Modern Data Mining - Final Project on Analyzing Credit Scores

Zhijian Wang Vikramaditya Singh Keshav Ramesh

Due: 11:59Pm, 4/30, 2023

Contents

Description

It is always difficult to find out what actually make our credit scores become better or become worse. So in this project, we are going to figure out which factors are the best predictors of credit scores, and which models give us the greatest accuracy in our results. We are going to use EDA and machine learning models in our data analysis process.

Variables:

ID: a unique identification for each row; Customer_ID: a unique identification of a person; Month: month of the year; Name: name of the person; Age: age of the person; SSN: social security number of the person; Occupation: occupation of the person; Annual_Income: annual income of the person; Monthly_Inhand_Salary: monthly base salary of the person; Num_Bank_Accounts: number of bank accounts the person holds;

Num_Credit_Card: number of other credit cards held by the person; Interest_Rate: interest rate on credit card; Num_of_Loan: number of loans taken from the bank; Type_of_Loan: types of loan taken by the person; Delay_from_due_date: average number of days delayed from the payment date; Num_of_Delayed_Payment: average number of payments delayed by the person; Changed_Credit_Limit: percentage change in credit card limit;

Num_Credit_Inquiries: number of credit card inquiries; Credit_Mix: classification of the mix of credits; Outstanding_Debt: remaining debt to be paid; Credit_Utilization_Ratio: utilization ratio of credit card; Credit_History_Age: age of credit history of the person; Payment_of_Min_Amount: checks whether only the minimum amount was paid by the person; Total_EMI_per_month: monthly EMI payments;

Amount_invested_monthly: monthly amount invested by the customer; Payment_Behaviour: payment behavior of the customer; Monthly_Balance: monthly balance amount of the customer; Credit_Score: bracket of credit score (Poor, Standard, Good);

Objectives

- Data reading/cleaning, EDA, ML model, Logistic regression, ML model, PCA, Neural Networks, Decision Trees, Random Forest; Data needed: `train.csv`

Data Reading and Cleaning

As we can see, we have 100000 rows and 28 columns.

```
df <- read.csv("/Users/keshavramesh/STAT471/finalproject/data/train.csv", header = T)
cat("Data set Shape:")
```

```
## Data set Shape:
```

```
dim(df)
```

```
## [1] 100000     28
```

We first need to change the Type_Of_Loan Column. After we observe the data, we find that these data needs to filtered and understood better. We have so many categories. We need to reduce them.

```
cat("Before Conversion: Number of Unique Type in Loan")
```

```
## Before Conversion: Number of Unique Type in Loan
```

```
loan_types <- unique(df$Type_of_Loan)
length(loan_types)
```

```
## [1] 6261
```

```
total_types <- c()
for (e in df$Type_of_Loan){
  e <- str_replace(e, "and ", " ")
  more_elements <- strsplit(e, ", ")
  for (e2 in more_elements){
    total_types <- append(total_types, trimws(tolower(e2)))
  }
  total_types <- unique(total_types)
}
cat("After ConversionNumber of Unique Type in Loan")
```

```
## After ConversionNumber of Unique Type in Loan
```

```
total_types
```

```
##  [1] "auto loan"                "credit-builder loan"
##  [3] "personal loan"            "home equity loan"
##  [5] "not specified"            "no data"
##  [7] "mortgage loan"            "student loan"
##  [9] "debt consolidation loan" "payday loan"
```

We first need to change the Type_Of_Loan Column. After we observe the data, we find that these data needs to filtered and understood better. We have so many categories. We need to reduce them.

```
for (l in total_types){
   df <- df %>%
    mutate(!!l := if_else(grepl(l, tolower(Type_of_Loan)), "1", "0"))
}
df <- clean_names(df)
```

We will now do the transformation for "Payment_Behaviour" To Extract "Spent" and "Value". However, because R is only so good in when doing analysis, we will first try to only include Payment Behaviour only include "spent" and "value."

```
cat("Unique payment_behaviour")
```

```
## Unique payment_behaviour
```

```
length(unique((df$payment_behaviour)))
```

```
## [1] 6
```

```
df$spent <- sapply(strsplit(tolower(df$payment_behaviour), "_"), "[", 1)
df$value <- sapply(strsplit(tolower(df$payment_behaviour), "_"), "[", 3)
cat("Unique class in spent")
```

```
## Unique class in spent
```

```
unique(df$spent)
```

```
## [1] "high" "low"
```

```
cat("Unique class in value")
```

```
## Unique class in value
```

```
unique(df$value)
```

```
## [1] "small"  "large"  "medium"
```

