Frontiers in Business, Economics and Management
ISSN: 2766-824X | Vol. 5, No. 1, 2022

Prediction of Car Loan Default Results Based on Multi Model Fusion

Zhaoyang Hong, Wenxuan Deng, Xiuyu Gong

The Hong Kong University of Science and Technology, Kowloon, Hong Kong, China

Abstract: With the prosperity and development of the asset management industry and various financial derivatives, many micro-loans and online loans have gradually entered the public view. How to predict the default probability of customer loans is a hot topic in the market. Therefore, in this paper, by collecting the data profile of more than 10 thousand car loan borrowers and fitting the fusion model of 4 methods: logistic model, decision model, Random Forest, and KNN model to the data, the author examines the behavioral data of borrowers to predict whether the borrowers will default in the future and find the best threshold to reach the lowest cost. The findings indicate that our final prediction can reduce costs by 38.9%. The excellent result shows that this model can be applied to the real market to help lending institutions predict default results and formulate strategies to avoid default risks in the process of borrowers' evaluation according to the model coefficient.

Keywords: Loan default, Multi-model fusion, Credit default, Credit risk, Machine learning.

1. Literature Review

With the deep development of big data and programming tools, classification and regression models are used to predict credit crisis [1]. Galindo used machine learning to research credit risk and got the conclusion that decision tree performed best in terms of classification results among the decision tree, neural networks, and k-nearest neighbor methods in the process of credit default prediction [2]. Malekipirbazari et al. conducted an empirical study in the Lending Club dataset and concluded that in the process of predicting borrower defaults, the random forest model outperformed FICO scores and Lending Club's own credit rating methodology [3]. Some scholars have improved loan default prediction by fusing different models [4,5,6], including GBDT, decision tree, XGBoost, and other models. All of the above previous studies provide the basis for this paper to integrate four models: logistic model, decision model, Random Forest, and KNN to predict the car loan default probability. The results are presented in the following sections.

Related models and methods:

Random forest is a classifier that uses multiple decision trees to train and predict samples. In the process of classification, for each node of the base decision tree, a subset containing k attributes is first randomly selected from the set of attributes of that node, and then an optimal attribute is selected from this subset for division, where the parameter k controls the degree of randomness, generally k=log2d [7]. In a random forest, each decision tree is disjoint, and the final classification result is determined by the plurality of the results obtained from each decision tree.

2. Introduction: Business Goal

Loan is an essential business for many financial institutions. Revenue can be earned from the commission fee (Interests) on any outstanding loan while a loss will be incurred if a loan defaults. Thus, whether to approve a loan of a certain amount to a certain customer becomes important for such financial institutions to maximize profit.

We hope to build a machine learning model to predict whether a customer will default in the future, given some key information provided. So that the institution can decide whether to approve or reject a loan application according to the prediction (Approve if non-default and vice versa).

Our models target to make predictions that maximize profits. And we would also like to conduct further analysis on our models to gain some insight into feature importance and data selection. Based on our findings, we attempt to propose some explanations and business suggestions for the financial institution.

3. Data Understanding and Processing

3.1. Dataset Description

Source: https://www.kaggle.com/saurabhbagchi/dish-network-hackathon?select=Train_Dataset.csvw

Label: 0 (Not default), 1 (Default), the distribution is shown in the pie chart (Exhibit 1)

Features: there are 38 features (meaning shown in the data dictionary/progress report) with different missing% as listed below (Exhibit 2). Considering the large missing proportion and feature contribution, we have dropped some variables at the beginning such as 'Own_House_Age' and 'Social_Circle_Default'. As 'Score_Scource_1' is relatively highly correlated to the label, we decide to keep it for now.

