Deen Huang

Capstone Final Report

Professor: Amir Jafari, PhD

December 2, 2018

Prediction of Default of Credit Card Clients

Introduction

Nowadays, the credit card plays an important role of changing people's life. Prior to the development of credit card, people have to carry a lot of cash in order to do business trade. However, accompanying with the development of credit card, banks are facing with risk based on clients' credit card default. In order to issue a credit card to the potential customers, banks would check the credit card default to make sure whether clients meet the legal obligations of a credit. There is a verity of factors would affect approval of credit cards, such like paying credit card billing on time, ability of paying the full balance. It is important to examine the credit card billing information to avoid high risk customers. In my project, I am focusing on the problems: how does the probability of default payment vary by categories of different demographic variables? Which variables are the strongest predictors of default payment? How do we predict the credit card clients may lead to default of credit card? Through solving those problems, banks are able to earn profits effectively and prevent high risk borrowers.

This project is created to classify and predict the credible credit card clients based on the classic and advanced machine learning methods. The idea is to use the "Default of Credit Card Clients Dataset" from *UCI Machine Learning*, and applied it in different machine learning models and compared the performance of these two models. This dataset was collected by I-Cheng Yeh, and it contains the information on default payment, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. In this dataset, there are 25 variables and 30,000 observations. My goal is supposed to predict the credible clients and figure out which model would predict the results more accurate.

Methods and ideas

My project serves as a starting point in understanding the probability of default payment vary by categories of different demographic variables. I cleaned the dataset first, and checked if

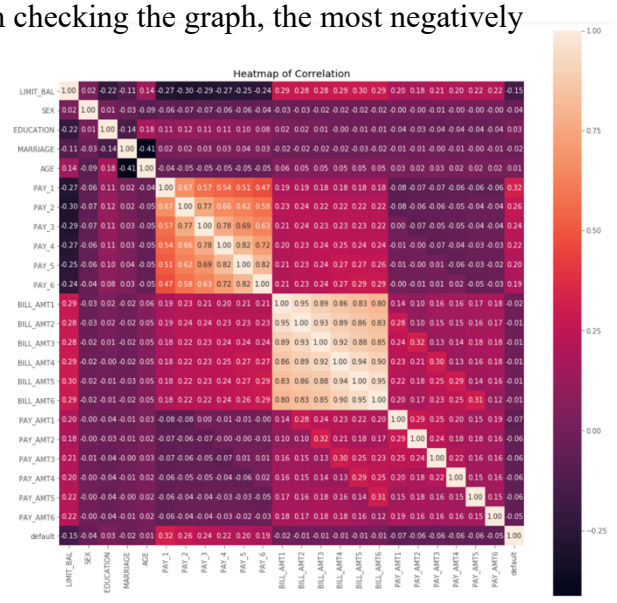
my data set exists any missing values. Through observing the summary of data frame, I decided to change the column name “default.payment.next.month” to “default”, which would be more easier to handle with the target variable. In addition, I would like to drop the irrelevant variable “ID”. After checking the balance of my data set, I also rescaled the features to a standard normal distribution with a mean of 0 and standard deviation of 1.

Regarding of modeling methods, I used both classical machine learning and deep learning algorithms. The classical machine learning algorithm I utilized was Gaussian Naïve Bayes. I also calculated accuracy, precision, recall, F1 score and ROC for my model. Furthermore, I plotted the confusion matrix and ROC curve to figure out the performance of my Gaussian Naïve Bayes model. For the deep learning algorithm, I used Neural Network associated with Tensorflow. I built a 5 hidden layers’ Neural Network, and optimized my model via AdamOptimizer method. After starting my training circle, I calculated the train accuracy, cost, validation accuracy and validation cost for every 10 epochs. Moreover, the python packages I used in my project were pandas, numpy, seaborn, tensorflow, matplotlib and skelearn.

