Credit Default Modeling Using Machine Learning Algorithms

Ozodbek Ozodov

August 23, 2022

Credit risk and default is one of the most essential topics in the lending institutions nowadays. As types of products in the loan market increase, various factors are becoming big part of the discussion while determining certain factors and considering them in the prediction models. I will consider some of the ML models, specifically logistic regression from generalized linear models, Random forest algorith from tree-based methods and K-Nearest-Neighbor classifier and compare them in terms of predictive power and computational complexity. This project uses *Lichman, M. (2013). UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science* in the first part, and simulation study in the last section.

1 Introduction

This project aims to compare the Machine Learning algorithms for measuring credit card default risk. I will directly use the data *UCICreditCard*. Further processing is carried out, and I will include those in the code scripts. The main idea of the project is to implement several Machine Learning algorithms in credit default context, and compare their predictive power, error rates and reliability in general. Some important measurements around Confusion Matrix will be discussed. Also, variable importance is considered across models to find the best fitting model and to decrease computational costs of the model implementation. Project also includes a simulation part where hypothetical data is generated and used for building models.

2 Data description

The dataset contains information on default payments in credit cards, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. Unlike conventional characteristics such as gender, marital status and age, there are four variables that needs to be explained:
1. LIMIT BAL - Amount of given credit in NT dollars (includes individual and family/supplementary credit
2. PAY N: Repayment status in respective month (1 - September, 2 - October...) of 2005. (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 9=payment delay for nine months and above). Later this variables will be unified taking their sum as *payment status*. The higher the *payment status* the longer the client has not been repaying the debt.
3. BILL AMT - Amount of bills for respective months (1 - September, 2 - October...) of 2005. These all six variables will be unified taking their sum as *bills*.
4. default.payment.next.month - Default payment (1=yes, 0=no)
The data set contains 25 variables and 30,000 observations. Detailed explanation from main source on each variable will be attached.

3 Methodology

Different types of Machine Learning techniques are used in the literatures. I decided to choose three main approaches to analyze the credit risk.

A. Logistic regression.
Logistic regression is a generalized linear model whose main use is to estimate the probability that a binary response occurs based on one or more dependent variables. Given the main goal of the study is to predict the credit card holder defaults or not, Logistic regression is the benchmark model for this classification problem. Instead of conventional straight line or hyperline, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.
The binary logistic model:

$$\mathrm{Y} = \begin{cases} 1 & \text{If the customer defaults} \\ 0 & \text{otherwise} \end{cases}$$

With conditional probabilities $P(\mathrm{Y}_i = 1|x_i) = \pi_i$ and $P(\mathrm{Y}_i = 0|x_i) = 1 - \pi_i$ where x_i is a vector of covariates associated with the customer ı. The conditional expectation is given by:
$E[Y_i|x_i] = \pi_i$
and this is associated to a linear predictor via the logit function, i.e.

$\text{logit } \pi_i = \log{(\pi_i)}/(1 - \pi_i) =_i \beta = \eta$

B. K-Nearest Neighbor (KNN) Algoritm.

KNN is the non-parametric method used for classification and regression. It involbes a training set of both default and non-default customer data. The standard case of this supervised learning algorithm has numerous advantages for the payment default case. It can easily learn the binary outcomes from the training set with labels given in advance. Also, the KNN is not built on strong assumptions to explicitly rely on. On the other hand, this model is computationally expensive in the case of many dependent variables.

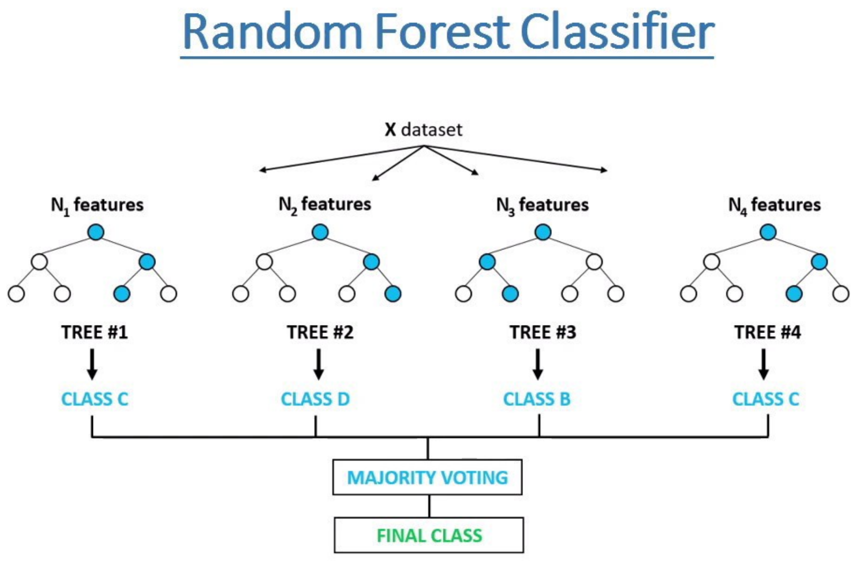
To determine the nearest neighbor, Euclidian Distance metrics is used in this project.