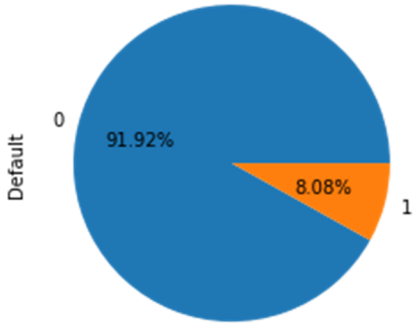


Figure 1. The distribution of default and non-default

	percent_missing
Client_Income	2.277737
Car_Owned	2.310393
Bike_Owned	2.718589
Active_Loan	2.441016
House_Own	2.563475
Child_Count	2.408360
Credit_Amount	2.245081
Loan_Annuity	3.314556
Accompany_Client	1.004164
Client_Income_Type	2.441016
Client_Education	2.196098
Client_Marital_Status	2.236917
Client_Gender	1.600131
Loan_Contract_Type	2.481835
Client_Housing_Type	2.538983
Population_Region_Relative	3.469671
Age_Days	2.367540
Employed_Days	2.269573
Registration_Days	2.212426
ID_Days	3.935015
Own_House_Age	55.800474
Mobile_Tag	0.000000
Homephone_Tag	0.000000
Workphone_Working	0.000000
Client_Occupation	22.834517
Client_Family_Members	1.608295
Cleint_City_Rating	1.616458
Application_Process_Day	1.657278
Application_Process_Hour	2.571639
Client_Permanent_Match_Tag	0.000000
Client_Contact_Work_Tag	0.000000
Type_Organization	2.351212
Score_Source_1	48.436607
Score_Source_2	3.575802
Score_Source_3	19.470977
Social_Circle_Default	46.322149
Phone_Change	2.628786
Credit_Bureau	14.899175

Figure 2. The missing percentage of 38 features

3.2. Process Methodology

Filling in the missing values:

Most of the numerical missing values are filled with average group by correlated features

Most of the categorical missing values are filled with mode group by correlated features

Normalization: Observing the distribution of different features, for more centralized and normally distributed variables, we apply Z-Normalization; for those concentrated in a small range, we adopt log transformation. (Exhibit 3)

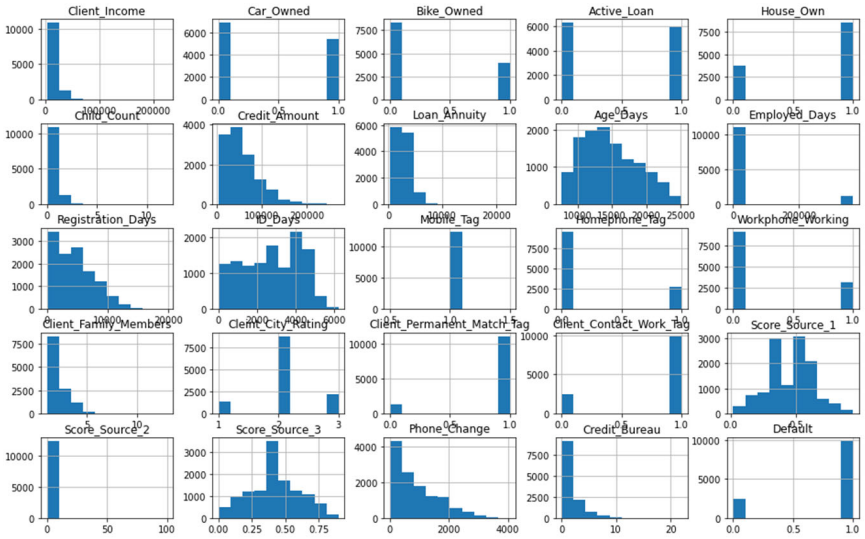


Figure 3. The visualization of concentrated variables

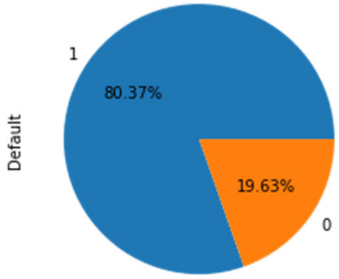


Figure 4. The distribution of default and non-default after dropping some non-default records with "NA"

Constricting a new data set:

We have found that 'default' records take around 8.08% in the raw data. (Exhibit 4) To help model better recognize the default patterns, we try to drop the non-default records with 'NA' features and get a dataset with 80.37% defaults (total 12249 records).

[please refer to Data _Processing_Version1 and Data Processing_Version2(D)]

And we build each type of model on both datasets:
(1)The original one with all features :train
(2) The one with Non-default NA dropped: train_1

4. Model Training and Evaluation

4.1. Cost and Benefit Analysis

Assumption

Commission fee: 6% (for non-default customers)

Exposure at Default (EAD) or Loan Amount per account: median of the credit amount

Loss given Default (LGD): 55% (refer to the HKMA 45% recovery for unsecured loan)

Financial Institution will reject all customers with 'Default' prediction

Opportunity Cost

Cost=# (False Positive) *Commission fee * Loan Amount + # (False Negative) *EAD*LGD

Goal: As the financial institution would like to maximize their profits from the car loans, we will select the optimal model threshold to minimize the opportunity cost in the later stage.