Experimental Results and Analysis

In this project, I noticed that there were no missing values in the data set and every variable is numeric, which could be much easier for data cleaning. I plotted the heatmap of correlation to find out the correlation between feature variables and target variable. Through observing the heatmap, the feature variable that mostly correlated with target variable is repayment status (PAY_1, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6). And the repayment status in September, 2015 is considered as the most correlated indicator among these six variables. Normally, the amount of given credits (LIMIT_BAL) should be considered as a significant influence for the credit card default. However, upon checking the graph, the most negatively correlated feature variable is LIMIT_BAL.

figure.1



After observing the heatmap of correlation, I plotted the bar chart of amount of default observations and non-default observations to check balance. It is obviously that the target variable is imbalanced. So I tried to find the ratio of default and non-default, such that I could split my train data more efficiently. By calculating the of ratio, I found a default has been observed as around 22% based on the total observations.

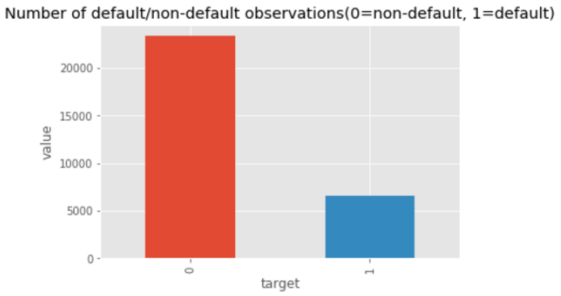


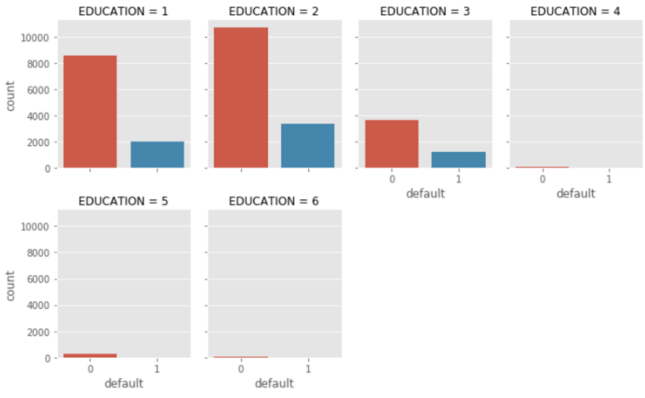
figure.2

```
0    23364
1     6636
Name: default, dtype: int64

default Ratio : 0.2212
```

figure.3

Visualizing and comparing the different feature variables to my target variable "default" could help me realize the basic relation among my variables more impressively. I compared the variable "EDUCATION" to defaultable observations by looking the graphs, the people who received higher education seems have more non-default payments but not a strong indicator. Then I did same procedure to compare the "MARRIAGE" with defaultable observations, it is clearly that the people who are single have more non-default payments. My hypothesis of the reason caused this situation was that people who got married would pay more expenses. And they have to devote all of their money into family expenses, such like education fee for their offspring. By comparing the variable "SEX" to defaultable observations, I found that females have overall more non-default payments compared to males, but it only has small effect on default payment.



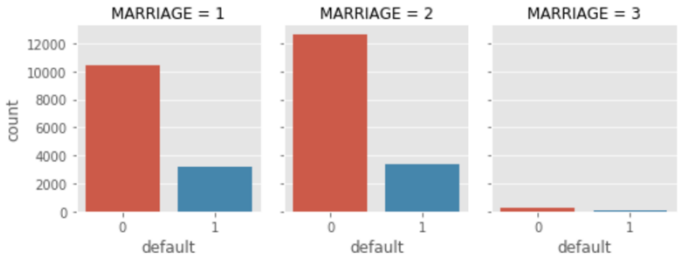


figure.4

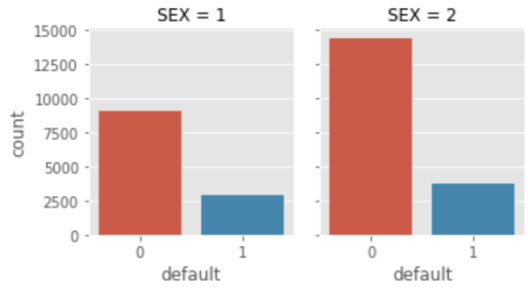


figure.5

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	GNB	0.554	0.295825	0.770239	0.427471	0.632302