And lastly, we are going to drop some columns and change some categorical variables to factor variables.

```
drops <- c("id","customer_id", "name", "ssn", "type_of_loan")
df$occupation <- as.factor(df$occupation)
df$credit_mix <- as.factor(df$credit_mix)
df$payment_of_min_amount <- as.factor(df$payment_of_min_amount)
df$payment_behaviour <- as.factor(df$payment_behaviour)
df$spent <- as.factor(df$spent)
df$value <- as.factor(df$value)
df$credit_score <- as.factor(df$credit_score)
df <- df[,!(names(df) %in% drops)]
df <- as.data.frame(unclass(df),stringsAsFactors=TRUE)
names(df)
```

```
##  [1] "month"                    "age"
##  [3] "occupation"               "annual_income"
##  [5] "monthly_inhand_salary"    "num_bank_accounts"
##  [7] "num_credit_card"          "interest_rate"
##  [9] "num_of_loan"              "delay_from_due_date"
## [11] "num_of_delayed_payment"   "changed_credit_limit"
## [13] "num_credit_inquiries"     "credit_mix"
## [15] "outstanding_debt"         "credit_utilization_ratio"
## [17] "credit_history_age"       "payment_of_min_amount"
## [19] "total_emi_per_month"      "amount_invested_monthly"
## [21] "payment_behaviour"        "monthly_balance"
## [23] "credit_score"             "auto_loan"
## [25] "credit_builder_loan"      "personal_loan"
## [27] "home_equity_loan"         "not_specified"
## [29] "no_data"                  "mortgage_loan"
## [31] "student_loan"             "debt_consolidation_loan"
## [33] "payday_loan"              "spent"
## [35] "value"
```

Exploratory Data Analysis (EDA)

Bar Plot of Credit Scores

```
level <- c("Poor", "Standard", "Good")
# df %>% dplyr::mutate(credit_score = factor(credit_score,
                                    # levels = level)) %>%
# ggplot(aes(x=credit_score, fill = credit_score) ) + geom_bar() + geom_text(stat='count', aes(label=af
        # ggtitle("Bar Plot Credit Score") + theme(plot.title = element_text(hjust = 0.5))
```

As we can see, more than half of the people are going to get a Standard credit score. There are only 1/5 of people that can have a Good credit score.

Spaghetti Plot of Age versus Income

```
df_age_credit_salary <- df %>% group_by(age, credit_score) %>% summarize(mean_salary = mean(annual_incor
# ggplot(df_age_credit_salary, aes(x = factor(age), y = mean_salary, group = credit_score, color = cred
```

After observing the data, we can say there is a strong correlation between annual salary and credit score. We can also find that the people tend to get higher credit score when their salary is higher. Also, we can

see that when people are over 47 years old there does not seem to be a big correlation between salary and credit score.

Correlation Heatmap

```
library(ggcorrplot)
# model.matrix(~0+., data=df) %>%
        # cor(use="pairwise.complete.obs") %>%
        # ggcorrplot(show.diag=FALSE,  lab=TRUE, lab_size=1)
```