4.2. Decision Tree

Data Source: train_cleaned.csv("train"), train_1_cleaned.csv ("train_1")

We prepared two datasets and decided to build one models on each, then we would compare the two models and adopt the better-performing one. The 'train' dataset is the original one which contains all the features, while the 'train_1' dataset is the one whose non-default examples with "NA" values have been dropped.

Specific Data Processing:

Discretization: As decision tree models tend to behave better on categorical data, we examined some numerical data. And for two features: Age_days and Credit_Amount, we conducted discretization and converted them to dummies. Intuitively, they are probably very important in predicting the results and are common targets for discretization in practice. While statistically, those 2 features have long-tail issues, so that we would like to further process.

Hyper-tunning:

Our major target for hyper-tuning of the decision tree is to eliminate the over-fitting issue. So, we picked 2 hyperparameters: 'max_depth' and 'min_samples_split' and did a grid search for cross validation with cv=10.

(Grid search range:

max_depth= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'min_samples_split': [2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, 57, 62, 67, 72, 77, 82, 87, 92, 97])

As the major target of this model is to maximize the profit, we care more about the true positive rate and false positive rate, and AUC more than sole accuracy, when doing the grid search, we set the scoring to be ROC-AUC. And we eventually adopted the best parameters output of max_depth=7 and min_sample_split=72 for model DT_Model (train)

max_depth=8 and min_sample_split=72 for model DT_Model_1 (train_1)

Model Training:

We have built our model on two data sets: train_1_cleaned (train_1_df) and train_cleaned (train_df), and would decide which one to use according to their performance on the same test set. We split the train_1 and train into train and test set. Considering we would need to conduct a cost-benefit analysis to determine an optimal threshold, we further split the train set in train into a sub-train set and a validation set (Exhibit 5).

Figure 5. Split the train set in train into a sub-train set and a validation set

Then we built a decision tree model DT_model_1 on train_1_df and fit test set of train_df into the model and got the result accuracy of only 0.26

```
Accuracy from y_test (test set in train_df): 0.26403847734933816
Confusion Matrix
 [[ 4829 17569]
 [  334  1594]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.22      0.35     22398
           1       0.08      0.83      0.15      1928

    accuracy                           0.26     24326
   macro avg       0.51      0.52      0.25     24326
weighted avg       0.87      0.26      0.33     24326
```

Figure 6. The result table of decision tree model built based on train_1_df

Due to the very poor performance of DT_model_1 in terms of accuracy, we decided to quit the model. Besides, we also noticed that this model has an issue of inadequate data size (Number of examples) considering the number of features.

Our next attempt is directly building a decision tree model based on train_df (the data set from train_cleaned.csv).

Recall that we have previously split 3 parts in train_df: (1) Sub_train set (2)Validation set (3) Test set. We built a second decision tree model, DT_model, on the sub_train set. And then we fit test set of train_df into the model and got a high result accuracy of 0.92 .

```
print("Accuracy from y_test (test set in train_df):",accuracy_score(y_test, y_predict, normalize=True, sample_weigh
print("Confusion Matrix", '\n', confusion_matrix(y_test, y_predict))

print("Classification Report:", '\n', classification_report(y_test, y_predict))
```

```
Accuracy from y_test (test set in train_df): 0.9195099893118474
Confusion Matrix
 [[22331    67]
 [ 1891    37]]
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     22398
           1       0.36      0.02      0.04      1928

    accuracy                           0.92     24326
   macro avg       0.64      0.51      0.50     24326
weighted avg       0.88      0.92      0.88     24326
```

Figure 7. The result table of decision tree model directly built based on train_df

Then we examined performance in terms of the AUC, it's around 0.51.

```
# Get AUC
DTM_auc = auc(DTM_fpr, DTM_tpr)
# Print results
pd.DataFrame({"Model":['Decision Tree'], "AUC":[DTM_auc]})
```

	Model	AUC
0	Decision Tree	0.5081

Figure 8. The final AUC of decision tree model directly built based on train_df

Considering the high accuracy and an acceptable AUC above 0.5, we decided to adopt DT_model.