$$\text{d(p,q)}=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$

The number of neighbors K is set to be equal to 5, 10 and 20 and compared. (??) KNN algorithm essentially reflects plurality voting between the K most close instances of a given new observation - a test observation in our case - in the classification settings.

C. Random Forest algorithm.

Random Forest is an algorithm used for both classification and regression problems. Again, I will use it within the scope of the classification algorithm. Random Forest is a supervised, ensemble-based machine learning algorithm that aggregates the number of decision trees to a final tree. In the classification case, the response variable in each iterated group of trees are selected upon the majority responses comprise the classification stage. In the training stage, random forests apply the general technique known as bagging - a bootstrap aggregation - to individual trees in the ensemble. Bagging repeatedly selects a random sample with replacement from the training set and fits trees to these samples. Each tree is grown without any pruning. This algorithm generally has high performance, at the same time this model is computationally demanding.



4 Implementation and results

4.1 Logistic regression scenario.

4.1.1 Conventional training and test splits

Logistic regression of this predictive model looks as follows:

$$P(x)= \frac{1}{1+e^{\beta_0+\beta X}}$$

Dependent variables for this regression equation are:

bills - sum of the last 6 month bills
payment status - number of months the payments delayed
duly - dummy variable with 1 if the penultimate monthly payment is paid and 0 otherwise
married - dummy variable with 1 if married and 0 if single, divorced and others.
male - the gender of the credit card holder. 1 if male, 0 otherwise.
age - age of the credit card holder, an integer variable.

After conventionally splitting the training and test data sets equally in size, logistic regression show that duly payment is not statistically significant (p=0.13), while other variables are found to be significant at 1 percent, except the age with 5 percent significance. Also, while experimental running settings, university education of the credit card holders is also found to be statistically not significant. Using the training model above, I predict the default payments in the test set. Given logistic regression predicts continuous values from 0 to 1, the model needs an optimal threshold value for classifying these continuous values into 0 and 1. In **caret** package, there is a command *optimalCutoff* that finds the optimal cutoff point between continuous values and factor values with minimum error. Using this command, I found the point (0.366) to be optimal in this case. Confusion matrix for the predicted values versus actual values looks as following:

		Reference		
		0	1	Total
Prediction	0	11025	617	11642
	1	2320	1038	3358
	Total	13345	1655	15000

4.1.2 Logistic regression with 10-fold Cross-validation

While building a model using the data with 30000 observations and more than 20 variables, outliers can create additional uncertainty. In this setting, I used cross validated training set to build a logistic regression model to avoid this problem.
Generally, cross-validation works as following:

1. Shuffle the data set randomly.
2. Split the data set into k groups.
3. For each unique group:
   (a) 1. Take the group as a hold out or test data set.
   (b) 2. Take the remaining groups as a training data set
   (c) 3. Fit a model on the training set and evaluate it on the test set.
4. Summarize the skill of the model using the sample of model evaluation scores.

In caret package there is a built-in function train with a functionality of cross-validation. The code is as following:

```
cv_train_log <- train(as.factor(default.payment.next.month) ~ bills +
                          payment_status +  duly + married + male + AGE , data=data,
                      method='glm',
                      trControl=trainControl(method='cv', number=10))
```

10-fold Cross-validated model performs as following:

		Reference		
		0	1	Total
Prediction	0	11472	250	11722
	1	2766	512	3278
	Total	14238	762	15000

