Chan_Hou_Long_3035745312_P

Chan Hou Long, Guyver

12/12/2021

1 Data importation

```
setwd("~/Documents/HKU/STAT2604/proj")
customer <- read.table("Customer Data", header = TRUE, sep = ";")
```

The dataset has 2000 rows and 18 columns.

```
dim(customer)
```

```
## [1] 2000   18
```

The 2 - 17 variables are the potential explanatory variables, the `Good_Customer` and `Bad_Customer` are the dependent variable. `ID` is the row number with an empty column `X`.

Since `Good_Customer` and `Bad_Customer` are the same when a customer is defined as good, `Bad_Customer` is No and `Good_Customer` is Yes, we only take `Good_Customer` for the study. There is only 296 good customer with 1683 bad customers and 21 undefined.

1.1 Data handling

[EDA q1] After changing empty value as `NA` and character columns as factors, the mean of `Annual_Income` of customers is $38003 with a maximum of 252193 and a minimum of 1377. They have applied for loans approximately twice in the past five years and currently held around 2 credit cards each. The maximum loan amount is 766612 and minimum 11020 with an average of 124010 among the customers. Most of them only have 1 or 2 family members that rely on themselves and `Number_of_Dependants` contains 246 missing values. The largest group of the employment level is part-time with 654 people. They have a mean of 17.56% of monthly gross earnings for monthly installments. With the largest 40% and smallest 15.2%, there are 24 NAs in `Installment_Percentage`. Most of the customers have around 7 years working years and 5 years living in the same places with an average age of 35.79. The majority of them do not miss or delay payments over the last 3 years and live on their own property. Most of them also have 1 extra line of credits and location 3 for receiving applications.

```
customer[customer==""] <- NA
customer = customer[, 2:16]
customer[,7] = readr::parse_number(customer[,7])
customer[,6] = factor(customer[,6], levels = c("1", "2", "3", "4", "5"))
customer[,11:15] <- lapply(customer[,11:15], factor)
summary(customer)
```

```
##  Annual_Income    Credit_History   Credit_Cards       Amount
##  Min.   :  1377   Min.   :0.000   Min.   :0.000   Min.   : 11020
##  1st Qu.: 16035   1st Qu.:1.000   1st Qu.:0.000   1st Qu.: 58075
##  Median : 25874   Median :2.000   Median :2.000   Median : 87600
##  Mean   : 38003   Mean   :2.094   Mean   :1.891   Mean   :124010
##  3rd Qu.: 50485   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:161415
##  Max.   :252193   Max.   :9.000   Max.   :4.000   Max.   :766612
##
##  Number_of_Dependants Employment Installment_Percentage
##  Min.   :1.000        1:128      Min.   :15.20
##  1st Qu.:1.000        2:315      1st Qu.:15.99
##  Median :1.000        3:654      Median :16.94
##  Mean   :1.658        4:370      Mean   :17.56
##  3rd Qu.:2.000        5:533      3rd Qu.:18.12
##  Max.   :3.000                   Max.   :40.00
##  NA's   :246                     NA's   :24
##  Time_at_Current_Employment Time_at_Address       Age
##  Min.   : 1.000             Min.   : 0.000   Min.   :19.00
##  1st Qu.: 5.000             1st Qu.: 3.000   1st Qu.:27.00
##  Median : 7.000             Median : 5.000   Median :33.00
##  Mean   : 6.897             Mean   : 4.997   Mean   :35.79
##  3rd Qu.: 9.000             3rd Qu.: 6.000   3rd Qu.:42.25
##  Max.   :17.000             Max.   :14.000   Max.   :75.00
##
##  Delayed_Missed_Payments        Residential_Status Existing_Credits
##  0:1685                  Live with Family: 201     1:1253
##  1: 288                  Own             :1437     2: 678
##  2:  27                  Rent            : 362     3:  58
##                                                    4:  11
##
##
##
##  Area_Indicator Good_Customer
##  0: 42          No  :1683
##  1:205          Yes : 296
##  2:586          NA's:  21
##  3:791
##  4:376
##
##
```