Threshold adjustment based on cost-benefit analysis

After getting the model (DT_model), we adopt cost and benefit analysis on our validation set to find the optimal decision threshold. The optimal threshold (for '0' prediction) is found to be 0.9. The 'Cost Curve' and 'Cost Comparison' are shown below :

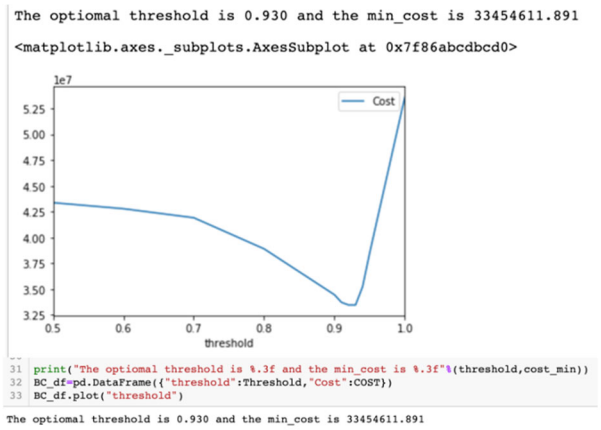


Figure 9. Cost graph with different threshold

We then adjusted the decision threshold of DT_model to 0.93.

		Cost	% Difference
0	Total Cost by majority classifier	43991056	NaN
1	Cost with 0.93 threshold	33454611	-23.951335

Figure 10. Cost comparison table between all predicted by majority and by our 0.9 ideal threshold

The cost incurred by a majority classifier (all predict to be non-default) is about 43.9 million ("Total Cost"). The cost decreased by 23.95% after the prediction with default threshold of 0.93.

Evaluation

We then checked the optimal model with the decision threshold adjusted on the test set of train_df to get the results as follows:

Accuracy: 0.6861 - Cost Reduction: 22.65% reduction compared to the majority classifier

```
[[15502  6896]
 [  739  1189]]
0.6861382882512538
```

Figure 11. The confusion matrix of test set based on DT_model

		Cost	% Difference
0	Total Cost by majority classifier	53042374	NaN
1	Cost with 0.93 threshold	41027806	-22.650887

Figure 12. The reduced cost result of test set based on DT_model

Conclusion

Regarding the 2 decision models we built, DT_model_1 (Nondefault NA dropped) and DT_model, though we eventually decided to adopt DT_model, we still think that the feature ranking of DT_model_1 may still give us some insights of the identification of default.

And for DT_model, though the final model has a relatively okay accuracy (0.6861) and cost reduction (22.65%), the AUC is still unideal, being just slightly above 0.5. It still needs improvement.

4.3. Logistic Model

Data Source: train_cleaned.csv("train"), train_1_cleaned.csv ("train_1")

Hyper-tunning: After searching for related research about hyper parameters in Logistic Model, we found that the hyper parameters that have the highest relationship and biggest influence towards logistic model is the penalty type (whether "l1" or "l2") and the value of C (which determines the strength of penalty). We confine those 2 parameters into the range of penalty = ['l1', 'l2']", and C = [100, 10, 1.0, 0.1, 0.01], then import the library – GridSearchCv to help us try different combinations of hyper parameters one by one. Finally, we got the best result of those values and redefine our model based on them, which are {'C': 100, 'penalty': 'l1'} and {'C': 0.01, 'penalty': 'l1'} respectively in our 2 trials.

Model Training and Threshold Selection

We have built our model mainly based on two data sets: train_1_cleaned (train_1_df) and train_cleaned (train_df), and would decide which one to use according to their performance evaluated on the same test set.

Then we build a Logistic model with the train_1_cleaned set and use this model to fit into all train data (without dropping those non-default items) to see the accuracy. And we found that the accuracy of this model is rather low in the whole train data (Exhibit 5), although it performed well in train_1_cleaned, so our first attempt failed.

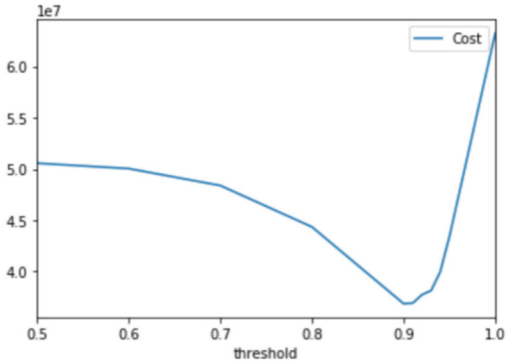
```
print(accuracy_score(y_fc1, pred_testc1, normalize=True, sample_weight=None))
0.22917213760504201
```

Figure 13. The accuracy of model 1 in predicting whole train data

Our next attempt is directly building a model based on the whole train data set that has been processed, which is train_cleaned.csv. We split this data set into traino set and test set, then further split traino set into train set and validation set(which we will use later for cost analysis), then build model based on train set and valuation based on test set. We found that this time the accuracy of our model is pretty high (0.919867), and the AUC (0.500804) also passed 50%, which shows the usefulness of our second trial model.