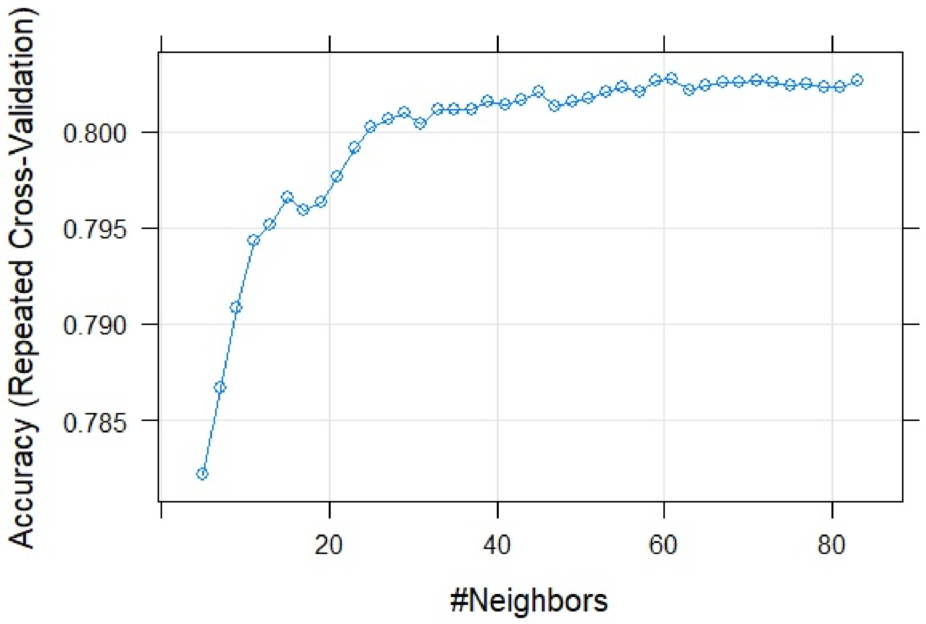
10-fold Cross-validation actually decreased the model accuracy, also other important measures such as sensitivity and specificity. In practice models that are trained with cross-validated resampling performs reliably compared to the conventional splitting. In the first scenario, it is perhaps not the most representative data randomly selected for training, therefore without taking the risk of having volatility, I take cross-validated model as the main one due, despite its lower measurements.

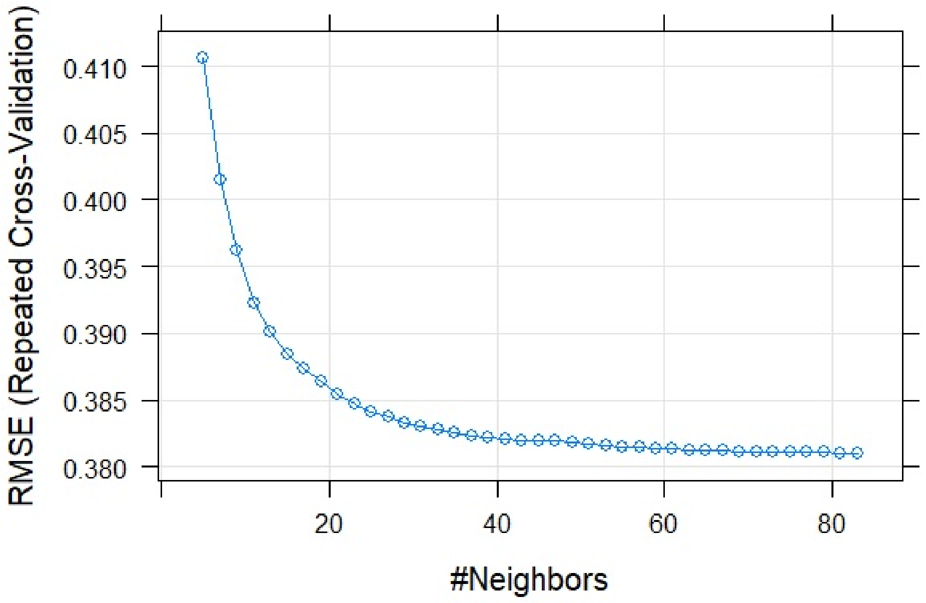
4.2 K-Nearest Neighbors classification scenario

Using caret package in R, it is generally easy to build, run, validate and use the KNN classifier. In the first setting, only three variables - Limited balance, bills and payment status are included in the classifier.

