiki/Gradient_boosting

[4] Microsoft. (n.d.). microsoft/LightGBM. Retrieved from https://github.com/microsoft/LightGBM/blob/master/docs/Features.rst#references

[5] Khandelwal, P., & Analytics Vidhya. (2020, March 27). Light GBM vs XGBOOST: Which algorithm takes the crown. Retrieved from https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

[6] Mandot, P. (2018, December 1). What is LightGBM, How to implement it? How to fine tune the parameters? Retrieved from https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

[7] Convolutional neural network. (2020, April 26). Retrieved from https://en.wikipedia.org/wiki/Convolutional_neural_network

[8] Dropout. (2019, December 11). Retrieved from https://zh.wikipedia.org/wiki/Dropout

[9] Batch normalization. (2020, April 7). Retrieved from https://en.wikipedia.org/wiki/Batch_normalization

[10] D, F. (2017, October 25). Batch normalization in Neural Networks. Retrieved from https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

Contributions.

Wang Xiang:

1. Data cleaning: Transform the value of features to the type that the model can use. Deal with the missing value.
2. Build XGBoost model and do model evaluation.
3. Make a presentation slide and do a presentation.
4. Write the preliminary and final report

Yanze Liu:

1. Create new features, applying random forest to deal with missing values, combine different datasets into a big dataset.
2. Using PCA to find the relationship between different data points, and choose suitable methods according to this.
3. Random forest, and optimize it with hyperparameters
4. Improve the feature selection method by considering the influence of imbalancement
5. Light GBM with the improved dataset, and make a comparison between new Light GBM and old methods.
6. Write the preliminary and final report, focusing on model improvement, feature selection, random forest, and conclusion

Yifeng He:

1. create new features, and combine all features to create the new dataset.
2. Data Preprocessing: Create heatmap, show the response variable's distribution based on time, and calculate the missing rate for each feature.
3. Light GBM, and optimize it with hyperparameters
4. Neural Network, and Optimize it with hyperparameters
5. Do a presentation
6. Write the preliminary and final report, and use latex to show it