Assuming data is MCAR, which missing completely at random, 5% of the total for large datasets is the safe maximum threshold. Since `Number_of_Dependants` is missing around 12.3%, NAs in this column should be dropped. `Installment_Percentage` and `Good_Customer` can keep as the percentage below 5%.

```r
per_of_missing <- function(x) {sum(is.na(x))/length(x)*100}
apply(customer, 2, per_of_missing)
```

```
##              Annual_Income             Credit_History
##                       0.00                       0.00
##               Credit_Cards                     Amount
##                       0.00                       0.00
##       Number_of_Dependants                 Employment
```

```
##                    12.30                      0.00
##   Installment_Percentage Time_at_Current_Employment
##                     1.20                      0.00
##          Time_at_Address                       Age
##                     0.00                      0.00
##    Delayed_Missed_Payments        Residential_Status
##                     0.00                      0.00
##         Existing_Credits            Area_Indicator
##                     0.00                      0.00
##            Good_Customer
##                     1.05
```

2 Error handling

[EDA q4] After removing NAs in `Number_of_Dependantsin`, `Installment_Percentage` and `Good_Customer` keep below the percentage below 5%.

```
customer = customer[!is.na(customer$Number_of_Dependants),]
# apply(customer, 2, per_of_missing)
```

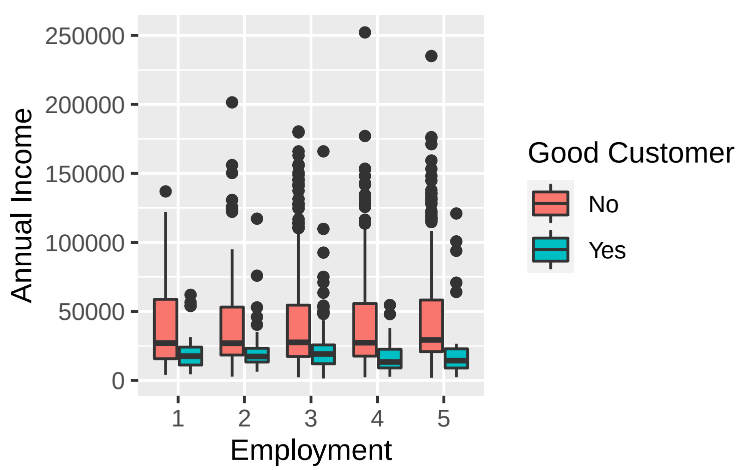
Then, data will be imputed by using `mice()`. Dataframe `df` is created with 1754 rows and no NA values.

```
df <- complete(impute, 1)
# apply(apply(df,2,is.na),2,sum) ; nrow(df)
```

3 Exploratory Analysis

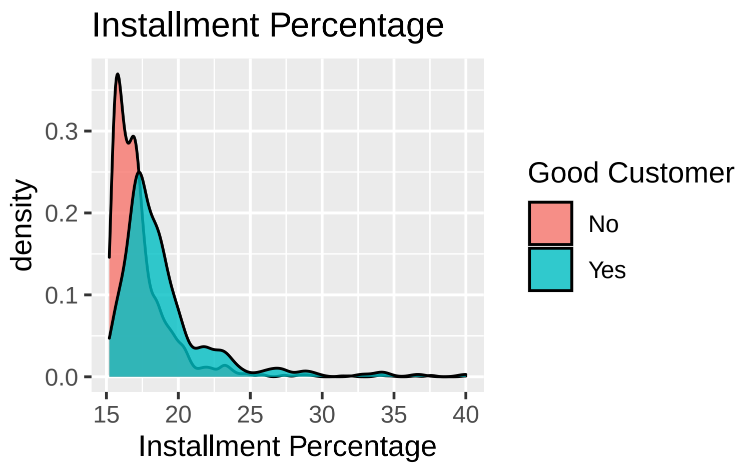
From the bar chart, it is interesting to discover that customer who is labelled as bad customer normally has higher annual income than the good customer in all employment status.

```
ggplot(df, aes(x=Employment, y=Annual_Income, fill=Good_Customer)) +
    geom_boxplot() + ylab("Annual Income") + labs(fill = "Good Customer")
```