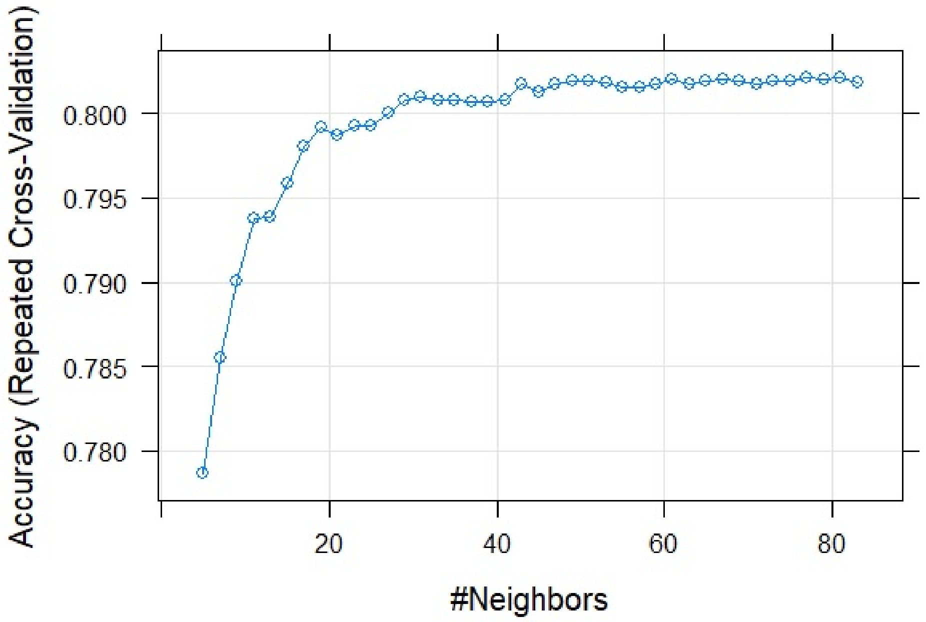
```
fit_knn <- train(as.factor(default.payment.next.month)~ bills+
                   LIMIT_BAL +
                   payment_status,
                 data=train_knn,
                 method="knn",
                 tuneLength=20,
                 trControl=trControl,
                 preProc=c('center', 'scale'))
```

This setting shows that KNN algorithm has 80.2% accuracy with K=43, and reaches stable rate as K increases showing 80.2% consistently.





In the second setting, I added payment status and bill amount for the last month, marriage status and university education dummy on top of previous dependent variables. It can be seen that these variables again has almost no effect on prediction accuracy.



Confusion matrixes for the two KNN models are quite similar:

		Reference		
		0	1	Total
Prediction	0	11036	686	11722
	1	2253	1025	3278
	Total	13289	1711	15000

Implementing this model to the all 30,000 observations yields 80.6% accuracy, 1261 incorrectly predicted non-defaults which actually defaulted.
Confusion matrix for second case (more variables):

		Reference		
		0	1	Total
Prediction	0	11088	540	11628
	1	2483	840	3363
	Total	13571	1420	14991

As it can be seen, newly added variables have no positive effect, yet increases incorrectly predicted defaults, compensating with correctly predicted non-defaults keeping the general model accuracy at the same rate.

4.2.1 Variable importance

From computational point of view, more dependent variables and higher tuning parameters require more computational complexity, time and power. Therefore, in the KNN case, and later in the Random Forest model, it's important to distinguish the importance of the variables, and optimize the models to decrease the computational costs of the models, and yet obtain similar results with few variables. In caret package there is a function varImp(model) that measures the importance of variables in the model built on. In the KNN setting, it is especially important since the algorithm deals with the distance a few times for each observation in the multidimensional setting.
Using this command, I found the variable importance as such:
1. Payment status - 100.00
2. Limited balance - 30.10
3. Payment amount of the 6th month - 4.07
4. University education - 0.89
5. Bills - 0.27
6. Bill amount of the 6th month - 0.02
7. Marriage status - 0.00

Following this results, in further models in the same context, we can only consider first 4 variables, and get the model with generally same results. It is important not only from computational point of view, but also prevents the model from having extra noise from other - least important variables.

4.3 Random Forest model

I used randomForest package in R to build this model. Given random forest model can be used for both classification and regression settings, it is important to show that we want to run exactly the classification model. Therefore, independent variable default has to be converted into factor variable rather than integer.
A random forest classifier is:

```
fit_rf <- randomForest(default.payment.next.month ~
                        bills +
                        LIMIT_BAL +
                        payment_status+
                        PAY_AMT6+
                        BILL_AMT6+
                        university_ed+
                        married, data=train_rf)
```