Machine Learning Model through Logistic Regression and LASSO

Logistic Regression Model with Lasso

Train and test split in R. In the class we have been taught binomial logistic regression, so we will only select good and poor credit scores and attempt to predict them.

```
df_ml <- df[seq(1, nrow(df), 12), ]
df_ml <- df_ml[which(df_ml$credit_score == "Good" | df_ml$credit_score == "Poor"), ]
cat("Before Conversion")
```

```
## Before Conversion
```

```
table(df_ml$credit_score)
```

```
##
##     Good     Poor Standard
##     1419     2438        0
```

```
i <- sapply(df_ml, is.factor)
df_ml[i] <- lapply(df_ml[i], as.character)
df_ml <- as.data.frame(unclass(df_ml),stringsAsFactors=TRUE)
cat("After Conversion")
```

```
## After Conversion
```

```
table(df_ml$credit_score)
```

```
##
## Good Poor
## 1419 2438
```

```
smp_size <- 0.75 * nrow(df_ml)
train_ind <- sample(seq_len(nrow(df_ml)), size = smp_size)
df_train <- df_ml[train_ind, ]
df_test <- df_ml[-train_ind, ]
X <- model.matrix(credit_score ~.,df_train)[,-1]
Y <- df_train[, "credit_score"]
fit1.cv <- cv.glmnet(X, Y, alpha=1, family="binomial", nfolds = 10, type.measure = "deviance")
# plot(fit1.cv)
```

From this graph, lambda.min = 32 and lambda.1se = 12.

```
coef.1se <- coef(fit1.cv, s="lambda.1se")
coef.1se <- coef.1se[which(coef.1se !=0),]
coef.1se[order(abs(coef.1se), decreasing = TRUE)]
```

```
##                                       (Intercept)
##                                     -1.845543e+00
##                                    credit_mixGood
##                                     -7.420761e-01
##                                   num_credit_card
##                                      2.378781e-01
##                             payment_of_min_amountYes
##                                      1.610780e-01
##                                          no_data1
##                                     -8.925116e-02
##                                     interest_rate
##                                      6.214836e-02
##                              payment_of_min_amountNo
##                                     -6.166783e-02
##                                       num_of_loan
##                                      5.773472e-02
## payment_behaviourLow_spent_Medium_value_payments
##                                      4.874576e-02
##                                 delay_from_due_date
##                                      4.681525e-02
##                                      payday_loan1
##                                      3.172312e-02
##                                 occupationDeveloper
##                                     -2.546193e-02
##                                    mortgage_loan1
##                                      2.466109e-02
##                                num_credit_inquiries
##                                      9.978322e-03
##                                  credit_history_age
##                                     -2.144022e-03
##                                 total_emi_per_month
##                                     -8.319864e-04
##                                   outstanding_debt
##                                      2.068377e-04
##                                    monthly_balance
##                                      3.826389e-05
```

```
coef.min <- coef(fit1.cv, s="lambda.min")
coef.min <- coef.min[which(coef.min !=0),]
coef.min[order(abs(coef.min), decreasing = TRUE)]
```

```
##                                       (Intercept)
##                                     -2.763833e+00
##                                    credit_mixGood
##                                     -8.414774e-01
## payment_behaviourLow_spent_Medium_value_payments
##                                      5.595028e-01
##                                 occupationWriter
```

```
##                                3.818317e-01
##                                  valuesmall
##                                3.815178e-01
##                           occupationDeveloper
##                                -3.701092e-01
##                              num_credit_card
##                                2.939359e-01
##                           occupationArchitect
##                                2.873645e-01
##                                    no_data1
##                                -2.160807e-01
##                                 payday_loan1
##                                2.063772e-01
##                       payment_of_min_amountYes
##                                2.010501e-01
##                           occupationScientist
##                                -1.823116e-01
##                            occupationEngineer
##                                1.822568e-01
##                               mortgage_loan1
##                                1.723600e-01
##                                 num_of_loan
##                                1.419718e-01
##                            occupationMechanic
##                                -1.224927e-01
##                                    spentlow
##                                1.150920e-01
##                              occupationDoctor
##                                -1.081813e-01
##                                interest_rate
##                                7.020056e-02
##                           delay_from_due_date
##                                6.348020e-02
##                             num_bank_accounts
##                                -4.012866e-02
##                            occupationMusician
##                                -3.804229e-02
##                               not_specified1
##                                3.021455e-02
##                        payment_of_min_amountNo
##                                -1.826428e-02
##                         num_of_delayed_payment
##                                -1.644384e-02
##                                       month
##                                -1.620208e-02
##                          credit_builder_loan1
##                                -1.410517e-02
##                          num_credit_inquiries
##                                1.295660e-02
##                                  auto_loan1
##                                9.071150e-03
##                       credit_utilization_ratio
##                                -3.713379e-03
##                                         age
```

```
##                                  -3.640749e-03
##                               credit_history_age
##                                    -2.345932e-03
##                              total_emi_per_month
##                                    -2.276768e-03
##                                 monthly_balance
##                                     1.605576e-03
##                           amount_invested_monthly
##                                    -1.475127e-03
##                                  outstanding_debt
##                                     2.776438e-04
##                                  annual_income
##                                    -9.760330e-07
```