From the density graph, both good and bad customers are right-skewed. However, the bad customer tends to have a smaller installment percentage compared to the good customer.

```
ggplot(df, aes(x = Installment_Percentage, fill = Good_Customer)) +
  geom_density(alpha=0.8) + ggtitle("Installment Percentage") +
  xlab("Installment Percentage") + labs(fill = "Good Customer")
```

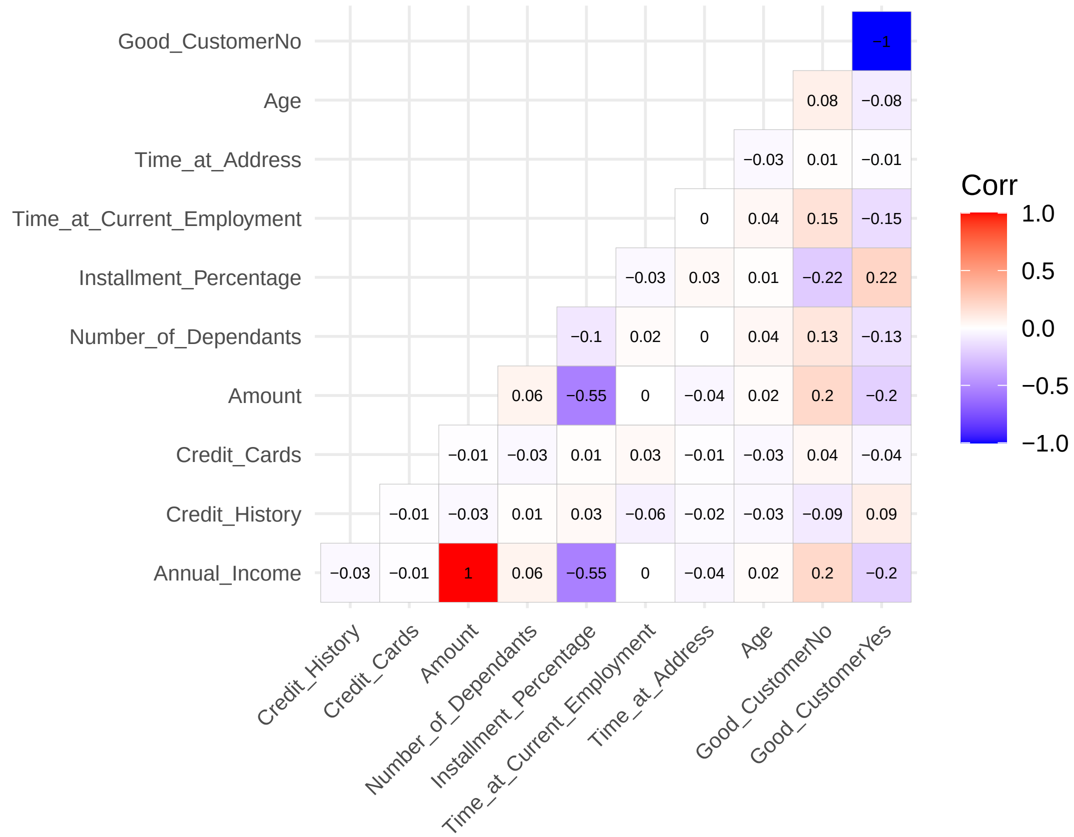


3.1 Correlation analysis

[EDA q2, q3] From the below two plots of the correlation matrix, it can be seen that the top 3 variables associated with `Good_Customer` with the largest correlation coefficient are `Area_Indicator` of 0.31, `Installment_Percentage` of 0.22, `Annual_Income` and `Amount` of 0.2.
Moreover, `Annual_Income` is associated with `Amount` forming a perfect positive relationship with the coefficient is 1. `Installment_Percentage` is associated with `Amount` and `Annual_Income` sharing a moderate negative relationship with the coefficient is -.55.

```
df_fac = df %>% dplyr::select(where(is.factor))
cramer = matrix(NA, ncol(df_fac), ncol(df_fac))

for (i in (1:ncol(df_fac))){
  for (j in (1:ncol(df_fac))){
    tab = table(df_fac[, i], df_fac[, j])
    chisq_results = chisq.test(tab)
    cramer[i, j] = sqrt(chisq_results$statistic/(nrow(df_fac) * (min(dim(tab)) -1)))
  }
}

cramer = round(cramer, 3)
colnames(cramer) = colnames(df_fac)
rownames(cramer) = colnames(df_fac)

corrplot(cramer, method = "shade", type = "upper", diag = F, tl.srt = 45,  tl.col = "black",
         tl.cex = 0.6, addCoef.col = "darkgreen", addCoefasPercent = T)
```


```
df_num = df %>% dplyr::select(where(is.numeric))
df_num$Good_Customer = df$Good_Customer

model.matrix(~0+., data=df_num) %>%
  cor(use="pairwise.complete.obs") %>%
  ggcorrplot(show.diag = F, type="lower", lab=TRUE, lab_size=2) +
  theme(axis.text.x = element_text(size=8),
        axis.text.y = element_text(size=8))
```



4 Predictive model

[Modeling q1] Data is split into training and validation sets using an 80%/20% split.

```
trainIndex <- createDataPartition(df$Good_Customer, p = .8, list = FALSE)
train <- df[ trainIndex,]
test  <- df[-trainIndex,]
```