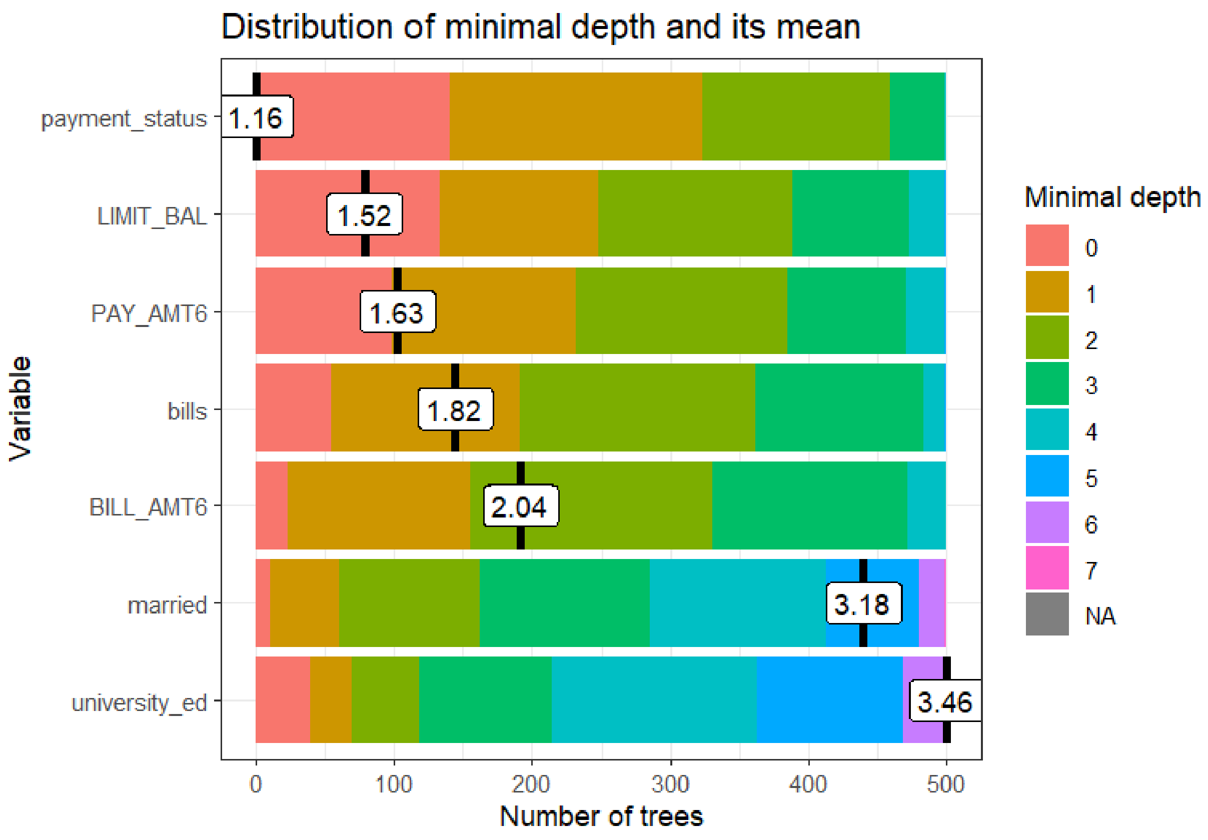
With the standard number of trees (500) the Confusion matrix for this Random Forest model is:

		Reference		
		0	1	Total
Prediction	0	10919	712	11631
	1	2357	1012	3369
	Total	13276	1724	15000

The Random Forest Classifier is trained using bootstrap aggregation where each new tree is built usng a bootstrap sample of the training observations $z_i = (x_i, y_i)$. In this classification there is the Ouf-of-bag error (OOB) which is the average error for each z_i calculated using predictions from the trees that do not contain z_i in their corresponding bootstrap sample. In this model, there is 20.46% OOB error, which is generally a good score in practice.
Also, classification error for non-default and default classes are 6% and 69% respectively. With 80.1% accuracy and 82.6% sensitivity, Random Forest is performing slightly better than the other two models previously built.

4.3.1 Variable importance in Random Forest

In R there are powerful functionalities which were built around Random Forest algorithm. Using **randomForest, randomForestExplainer, tree** and **tidyverse** packages, we can illustrate the variable importance measures in the plot.



The plot shows the variable importance, minimal depth and it's mean. First variable *payment status* is on top with 1.16 score. This means, this is the most decisive variable for building the trees. 1.16 is the average position of the variable in the tree. Obviously, *payment status* is the root node. Other variables can also be interpreted in the same way. As it can be seen, marriage status and educational background has least effect, and therefore, are the least important variables in the tree.

5 Simulation study

5.1 Data generating process

In the simulation part, two approaches will be used. The main difference is in the first method I assign the independent variable *default* using a functional form of dependent variables and I will add some form of error. In the second method, like other variables, response variable *default* is also generated independently. In both scenarios, dependent variables *limited balance, bills, payment status, marriage status, age, university education* are drawn following their empirical distribution observed in the previously used data set.