After observing both of the models, it seems that we will use model.1se as the column in my final model, as model.min seems to be too complex.

```
final.model.1 <- glm(credit_score ~num_credit_card + interest_rate +num_of_loan + delay_from_due_date +
        payment_behaviour + delay_from_due_date + num_credit_inquiries + outstanding_debt +
        credit_history_age + payment_of_min_amount + total_emi_per_month + no_data +
        value,data = df_train, family = "binomial")
Anova(final.model.1)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: credit_score
##                         LR Chisq Df Pr(>Chisq)
## num_credit_card           67.411  1  < 2.2e-16 ***
## interest_rate             30.203  1  3.890e-08 ***
## num_of_loan                6.176  1   0.012946 *
## delay_from_due_date       76.172  1  < 2.2e-16 ***
## credit_mix                 4.581  2   0.101192
## payment_behaviour          8.115  3   0.043695 *
## num_credit_inquiries       0.324  1   0.569310
## outstanding_debt           7.446  1   0.006358 **
## credit_history_age         8.012  1   0.004646 **
## payment_of_min_amount      0.573  2   0.750824
## total_emi_per_month       18.221  1  1.967e-05 ***
## no_data                    1.614  1   0.203930
## value                          0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value for num_credit_inquries and value and payment_of_min_amount and outstanding_debt is larger than alpha = 0.05, so the result is not statistically significant and we can remove them to proceed the analysis.

```
final.model.2 <- glm(credit_score ~num_credit_card + interest_rate + num_of_loan + credit_mix +
        payment_behaviour + credit_history_age + total_emi_per_month +
        no_data,data = df_train, family = "binomial")
summary(final.model.2)
```

```
##
```

```
## Call:
## glm(formula = credit_score ~ num_credit_card + interest_rate +
##     num_of_loan + credit_mix + payment_behaviour + credit_history_age +
##     total_emi_per_month + no_data, family = "binomial", data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4210  -0.5842   0.1171   0.2737   2.3410
##
## Coefficients:
##                                                  Estimate Std. Error z value
## (Intercept)                                    -0.2505503  0.5660204  -0.443
## num_credit_card                                 0.3385716  0.0362195   9.348
## interest_rate                                   0.0864421  0.0129050   6.698
## num_of_loan                                     0.1707412  0.0538816   3.169
## credit_mixGood                                 -2.0957333  0.3872674  -5.412
## credit_mixStandard                             -1.0000662  0.3403651  -2.938
## payment_behaviourHigh_spent_Medium_value_payments  0.0016492  0.1995858   0.008
## payment_behaviourHigh_spent_Small_value_payments   0.2520871  0.2242751   1.124
## payment_behaviourLow_spent_Large_value_payments   -0.2623807  0.2432611  -1.079
## payment_behaviourLow_spent_Medium_value_payments   0.4955928  0.2125814   2.331
## payment_behaviourLow_spent_Small_value_payments    0.1247225  0.1950291   0.640
## credit_history_age                             -0.0036488  0.0008958  -4.073
## total_emi_per_month                            -0.0028174  0.0006521  -4.321
## no_data1                                       -0.1535508  0.2122042  -0.724
##                                                Pr(>|z|)
## (Intercept)                                     0.65802
## num_credit_card                                 < 2e-16 ***
## interest_rate                                  2.11e-11 ***
## num_of_loan                                     0.00153 **
## credit_mixGood                                 6.25e-08 ***
## credit_mixStandard                              0.00330 **
## payment_behaviourHigh_spent_Medium_value_payments  0.99341
## payment_behaviourHigh_spent_Small_value_payments   0.26101
## payment_behaviourLow_spent_Large_value_payments    0.28077
## payment_behaviourLow_spent_Medium_value_payments   0.01974 *
## payment_behaviourLow_spent_Small_value_payments    0.52249
## credit_history_age                             4.64e-05 ***
## total_emi_per_month                            1.55e-05 ***
## no_data1                                        0.46931
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3793.9  on 2891  degrees of freedom
## Residual deviance: 1816.9  on 2878  degrees of freedom
## AIC: 1844.9
##
## Number of Fisher Scoring iterations: 7
```

The above variables is associated with the credit score. Then we will get the coefficient.