4.0.1 Functions for models

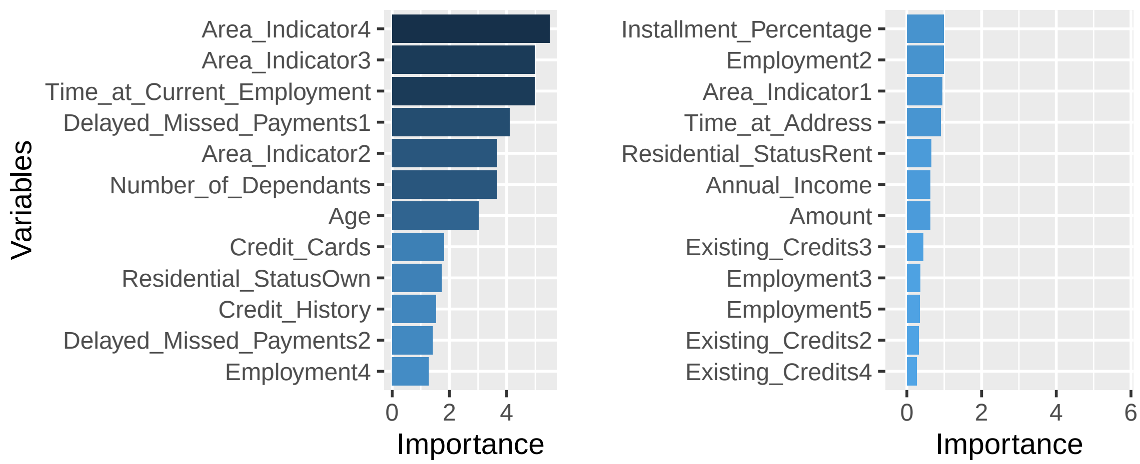
4.1 Logistic regression

[Modeling q3] The logistic regression model is used to understand the relationship between the dependent variable `Good_Customer` and the independent variables. The logistic regression contains every remaining feature. It can be seen that a lot of features are non-significant in this model including `Annual_Income`, `Credit_History`, `Amount`, `Employment`, `Installment_Percentage` and `Existing_Credits`. The top 3 most important explanatory variables are `Area_Indicator` factor 4, 3 of 5.49 and 4.97 respectively, `Time_at_Current_Employment` of 4.96 and `Delayed_Missed_Payments` factor 1 of 4.1. `Installment_Percentage`, `Annual_Income` and `Amount` are not an essential variable anymore compared to the result in EDA.

```r
set.seed(5312)
lr.fit <- glm(formula = Good_Customer ~ ., data = train, family=binomial)
summary(lr.fit)
```

```
##
## Call:
## glm(formula = Good_Customer ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0041  -0.5262  -0.3189  -0.1614   3.4657
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.6051707  8.4551665  -0.190 0.849431
## Annual_Income             -0.0015860  0.0025237  -0.628 0.529710
## Credit_History             0.0891120  0.0582376   1.530 0.125980
## Credit_Cards              -0.1110654  0.0616266  -1.802 0.071509 .
## Amount                     0.0005208  0.0008412   0.619 0.535843
## Number_of_Dependants      -0.4366003  0.1190781  -3.667 0.000246 ***
## Employment2                0.3764137  0.3840221   0.980 0.326994
## Employment3               -0.1309162  0.3684652  -0.355 0.722364
## Employment4               -0.5154133  0.4037681  -1.277 0.201776
## Employment5               -0.1282863  0.3808623  -0.337 0.736244
## Installment_Percentage     0.0358266  0.0363101   0.987 0.323797
## Time_at_Current_Employment -0.1823571  0.0367334  -4.964 6.89e-07 ***
## Time_at_Address           -0.0353722  0.0394051  -0.898 0.369370
## Age                       -0.0261268  0.0086589  -3.017 0.002550 **
## Delayed_Missed_Payments1   0.8895632  0.2167221   4.105 4.05e-05 ***
## Delayed_Missed_Payments2   0.9272835  0.6577541   1.410 0.158607
## Residential_StatusOwn     -0.4697772  0.2723854  -1.725 0.084585 .
## Residential_StatusRent    -0.2049847  0.3149101  -0.651 0.515091
## Existing_Credits2         -0.0595989  0.1893187  -0.315 0.752908
## Existing_Credits3         -0.2284559  0.5210192  -0.438 0.661039
## Existing_Credits4          0.2556108  0.9737457   0.263 0.792934
## Area_Indicator1           -0.5114393  0.5436717  -0.941 0.346852
## Area_Indicator2           -1.9307054  0.5260405  -3.670 0.000242 ***
## Area_Indicator3           -2.6402724  0.5310972  -4.971 6.65e-07 ***
## Area_Indicator4           -3.1664162  0.5769550  -5.488 4.06e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1170.89  on 1403  degrees of freedom
## Residual deviance:  881.79  on 1379  degrees of freedom
## AIC: 931.79
##
## Number of Fisher Scoring iterations: 6
```

```r
lr_imp_split(lr.fit)
```

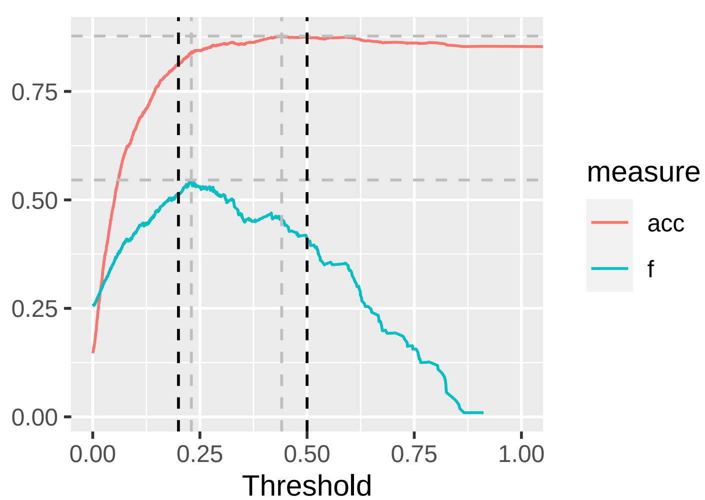


Then, both training and test data are used to evaluate how well did the training go. The validation scores will be used to validate the model.

```
pred.lr.train = predict(lr.fit, newdata = train, type = "response")
pred.lr.test = predict(lr.fit, newdata = test, type = "response")
```

This plot shows the evolution of the accuracy and F1 score rates according to the cut level. The result should have a good F1 score without dropping too much on accuracy. The 0.2 cut seems a good settlement that is the trade-off between accuracy and F1 score.

```
train_score = lr_cutoff(pred.lr.train, train$Good_Customer, "acc", "f")
train_score + geom_vline(xintercept = c(0.2, 0.5), linetype = "dashed")
```