figure.6

figure.7

After examining the relation between variables, I began to build two of my models. The first classic machine learning model I would like to put forward is Gaussian Naïve Bayes. The first step I would like to do is setting the data frames for both features and target variable. In order to train the model more efficiently and prevent overfitting, I split my data into 80% of training data and 20% of testing data. Then I scaled the features to a standard normal distribution with a mean of 0 and standard deviation of 1 due to the imbalanced data. After well trained and tested data, I used Gaussian Naïve Bayes classifier to fit the best algorithm to the data. By applying machine learning model, I obtained a table including accuracy, precision, recall, F1 score and ROC. Indeed, the accuracy this model got was 55.4%, which indicated that I should change to a more complex model to get a higher accuracy. Furthermore, the precision was low as 0.296 as well. The confusion matrix also implied that the non-default of test data had a very low accuracy. In addition, I graphed a ROC curve to figure out the performance of model. However, it's hard to conclude from the ROC of this Gaussian Naïve Bayes model. Therefore, a more advanced and complex model was regarded as an indispensable way to achieve a higher accuracy and better model performance.

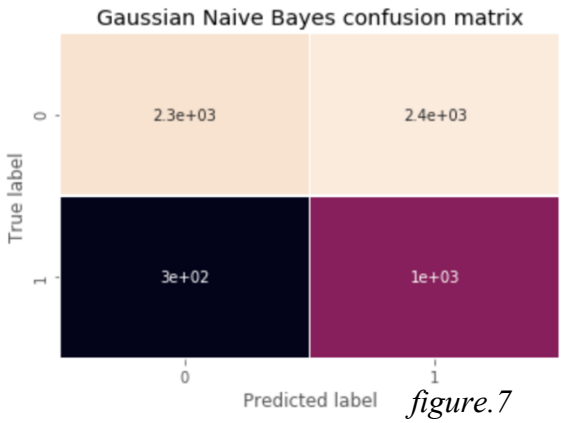


figure.7

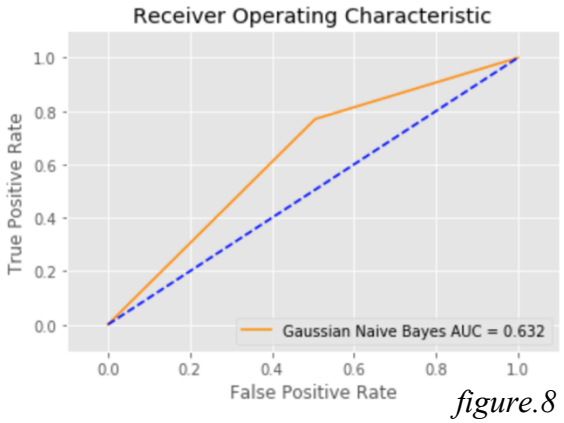


figure.8

The Deep Learning model I used was Neural Network associated with Tensorflow. In order to deal with the imbalanced observation for target variable, I created a new class for non-default observations. After finishing creating a new data frame that only included default and non-default observations, I set the features training data to 80% of default observations, and then added another 80% of non-default observations to it. Later, I set features test data with all the remaining observations and shuffled both of them into a random order. For the target data training and test, I added my target variable to y_train, y_test. Then I dropped the target variable from my features train and test data set. Same as the similar process I completed in the classic machine learning model, I also scaled the features to a standard normal distribution with a mean of 0 and standard deviation of 1. Furthermore, I also split the test data into validation and testing data set in this case. Getting along with preparing the training, validation and testing data set, I started to build a five hidden layers Neural Network based on Tensorflow. First, I set layers neurons and learning rate as a starting point. Second, I declared the basic structure of the data and define two functions for weight variable and bias variable. With the help of those procedures, I could easily apply my five hidden layers and one output layer using the functions. Then I defined several parameters such as training epoch, training dropout, batch size and so on. After setting the cost function, I could optimize my model via AdamOptimizer. The correct prediction function could correct the prediction if the values from softmax were same as target value. Then I initialized my training circles and displayed the logs every ten epochs. The final optimized results I got were both accuracy and cost for training, validation and testing data sets. After that, I plotted the accuracy and cost function graph for those data sets in separated trends lines.