```
summary(final.model.2)$coefficients[order(abs(summary(final.model.2)$coefficients[, "Estimate"]), decrea
```

```
##                                                    Estimate   Std. Error
## credit_mixGood                                 -2.095733345 0.3872673884
## credit_mixStandard                             -1.000066223 0.3403651019
## payment_behaviourLow_spent_Medium_value_payments  0.495592789 0.2125813720
## num_credit_card                                  0.338571581 0.0362194681
## payment_behaviourLow_spent_Large_value_payments  -0.262380678 0.2432611285
## payment_behaviourHigh_spent_Small_value_payments  0.252087112 0.2242751317
## (Intercept)                                    -0.250550299 0.5660204244
## num_of_loan                                      0.170741231 0.0538816239
## no_data1                                       -0.153550824 0.2122041961
## payment_behaviourLow_spent_Small_value_payments   0.124722493 0.1950291461
## interest_rate                                    0.086442126 0.0129050457
## credit_history_age                             -0.003648776 0.0008958219
## total_emi_per_month                            -0.002817439 0.0006520638
## payment_behaviourHigh_spent_Medium_value_payments  0.001649247 0.1995858360
##                                                     z value     Pr(>|z|)
## credit_mixGood                                 -5.411592630 6.246667e-08
## credit_mixStandard                             -2.938216103 3.301068e-03
## payment_behaviourLow_spent_Medium_value_payments  2.331308637 1.973709e-02
## num_credit_card                                  9.347778952 8.950750e-21
## payment_behaviourLow_spent_Large_value_payments  -1.078596813 2.807675e-01
## payment_behaviourHigh_spent_Small_value_payments  1.124008312 2.610095e-01
## (Intercept)                                    -0.442652400 6.580172e-01
## num_of_loan                                      3.168821181 1.530585e-03
## no_data1                                       -0.723599374 4.693117e-01
## payment_behaviourLow_spent_Small_value_payments   0.639506943 5.224932e-01
## interest_rate                                    6.698320031 2.108292e-11
## credit_history_age                             -4.073104050 4.639069e-05
## total_emi_per_month                            -4.320802252 1.554629e-05
## payment_behaviourHigh_spent_Medium_value_payments  0.008263346 9.934069e-01
```

From this graph, we can say people who set payment_of_min_amount to yes and have many credit cards will tend to have bad credit scores. What is more interesting is that people do not necessarily always need a reason to borrow the money if they have a higher credit score. This strange situation happens because the bank will only lend money to people with good credit scores with no provided reason. Now, we are going to test the accuracy of our model.

```
final.model.2 <- glm(credit_score ~num_credit_card + interest_rate + num_of_loan + credit_mix +
        payment_behaviour + credit_history_age + total_emi_per_month +
        no_data,data = df_train, family = "binomial")
y_pred <- predict(final.model.2, newdata = df_train, type = "response")
y_pred_binary <- ifelse(y_pred > 0.5, "Bad", "Good")
y_test <- df_train$credit_score
accuracy <- sum(y_pred_binary == y_test) / length(y_test)
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.3250346
```

It seems that the accuracy is really low. We believe that the neural network will fix the low accuracy problem, but logistic regression demonstrates to us the idea that "credit_mix", "payment_behaviour", and "num_of_loan" are predictor variables that are strongly associated with credit scores.

Principal Component Analysis (PCA)

We will now conduct PCA to identify and select our most important factors and remove the ones not needed.