5.2 Simulation method 1

In this simulation approach, main eight dependent variables are generated. They are scaled and centered around mean of 0. However, these eight variables has no previously assigned relationship, but generated separately. In addition, in the logistic regression *default* variable is also generated randomly. All dependent variables and independent variable are generated inside the loop, and iterated 1000 times. In this pure random setting, I mainly considered the statistical significance of the variables. In the logit model, I controlled for *bills, payment status* and *limited balance.* Within 1000 iterations, generally, variables with p-values under 0.10 occured 80-95 times. A pair of P-values both smaller than 0.10 occured 14 times only. In general, p-values are uniformly distributed across (0,1). Considering pure random behaviour of all the variables in this setting, it is not striking that there are p-values closer to zero. However, there is no evidence to claim that models can perform as good as (p¡10) in such data generating process. I also tested the same procedure using KNN and Random Forest algorithms, and obtained quite similar results: low model accuracy, inconsistent statistical significance of the variables and having no clear trend can be observed in this scenario. Therefore, **I did not implement this approach to the other two algorithms**, and only used second method for DGP - generating the variables using multivariate normal distribution.

5.3 Simulation method 2

As I mentioned above, in this approach I generate multivariate normally distributed data for all eight variables at once using *mvrnorm* command in R. This command helps generate similarly interacted, correlated set of variables. The main important parameter of this setting is the *Sigma* - a covariance matrix of the multivariate normal distribution. For simplicity, I used the correlation table of the previously used data using *cor* command.

Important: while generating the data, there were of a necessity to cut the probabilities into default and non-default. Since I generated everything from scratch, there is no assigned default variable. I chose the threshold values considering the proportion of default (1) and non-default (0) values similar to the *UCICreditCard* data. Around 21% of the users were defaulted, so this is the benchmark for the simulation as well.

5.3.1 Logistic regression

This simulation is run 1000. There are two main differences from the first simulation:

1. All necessary variables are drawn using multivariate normal distributing using *mvrnorm* command. The covariance matrix sigma is taken from previously used data. Therefore, in this setting variables have similar correlations to the ones in the dataset.

2. Response variable *default* is assigned as following:

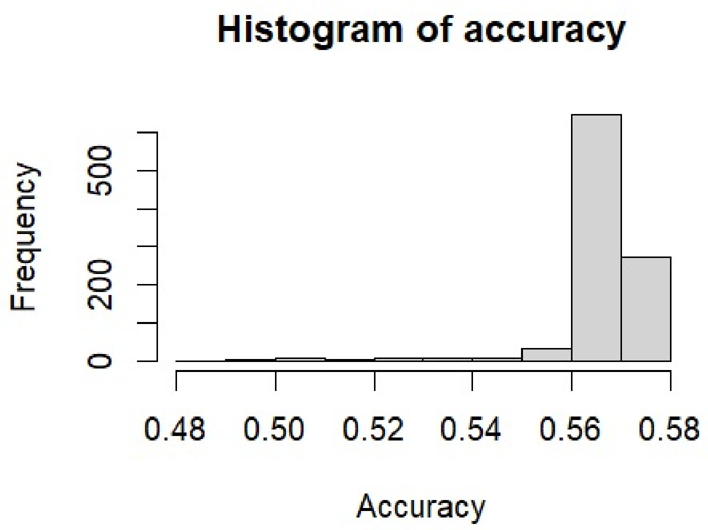
$$P(x)= \frac{1}{1+e^{\beta_0+\beta X}} + \epsilon$$

That is a functional form of a logistic regression and some form of error. Parameters of the logistic model is taken using the first logit model I built.
ϵ - uniform distribution

Second simulation results yield quite satisfactory results compared to the first one. In 1000 iterations, average model accuracy for logistic regression is 57%. From previous models, we have seen that *limited balance, bills and payment status* are found to be the most important variables. In the simulation part for simplicity, I will be running models with only these three variables in order not to increase computational complexity and save time.
Calculating the confusion matrix in each iteration, taking their corresponding values in their positions gives the following result:

		Reference		
		0	1	Total
Prediction	0	127	167	294
	1	6341	8365	14706
	Total	6468	8532	15000

And the distribution of the accuracy rates:
We can see that in this DGP approach, there is a clear trend that the model is performing in the similar scenarios in each iteration. This means, in the second method we could eliminate the pure random behavior of the data. Although 0.57% accuracy is not enough to say this is generally a reliable model, consistency of the main measures such as model accuracy are clearly noticeable.