```
Epoch: 0 Acc = 0.779 Cost = 1.059 Valid_Acc = 0.787 Valid_Cost =  1.040 test_Acc = 0.771 test_Cost =  1.077
Epoch: 10 Acc = 0.813 Cost = 0.907 Valid_Acc = 0.812 Valid_Cost =  0.901 test_Acc = 0.802 test_Cost =  0.936
Epoch: 20 Acc = 0.813 Cost = 0.901 Valid_Acc = 0.813 Valid_Cost =  0.896 test_Acc = 0.804 test_Cost =  0.935
Epoch: 30 Acc = 0.814 Cost = 0.901 Valid_Acc = 0.814 Valid_Cost =  0.897 test_Acc = 0.803 test_Cost =  0.938
Epoch: 40 Acc = 0.814 Cost = 0.898 Valid_Acc = 0.814 Valid_Cost =  0.894 test_Acc = 0.804 test_Cost =  0.933
Epoch: 50 Acc = 0.816 Cost = 0.897 Valid_Acc = 0.818 Valid_Cost =  0.890 test_Acc = 0.803 test_Cost =  0.936
Epoch: 60 Acc = 0.818 Cost = 0.892 Valid_Acc = 0.822 Valid_Cost =  0.887 test_Acc = 0.809 test_Cost =  0.929
Epoch: 70 Acc = 0.821 Cost = 0.888 Valid_Acc = 0.823 Valid_Cost =  0.882 test_Acc = 0.811 test_Cost =  0.922
Epoch: 80 Acc = 0.819 Cost = 0.885 Valid_Acc = 0.825 Valid_Cost =  0.881 test_Acc = 0.806 test_Cost =  0.924
Epoch: 90 Acc = 0.824 Cost = 0.863 Valid_Acc = 0.829 Valid_Cost =  0.857 test_Acc = 0.811 test_Cost =  0.901
Epoch: 100 Acc = 0.824 Cost = 0.852 Valid_Acc = 0.829 Valid_Cost =  0.848 test_Acc = 0.810 test_Cost =  0.892

Optimization Finished!
```

figure.9

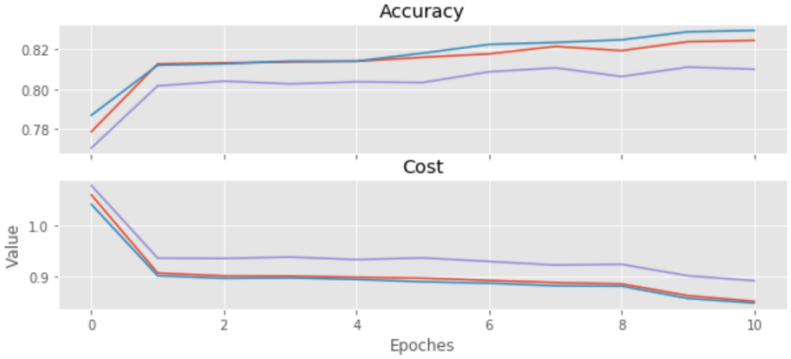


figure.10

Conclusion

Through comparing the performance of both classic machine learning model and deep learning model, I could easily figure out that the deep learning model – "Neural Network" provided a higher accuracy and much better performance. The prediction accuracy I gained could reach to 82%, which was much better compared to the accuracy of 55.4% that I received from Gaussian Naïve Bayes model. In addition, through watching the accuracy and cost function graph in Neural Network model, it is evidently that cost becomes lower and accuracy becomes higher when the epochs increased. There were not any extreme demographic variables that indicated the default payment. For the further process, I would utilize more advanced machine learning models to achieve higher accuracy such as Adaboost, Gradient Boosting, SGD and so on.

Reference

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Appendix

- ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years
- PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2: Repayment status in August, 2005 (scale same as above)
- PAY_3: Repayment status in July, 2005 (scale same as above)
- PAY_4: Repayment status in June, 2005 (scale same as above)
- PAY_5: Repayment status in May, 2005 (scale same as above)
- PAY_6: Repayment status in April, 2005 (scale same as above)
- BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)
- default.payment.next.month: Default payment (1=yes, 0=no)