```
# Conduct PCA and print summary
numeric_vars <- which(sapply(df_ml, is.numeric))
pca <- prcomp(df_ml[, numeric_vars], center = TRUE, scale. = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.6675 1.6343 1.02632 1.01176 0.95473 0.92883 0.89492
## Proportion of Variance 0.3953 0.1484 0.05852 0.05687 0.05064 0.04793 0.04449
## Cumulative Proportion  0.3953 0.5437 0.60220 0.65907 0.70970 0.75763 0.80213
##                           PC8     PC9    PC10    PC11    PC12   PC13    PC14
## Standard deviation     0.74132 0.68551 0.66787 0.62341 0.61717 0.5613 0.55102
## Proportion of Variance 0.03053 0.02611 0.02478 0.02159 0.02116 0.0175 0.01687
## Cumulative Proportion  0.83266 0.85876 0.88355 0.90514 0.92630 0.9438 0.96067
##                           PC15    PC16    PC17    PC18
## Standard deviation     0.50985 0.49945 0.44387 0.03909
## Proportion of Variance 0.01444 0.01386 0.01095 0.00008
## Cumulative Proportion  0.97511 0.98897 0.99992 1.00000
```

From the above results we can see that we can see the first 14-15 variables seem to have good Proportion of Variance along with a decent Standard Deviation. They also have a cumulative proportion of 0.96067.

```
# Visualize explained variance by each principal component
scree_plot <- function(pca_obj) {
  variance_pct <- (pca_obj$sdev^2) / sum(pca_obj$sdev^2)
  df_var <- data.frame(component = 1:length(variance_pct), variance = variance_pct)
  ggplot(df_var, aes(x = component, y = variance)) + geom_point() + geom_line(group = 1) +
    xlab("Principal Component") + ylab("Proportion of Explained Variance") +
    ggtitle("Scree Plot") + theme_minimal()
}
# scree_plot(pca)
```

We see the scree plot leveling off at 11. Thus, our PCA choice should be around that number.

```
# Calculate the cumulative proportion of variance explained
eigenvalues <- pca$sdev^2
variance_pct <- eigenvalues / sum(eigenvalues)
cumulative_variance_pct <- cumsum(variance_pct)
cumulative_variance_pct
```

```
##  [1] 0.3952945 0.5436764 0.6021951 0.6590652 0.7097042 0.7576335 0.8021273
##  [8] 0.8326579 0.8587647 0.8835456 0.9051369 0.9262981 0.9438018 0.9606697
## [15] 0.9751111 0.9889693 0.9999151 1.0000000
```

We see the 14th component has a cumulative variance of approximately 96%, which is good enough for our model. At 11, we only have approximately 91%, which is not acceptable. Thus, despite the scree plot leveling off at 11, the cumulative variance is not enough for us to accept only 11 components.

```r
# Create a scree plot of the cumulative proportion of variance explained
df_cum_var <- data.frame(component = 1:length(cumulative_variance_pct),
                         cumulative_variance = cumulative_variance_pct)

scree_plot_cumulative <- ggplot(df_cum_var, aes(x = component, y = cumulative_variance)) +
  geom_point(color = "blue") + geom_line(color = "blue", group = 1) +
  geom_hline(yintercept = 0.95, linetype = "dashed", color = "red") +
  labs(x = "Principal Components", y = "Cumulative Proportion of Variance Explained",
       title = "Scree Plot: Cumulative Proportion of Variance Explained") +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5))
# print(scree_plot_cumulative)
```

Here, we choose 14 as the optimal number of components.

```r
# Determine the optimal number of principal components and create a new dataframe
optimal_n <- which(cumulative_variance_pct >= 0.95)[1]
cat("Optimal number of principal components:", optimal_n)
```

```
## Optimal number of principal components: 14
```

```r
pca_df <- as.data.frame(pca$x[, 1:optimal_n])
```

```r
# 2D plot of the first two principal components
pca_2D <- ggplot(pca_df, aes(x = PC1, y = PC2)) + geom_point() + theme_minimal() +
  labs(x = "Principal Component 1", y = "Principal Component 2", title = "PCA 2D Plot")
# print(pca_2D)
```

Neural Network

It is quite complex to convert categorical variable into dummy variable in the neural net, so finally, we applied the fastDummies package in R to fix the problem.