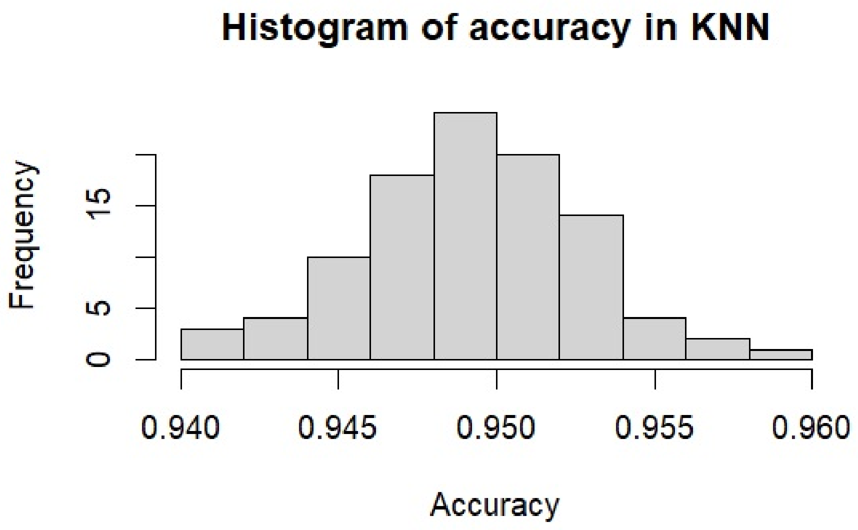


5.4 Simulation with the KNN algorithm

In this simulation, I set K=40 to keep the process similar to the empirical setting we used in the 1st part. The KNN algorithm performs surprisingly better than logistic regression. However, unlike logit model, the KNN predicts only a few non-defaults wrongly. Also, prediction accuracy is significantly higher than that of logit model. Last point to mention, I iterated the simulation loop for KNN 100 times, instead of 1000 like in logit case, since the KNN is computationally more demanding.
The confusion matrix looks as following:

		Reference		
		0	1	Total
Prediction	0	599	0	1851
	1	1252	13149	13149
	Total	599	14401	15000

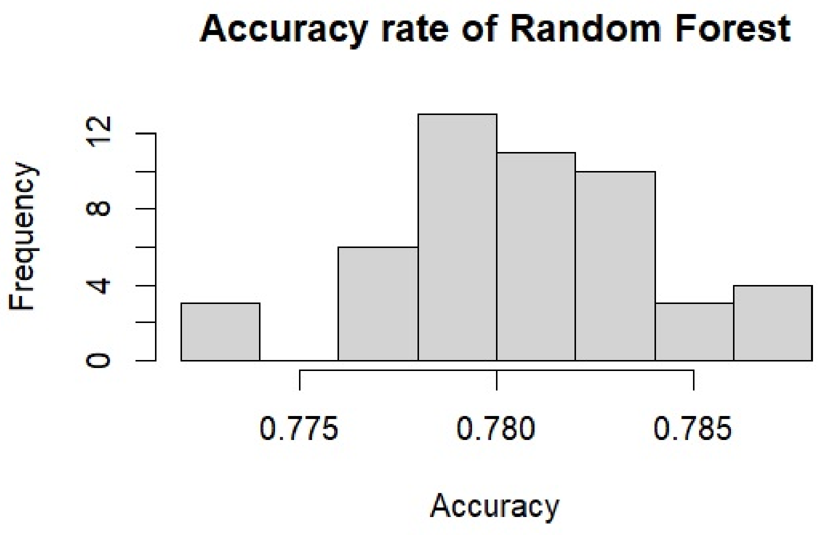
KNN yields on average of 92% accuracy, 32% sensitivity, full 100% specificity and kappa=0.42. Having said this, there could be a potential problem of overfitting in this part. Yet, the main parameter K set to be 40, similar to the one we looked at in empirical part. The main concern for overfitting, supposedly, K is large enough to avoid this problem. Suprisingly big accuracy rate for KNN needs to be further researched, and be shown that the model is actually working, or the potential biases should be eliminated. I leave this question unanswered due to time and size constraints.



5.5 Random Forest Simulation

Holding the data generating process similar to the previous two algorithms, in this simulation only the classification algorithm has been changed. Conventional random forest model is built with 500 trees and 50 iterations. Unlike the other two simulations, I kept the iterations lower in the Random forest due to the limitations of this computationally expensive process.
In this environment, Random Forest could not outperform the KNN algorithm yielding 78% accuracy. The confusion matrix for this case is:

		Reference		
		0	1	Total
Prediction	0	2090	1646	3736
	1	1642	9622	13144
	Total	3732	11268	15000



Random Forest prediction accuracy in 50 iteration.