Threshold Selection:

After getting the model, we adopt cost and benefit analysis on our validation set. The optimal threshold (for '0' prediction) is found to be 0.9. The 'Cost Curve' and 'Cost Comparison' are shown below:



		Cost	% Difference
0	Total Cost by majority classifier	50662719	NaN
1	Cost without threshold	50561611	-0.199571
2	Cost with 0.90 threshold	36880296	-27.058701

Figure 14. Cost graph with different threshold & Cost comparison table between all predicted by majority, predicted without threshold adjustment, and by our 0.9 ideal threshold

The cost of validation set incurred by a majority classifier (all predict to be non-default) is about 50.7 million ("Total Cost"). The cost decreased by 0.2% in our model without threshold adjustment (threshold being 0.5) and 27.06% after the prediction with the 0.9 threshold.

Evaluation

We have applied the model and the optimal threshold on the test set and get the results as follows:

```
Confusion Matrix:
[[ 1087   860]
[ 5218 17207]]

Accuracy is  0.7506154603643526
```

Figure 15. The confusion matrix of test set based on logistic model

		Cost	% Difference
0	Total Cost	62948509	NaN
1	Cost with 0.9 threshold	46208679	-26.592894

Figure 16. The reduced cost result of test set based on logistic model

The cost is reduced by about 26.6% in test set after applying the model.

Conclusion

Although our Logistic model has pretty high accuracy after threshold adjustment (0.7507) and helps reduce costs by about 26.6%, but our AUC (0.500804) is still not so satisfactory as we expected. So, we kept trying other models to find more precise results.

4.4. KNN Model

Data Source: train_cleaned.csv("train"), train_1_cleaned.csv ("train_1")

Hyper-tunning: In the model, we have used 2 parameters [neighbor count (k), distance type (p)] to assign weights to neighbors by distance. As we care more about positive labels, we use AUC as the performance index in the tunning stage. For the train_1 data, 15 combinations of the two parameters are used to tune the optimal parameters with cross-validation. For train data, 11 combinations are used to tune the model. Among the combinations, we figure out that the relative AUC maximum is achieved when k = 33/35 and p =1 for train_1; and when k=71 and p=1 for train.

Model Training and Threshold Selection

We have trained 2 models with 2 datasets. The model performance is shown below:

With train_1_cleaned:

Model	AUC	Accuracy
KNN	0.931162	0.852536

```
Confusion Matrix:
[[ 586 1325]
[ 120 7768]]
```

With train_cleaned:

Model	AUC	Accuracy
KNN with train_cleaned	0.694756	0.923769

```
Confusion Matrix:
[[71725     0]
[ 5945   317]]
```

The AUC, Confusion Matrix & Accuracy of KNN Model of train_1_cleaned and train_cleaned

However, considering the model nature of KNN, we decided to fit our final model with 'train' (k =71 and p =1) which shares the similar label compositions as the real business data. The model performance on validation set by default threshold is (note that Positive :1, Negative:0):

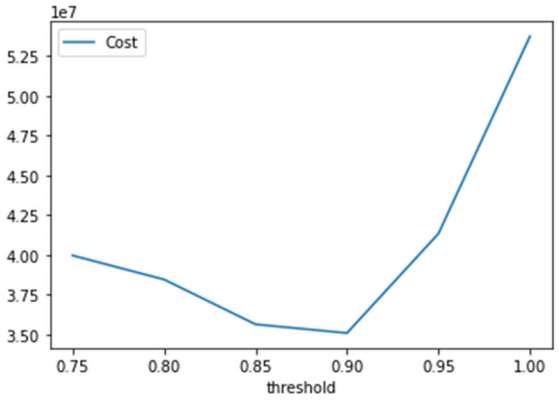
```
Confusion Matrix:
[[   97  1497]
[    0 17903]]
```

	Model	AUC	Accuracy
0	KNN-Validation Set	0.694575	0.923219

(fit with train_cleaned)

The AUC, Confusion Matrix & Accuracy of KNN Model of train_cleaned

After getting the model, we adopt cost and benefit analysis on the validation set. The optimal threshold (for '0' prediction) is found to be 0.9. The 'Cost Curve' and 'Cost Comparison' are shown below:



	Cost	% Difference
Total Cost	43853498	NaN
Cost with Default Prediction	41184872	-6.085321
Cost with 0.9 threshold	35086800	-14.806582

Cost graph with different threshold & Cost comparison table between all predicted by majority, predicted without threshold adjustment, and by our 0.9 ideal threshold

The cost incurred by a majority classifier (all predict to be non-default) is about 43.9 million ("Total Cost"). The cost decreased by 6.09% after the prediction with default threshold, and further decreased by 14.81% with 0.9 threshold.