Best cut for acc : x = 0.441
Best cut for f : x = 0.23

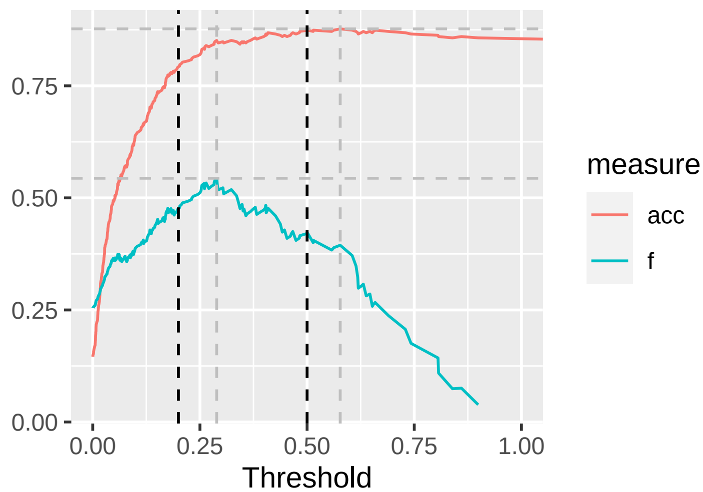
From the summary of the training set, the accuracy reaches 81.34% and the sensitivity rate is 67.48%, which means the model manages to correctly label 81.34% of the times and 67.48% of the good customers are correctly detected.

```r
lr_train_cut = 0.2
lr_train_class = lr_cut_pred(pred.lr.train, lr_train_cut)
confusionMatrix(lr_train_class, as.factor(as.numeric(c(0,1))[train$Good_Customer]),
                                       positive = "1", mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1003   67
##          1  195  139
##
##                Accuracy : 0.8134
##                  95% CI : (0.792, 0.8335)
##     No Information Rate : 0.8533
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4072
##
##  Mcnemar's Test P-Value : 4.292e-15
##
##             Sensitivity : 0.6748
##             Specificity : 0.8372
##          Pos Pred Value : 0.4162
##          Neg Pred Value : 0.9374
##               Precision : 0.4162
##                  Recall : 0.6748
##                      F1 : 0.5148
##              Prevalence : 0.1467
##          Detection Rate : 0.0990
##    Detection Prevalence : 0.2379
##       Balanced Accuracy : 0.7560
##
##        'Positive' Class : 1
##
```

4.2 Validation for Logistic regression

```r
test_score = lr_cutoff(pred.lr.test, test$Good_Customer, "acc", "f")
test_score + geom_vline(xintercept = c(lr_train_cut, 0.5), linetype = "dashed")
```



[Modeling q2] The performance is close to the training set, which means it does not suffer from over fitting. From the summary of the test data set, the accuracy reaches 79.43% and the sensitivity rate is 64.71%, which means the model manages to correctly label 79.43% of the times and 64.71% of the good customers are correctly detected.
While ensuring that 5% of the bad customers are wrongly identified, which is false negatives, the proportion of good customers that can be granted loans which is true positives (1-0.05) = 0.95. If 1% FN, TP is 0.99. If 0.5% FN, TP is 0.995. From the confusion matrix, since the sensitivity is 0.6471 while ensuring that (1-.6471) = 35.29% of the bad customers are wrongly identified.

```
lr_test_class = lr_cut_pred(pred.lr.test, lr_train_cut)
confusionMatrix(lr_test_class, as.factor(as.numeric(c(0,1))[test$Good_Customer]),
                                               positive = "1", mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245  18
##          1  54  33
##
##                Accuracy : 0.7943
##                  95% CI : (0.7481, 0.8354)
##     No Information Rate : 0.8543
##     P-Value [Acc > NIR] : 0.9991
##
##                   Kappa : 0.3608
##
##  Mcnemar's Test P-Value : 3.711e-05
##
##             Sensitivity : 0.64706
##             Specificity : 0.81940
##          Pos Pred Value : 0.37931
##          Neg Pred Value : 0.93156
##               Precision : 0.37931
##                  Recall : 0.64706
##                      F1 : 0.47826
```

```
##             Prevalence : 0.14571
##         Detection Rate : 0.09429
##   Detection Prevalence : 0.24857
##      Balanced Accuracy : 0.73323
##
##       'Positive' Class : 1
##
```