Simulated setting of this Random Forest model is clearly performed quite similar to the actual data. Both performing around 80% accuracy. **See the appendix to compare all the models.**

6 Conclusions and heuristics

We have seen three widely used machine learning algorithms within the scope of credit card default modeling. First of all, taking a look at the research background of this topic, we can see that Artificial Neural Networks and Support Vector Machines are also widely used to solve this and similar kind of prediction problems. Following the module, I decided not to implement these two models, yet would like to further research on this topic.
This and similar settings can be further developed, deployed and used in the final products such as online banking, credit fraud detection and further allocating commercial and household loans.
In the variable importance context, we have seen that limited balance, bills and payment status are the most important variables that determines the client's default or non-default. Demographic variables - age, education and marital status has little, if not any effect on being defaulted.
Having the same model accuracy does not always mean models are performing in the same way. In the empirical part, we have seen that specificity - the prediction of true negatives - is noticeably higher in the logistic regression, meaning it can predict non-default clients better than other models. The same can be said to other measurements across models. Even with the similarly performing models, we can specifically choose one of them depending on the question - whether we want to predict who is actually going to default, or the best model that yields least error overall and so on.
In the simulated logistic scenario, there was significantly higher noise in the generated data since 30000 observations are placed between 0 and 1. This is, in my opinion, the main reason for not being able to predict well enough like other two models, because during model building process finding the best cutoff points was tricky part, and slight changes could make the model upside-down. Therefore, in such environment pure classification algorithms performed significantly better by all measurements.

7 Important and Frequently Used Definitions

Model Accuracy - accuracy is defined as the number of classifications a model correctly predicts divided by the total number of predictions made.
Sensitivity - the proportion of true positives that are correctly predicted by the model.
Specificity - the proportion of true negatives that are correctly predicted by the model.
Detection Rate - the proportion of the whole sample where the events were detected correctly.
Balanced Accuracy - the proportion of real positives that are correctly predicted out of total positive prediction made by the model.
Prevelance -
Kappa - a measure of how closely the instances classified by the machine learning classifier matched the data labeled as ground truth, controlling for the accuracy of a random classifier as measured by the expected accuracy. There is no standardized way to interpret its values. Landis and Koch (1977) provide a way to characterize values. According to their scheme a value ¡ 0 is indicating no agreement , 0–0.20 as slight, 0.21–0.40 as fair, 0.41–0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1 as almost perfect agreement.
McNemar's p-value - the probability of observing this empirical (or a larger) chi-squared value.

References

1. An Introduction to Statistical Learning: With Applications in R. Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, 2013.
2. Prediction of credit card defaults through data analysis and machine learning techniques Saurabh Arora, Sushant Bindra, Survesh Singh, Vinay Kumar Nassa. Materials Today: Proceedings.
3. Yanxia Zhang and Yongheng Zhao. Classification in Multidimensional Parameter Space: Methods and Examples. Astronomical Society of the Pacific, Vol. 115, No. 810 (August 2003).
4. John A. Haslem and William A. Longbrake. A Credit Scoring Model for Commercial Loans.

Appendix I

	Empirical								
	Logit			KNN			Random Forest		
		0	1		0	1		0	1
Sample Confusion matrix	0	11472	250	0	10994	728	0	10978	744
	1	2766	512	1	2207	1071	1	2243	1035
Model Accuracy	79.89%			80.43%			80.09%		
Sensitivity	80.57%			83.28%			83.03%		
Specificity	67.19%			59.53%			58.18%		
Detection Rate	76.48%			73.29%			73.19%		
Balanced Accuracy	73.88%			71.41%			70.61%		
Kappa	0.1864			0.316			0.302		
McNemar's p-value	<2e-16			<2e-16			<2e-16		
	Simulated								
		0	1		0	1		0	1
Sample Confusion matrix	0	586	5615	0	2599	0	0	2090	1637
	1	872	7927	1	696	11705	1	1622	9651
Model Accuracy	56.75%			95.36%			78.27%		
Sensitivity	40.19%			78.88%			56.30%		
Specificity	58.53%			100.00%			85.50%		
Detection Rate	3.90%			17.33%			13.93%		
Balanced Accuracy	49.36%			89.44%			70.90%		
Kappa	0.052			0.8535			0.4175		
McNemar's p-value	<2e-16			<2.2e-16			0.8063		