Evaluation

We have applied the model and the optimal threshold on test set and get the results as follows:

Model	AUC
KNN-Test Set	0.704615

	Cost	% Difference
Total Cost	54720582	NaN
Cost with 0.9 threshold	41991212	-23.26249

The AUC & reduced cost result of test set based on KNN model

The cost is reduced by about 23.26% after applying the model. KNN model can fit the data well while also have limitations. (e.g. the actual business data volume and feature size are large)

4.5. Summary of Performance

We've tried 3 types of different models, namely, Decision tree model, Logistic model and KNN model, and we got 3 different evaluations based on them.

Firstly, the decision tree model, after doing the hyper-parameter tuning, the AUC of our finalized model is 0.525804 based on our separate test set. After doing the cost analysis, we used the threshold of 0.93 and under this threshold our cost can be reduced by 22.98% in our separate test set compared to majority classifier (all predict to be non-default).

Secondly, the Logistic model, after doing the hyper-parameter tuning, the AUC of our finalized model is 0.500804 based on our separate test set. After doing the cost analysis, we used the threshold of 0.9 and under this threshold our cost can be reduced by 26.6% in our separate test set compared to majority classifier (all predict to be non-default).

Lastly, the KNN model, after doing the tuning for the values of k and p, the AUC of our finalized model is 0.704615 based on our separate test set. After doing the cost analysis, we used the threshold of 0.9 and under this threshold our cost can be reduced by 23.26% in our separate test set compared to majority classifier (all predict to be non-default).

In conclusion, we can find that KNN did the best in predicting default situations under all thresholds overall with the biggest AUC of 0.704615 on test set. But it didn't do that well in cost analysis under the specific threshold of 0.9, Logistic model did best in reducing cost with ideal threshold, which lead to the biggest decrease among all 3 models of 26.6%, but the AUC of it is not as high.

So, we can further ensemble them together to see whether we can achieve an even better model doing well both in prediction and reducing cost.

5. Ensemble Model

We have incorporated the 3 models elaborated above and tried 2 other new models (Random Forest and Adaptive Boost) to collectively predict 'default'. Results show that random forest works well on the data while adaptive boost does not. Thus, we discard adaptive boost model and use Majority Voting Classifier to build the ensemble model in 2 versions. Next, we have also explored how it performs with and without the prediction threshold (0.9) on the test set. The test results for prediction with threshold 0.9 (shown below) are much better than the default prediction.

5.1. Version 1: KNN, Logistic, Decision Tree, Random Forest

Prediction 1 (Majority Prediction)

	Cost	% Difference
Total Cost	53922745	NaN
Cost2 with 0.9 threshold	34524563	-35.974025

Prediction 2 ('1' if > = 0.5 models predict '1')

	Cost	% Difference
Total Cost	53922745	NaN
Cost with 0.9 threshold	37489313	-30.475882

Cost comparison table between 2 prediction methods by our 0.9 ideal threshold for fusion model version 1

We have included all models at first and tried two types of prediction (> or > = 0.5). Results showed that prediction 2 works better and can reduce costs by 35.98%.

5.2. Version 2: KNN, Logistic, Random Forest

Prediction 1(Majority Prediction)

		Cost	% Difference
0	Total Cost	54885651	NaN
1	Cost with 0.9 threshold	33530143	-38.909091

Prediction 2 (0 if all 3 models predict 0)

		Cost	% Difference
0	Total Cost	54885651	NaN
1	Cost2 with 0.9 threshold	41503006	-24.382775

Cost comparison table between 2 prediction methods by our 0.9 ideal threshold for fusion model version 2

We exclude decision tree in this version since it has a relatively poor performance separately. As we found that separate models fail to predict more 'default' than 'non-default'. We try to recognize all the default predictions (pred 2). After comparing both, we found that majority predictions work better and can reduce costs by 38.9%.

