STAT652 Project

Chun Yin Kong

Abstract

In the lesson we covered a number of machine learning algorithms which able to build classifiers to data for prediction. They are:

- Null model
- Logistic Regression
- Decision Tree
- Random forest
- Neural network
- Naive Bayes
- k-NN

In this project we are going to cover all algorithms which listed above and the relevant codes will be listed in separate rmd files for easier reference.

Initial Setup

```
library(tidyverse)
library(tidymodels)
library(mdsr)
library(parallel)
library(remotes) # For installing traversc/trqwe
library(trqwe) #parallel 1
library(tictoc)
library(future) # parallel processing 2
set.seed(123123)
```

Data Preparation

Loading Data

Reading CSV into R and turn it to RDS format

```
#Run Once Only
accepted_data <- read.csv("data/accepted_2007_to_2018Q4.csv")
rejected_data <- read.csv("data/rejected_2007_to_2018Q4.csv")
#Run Once Only
saveRDS(accepted_data, file="data/accepted_data.rds")
saveRDS(rejected_data, file="data/rejected_data.rds")
remove(accepted_data, rejected_data) # Avoid insufficient RAM Removing data frames in R
```

Introduction

This project is going to analyze the status of loan (loan_status) using different machine learning algorithms and perform prediction of classifying the new loan and the repayment status.

Refitting all 2012-2014 Data to classify all 2015 data

Throughout all the testing, I found that using Neural Network gives the best accuracy among all the algorithms used. Hence in the following paragraphs, I will use Random Forest to fit all the data from 2012 to 2014 again to the model, and classify all 2015 of the loan data.

```
accepted_data <- readRDS(file="data/accepted_data.rds")
```

Preparing Data for training and testing

```
data_2012_2014_train <- accepted_data %>%
  select(-id, -member_id) %>% #removing id variables
  mutate(loan_status = as.factor(loan_status)) %>%
  filter(str_detect(issue_d, "2012|2013|2014")) %>%
  select(loan_status, where(is.numeric))

data_2015_test <- accepted_data %>%
  select(-id, -member_id) %>% #removing id variables
  mutate(loan_status = as.factor(loan_status)) %>%
  filter(str_detect(issue_d, "2015")) %>%
  select(loan_status, where(is.numeric))
```

Neural Network Training on 2012-2014 Data

```
form <- as.formula("loan_status ~ loan_amnt + funded_amnt +
                   funded_amnt_inv + int_rate + installment + annual_inc +
                   delinq_2yrs + fico_range_low + fico_range_high")
```

In this training, I set the number of epochs to 5 for two reasons. - Saving Computer Resources. For a CPU based R setting, it is very time consuming to use run a very large neural network without parallelization or GPU computing. Setting a smaller number of epochs can save computing time and computer resources. - Maintaining best result. Since in the neural network R markdown file, I trained the model with 100 epochs. The loss in the training is flattened started around 5 epochs. Which means, the performance of the model becomes steady after 5 epochs passed. So I reckon for training 2012-2014 data, using 5 epochs would be enough, for both minimizing the computing time and maximize the model accuracy on prediction.

```
biv_rec <-
  recipe(form, data = data_2012_2014_train) %>%
  step_BoxCox(all_predictors()) %>%
  step_normalize(all_predictors()) %>%
  prep(training = data_2012_2014_train)

test_normalized <- bake(biv_rec, new_data = data_2015_test, all_predictors())

mod_nn2 <-mlp(epochs = 5, hidden = 20, dropout = 0.1) %>%
  set_mode("classification") %>%
  set_engine("keras") %>%
  fit(form, data=bake(biv_rec, new_data = NULL))
```

Evaluating Model Performance

```r
pred2 <- data_2015_test %>%
  bind_cols(
    predict(mod_nn2, new_data = test_normalized, type = "class")
  ) %>%
  rename(loan_status_nn = .pred_class)
```

```r
pred2 %>%
  accuracy(loan_status, loan_status_nn)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.712
```

The accuracy of neural network model in fitting all 2015 data seems getting a drop when comparing to fitting a training data. In the later setting, I believe we can include all variables to be train in the neural network and achieve a higher accuracy on predicting new loan status.

Conclusion

The performance of fitting all 2015 data to the neural network model that trained by fitting 2012-2014 data is not very accurate as reflected by the accuracy table above. There may be reason that because I dropped all non-numeric variables in the dataset for training and testing. Secondly, the hidden-layer of the neural netowrk may not be enough to process the loan dataset. Due to the limit computing power of the computer, at this time we are not able to train a very large neural network with this amount of data (around 400000 observations.)