5.3. Conclusion

Ensemble model can greatly improve the prediction quality and boost the cost reduction compared with a single model. Among all the combinations we've tested, ensemble model (KNN, logistic and random forest) with majority prediction and 0.9 threshold performs the best.

6. Business Application

Data Collection Advice

i. Logistic:

We have the table of coefficients and intercept as shown:

```
[[-2.30617343e-01  0.00000000e+00  0.00000000e+00  8.55889555e-02
  -3.20115386e+00  7.79598521e-02 -9.40807280e-02 -2.21998723e-01
  -1.89639540e-02 -1.58445559e+00 -3.89683705e-02  0.00000000e+00
  -5.42724342e-02  8.27630864e-02  3.89390578e-02 -5.10227935e-02
  -7.61981763e-02 -1.23037106e-02  2.31498667e-02  8.98384323e-02
  -8.14223097e-02  3.25455440e-02 -1.71031412e+00 -4.49062255e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  -4.61336922e-02  0.00000000e+00 -9.78094349e-02 -1.11739811e-01
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  1.14269831e-01  3.81124805e-01  0.00000000e+00
   3.61365994e-01 -8.71746344e-02  0.00000000e+00  0.00000000e+00
  -3.01498568e-01  0.00000000e+00 -3.42798569e-01 -1.95809806e-02
   1.16078641e-01  0.00000000e+00  0.00000000e+00  0.00000000e+00
   7.18267466e-02 -1.55290884e-01  0.00000000e+00  0.00000000e+00
   7.64520329e-02  0.00000000e+00  2.69998403e-01  0.00000000e+00
   0.00000000e+00  0.00000000e+00 -2.96290226e-02  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -2.08049066e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -2.57443025e-02 -1.83103280e-01  0.00000000e+00
   0.00000000e+00 -1.21955080e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00 -1.29752452e-01  0.00000000e+00
  -2.09110859e-01  1.31983602e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -2.91633658e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  1.03966181e-02  0.00000000e+00
  -7.12889927e-04  2.36818319e-01  1.18272935e-01  0.00000000e+00
   0.00000000e+00  0.00000000e+00  1.42105843e-01 -1.79343360e-01
   9.79005934e-02  0.00000000e+00 -1.47583716e-01  0.00000000e+00
   7.04269806e-02  2.00562588e-01 -2.78117795e-03  0.00000000e+00
  -4.03971165e-02  0.00000000e+00  2.99700920e-02  0.00000000e+00
   1.63490592e-01 -8.33307814e-03  0.00000000e+00]] [-3.34552044]
```

Table of coefficient and intercepts of our logistic model

As the table shown, we can find that the features with 5-top highest absolute value of coefficient (which means strong impact of the feature on the probability of prediction outcome being 1, which is default) are 'Mobile_Tag'(Mobile Number provided by Client, 1 means Yes and 0 means No), 'Score_Source_2'(the second trust score from third party), 'Score_Source_1_z'(Z score of the first trust score from third party), 'Score_Source_3_z'(Z score of the third trust score from third party), 'Client_Education_Junior secondary'(whether clients have finished the Junior secondary education).

ii. Decision Tree:

Our top 10 features with the most importance are shown:

```
{'Score_Source_3_z': 0.3559637452241733,
 'Score_Source_2': 0.3264447016307177,
 'Score_Source_1_z': 0.0886359812236947,
 'Employed_Days_z': 0.04300526442451751,
 'Client_Education_Secondary': 0.029871648409456077,
 'Phone_Change_z': 0.023202476996209306,
 'ID_Days_z': 0.01924632672444942,
 'Client_Gender_Male': 0.016591377504722404,
 'Loan_Annuity_z': 0.01051241139655702,
 'Registration_Days_z': 0.009728691126539513}
```

top 10 features with the most importance of our decision tree model

Surprisingly, the first 5 nodes in decision tree are 'Score_Source_3_z', 'Score_Source_2', 'Score_Source_1_z', 'Employed_Days_z' (Z score of Days before the application, the client started earning), 'Client_Education_Secondary', which show large similarity with those in our Logistic model.

So, we can conclude that the features of Score 1, 2, 3 and whether clients have finished junior secondary education are key features in predicting default states. And we need to pay special attention to those features when collecting data, setting those features as required fields when collecting the information of clients during their application of our car loan.

Business Strategy

We suggest the financial institution to reject car loan applications which predict to be 'Default'. For the car loan application with 90%-95% probability not to default, FI can consider set limits on the loan amount.

After the application stage, FI can consider set a 6 -12-month time window to observe the customer behavior and collect data such as 'Day Past Due' to adjust the strategy accordingly.

7. Reflections

a) Data Processing

i). The proportion of default examples:

As previously discussed in this report, we constructed 2 datasets, one with all non-default NA examples dropped (train_1) while the other dataset keeps all examples (train). In our original expectation, we thought models built on train_1 would capture the essential characteristics of default cases better, which we care about the most for profit reasons. However, though models built on train_1 turn to have higher AUC in general, they perform too poorly in terms of accuracy.

This may be due to the train_1 dataset having a very different proportion of default nom-default from reality, so there would be accuracy pitfalls when we try to use the model to predict some real data.

ii). Possible interaction between features unknown:

There may exist some interactional relationships between features, which we currently are unaware of. Exploration and utilization of those relationships may be beneficial to model building.

b) KNN model in practice

Though individually, the KKN model performed the best among all the models in terms of AUC. However, it may not be very practical for financial institutions to use in real life, as it's time-consuming and takes up too much computational power when the volume of data to be processed is huge.

c) Underlying logic and reasons of feature importance

We can only have importance scores of features in the decision tree and logistic models, but we are unaware of the underlying logic and reason of a feature being important. For example, 'Mobile_Tag' (Mobile Number provided by Client, 1 means Yes and 0 means No) feature ranks top among

logistic model features. However, the real reason for 'Mobile_Tag' having high predictivity could possibly be that whether a mobile phone number can be provided may reflect the stability and financial state of a customer. Then we may also try to collect more information about the stability of a customer applying for a loan. If we stretch further to those underlying logic and reasons; we may collect data more effectively to achieve better prediction results.

d) Sensitivity to change of cost analysis

Considering the practice of selecting an optimal decision threshold based on the cost analysis, the sensitivity to change of the cost analysis should be considered. Changes in parameters like average exposure, commission rate, etc., may affect the result of the analysis, changing the optimal model.

8. Summary

According to the characteristics of our data (large size with a great number of features; in-balanced in general), we applied 2 different data processing methods (Dropping nondefault NAs or keeping them) and got 2 datasets.

And with our existing skill sets, we built the following individual models. And selected 3 models to be adopted based on their performance in terms of accuracy, confusion metrix, and AUC.

Model Type	Data Set Used	Adopted
Decision Tree Model	train	Yes
	train_1	No
Logistic Model	train	Yes
	train_1	No
KNN Model	train	Yes
	train_1	No

Keeping our target of making predictions that maximize profits in mind, we did cost analysis and adjusted the thresholds to be optimal in each adopted model.

According to the feature importance ranking of decision tree model and logistic model, we can conclude that the features of Score 1, 2, 3 and whether clients have finished junior secondary education have the highest predictivity in the loan default results. And we suggest the institution outsources and pays more attention to the above 4 features in terms of data collection.

And we stretched further to incorporate the 3 models and tried 2 other new models (Random Forest and Adaptive oost) to build ensemble model with different combinations. And we also tried out 2 rules, which are 'Predict default when majority predicts default' and 'Predict default when all predict default'. Eventually, we decide that ensemble model (KNN, logistic and random forest) with majority prediction and 0.9 threshold performs the best.

References

[1] Wang P, Zheng H , Chen D , et al. Exploring the critical factors influencing online lending intentions[J]. Financial Innovation, 2015, 1(1):1-11.

[2] Galindo J , Tamayo P . Credit Risk Assessment Using Statistical and Machine Learning[J]. Computational Economics, 2000.

[3] Malekipirbazari M , Aksakalli V . Risk assessment in social lending via random forests[J]. Expert Systems with Applications, 2015, 42(10):4621-4631.

[4] Li Y, Chen W. Entropy method of constructing a combined model for improving loan default prediction: A case study in China[J]. Journal of the Operational Research Society, 2019(4):1-11.

[5] Zhou, J., Li, W., Wang, J., Ding, S., Xia, C., 2019. Default prediction in P2P lending from high-dimensional data based on machine learning. Physica A Statistical Mechanics and its Applications 534, 122370. doi: 10.1016/j.physa.2019.122370.

[6] Zurada J. Data mining techniques in predicting default rates on customer loans[M]// Databases and Information Systems II. Springer Netherlands, 2002.

[7] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.