```r
# create dummy variable
i <- sapply(df_ml, is.factor)
df_ml_X <- data.frame(df_ml)
df_ml_X[i] <- lapply(df_ml_X[i], as.character)
df_ml_X <- df_ml_X[,-which(names(df_ml_X) == "credit_score")]
df_ml_X <- fastDummies::dummy_cols(df_ml_X)
cat <- sapply(df_ml_X , is.character)
df_ml_X <- df_ml_X[,!cat]
data_xtrain <- as.matrix(df_ml_X[train_ind, ])
data_ytrain <- as.matrix(ifelse(df_train[,which(names(df_train) == "credit_score")] == "Good", 1, 0))

data_xtest <-as.matrix(df_ml_X[-train_ind, ])
data_ytest <-as.matrix(ifelse(df_test[,which(names(df_test) == "credit_score")] == "Good", 1, 0))
p <- dim(data_xtrain)[2] # number of input variables
model <- keras_model_sequential() %>%
        layer_dense(units = 64, activation = "relu", input_shape = c(p)) %>%
        layer_dense(units = 32, activation = "relu", input_shape = 64) %>%
        layer_dense(units = 1, activation = "sigmoid") # output
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_2 (Dense)                    (None, 64)                      4544
##  dense_1 (Dense)                    (None, 32)                      2080
##  dense (Dense)                      (None, 1)                       33
## ================================================================================
## Total params: 6,657
## Trainable params: 6,657
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model %>% compile(
        optimizer = optimizer_adam(), loss = "binary_crossentropy", metrics = c("accuracy"))
nn.fit1 <- model %>% fit(
        data_xtrain, data_ytrain, epochs = 30, batch_size = 128, validation_split = .15)
# plot(nn.fit1)
```

The best epoch in this graph seems to be around 10. Thus, we will retrain this model to finally test our model.

```
p <- dim(data_xtrain)[2] # number of input variables
model <- keras_model_sequential() %>%
        layer_dense(units = 64, activation = "relu", input_shape = c(p)) %>%
        layer_dense(units = 32, activation = "relu", input_shape = 64) %>%
        layer_dense(units = 16, activation = "relu", input_shape = 32) %>%
        layer_dense(units = 1, activation = "sigmoid") # output
model %>% compile(
        optimizer = optimizer_adam(), loss = "binary_crossentropy",
        metrics = c("accuracy")
)
model %>% fit(
        data_xtrain, data_ytrain, epochs = 10, batch_size = 128)
results <- model %>% evaluate(data_xtest, data_ytest) ;
```

After retraining we see that our accuracy becomes 0.8124352, which is much higher than the logistic regression, whose accuracy is around 0.35. However, compared to logistic regression, the drawback of the neural network is that we cannot understand which attributes will increase or decrease the credit score. Thus, we will use a decision tree to help us fix this problem.

Decision Trees

```
library(rpart)
library(rpart.plot)
library(randomForest)

# Create and visualize a decision tree
dt <- rpart(credit_score ~ ., data = df_train, method = "class")
# rpart.plot(dt)
```

```r
# Predict the credit scores on the test set and calculate accuracy
dt_pred <- predict(dt, newdata = df_test, type = "class")
dt_accuracy <- mean(dt_pred == df_test$credit_score)
cat("Decision Tree Accuracy:", dt_accuracy)
```

```
## Decision Tree Accuracy: 0.8611399
```

We see more progress with the decision trees. The accuracy of the decision tree model is approximately 0.87, giving us more accuracy than both the neural network and logistic regression models.

Random Forest

```r
# Create a random forest and print the model
rf <- randomForest(credit_score ~ ., data = df_train, ntree = 500)
print(rf)
```

```
##
## Call:
##  randomForest(formula = credit_score ~ ., data = df_train, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 11.17%
## Confusion matrix:
##      Good Poor class.error
## Good  952  102  0.09677419
## Poor  221 1617  0.12023939
```

```r
# Predict the credit scores on the test set and calculate accuracy
rf_pred <- predict(rf, newdata = df_test)
rf_accuracy <- mean(rf_pred == df_test$credit_score)
rf_accuracy
```

```
## [1] 0.8849741
```

The accuracy of the random tree model is approximately 0.89, giving us more accuracy than the decision tree, neural network, and logistic regression models.

Conclusion

- We established the first 14 components are good predictors of credit score, and that the three best predictor variables seem to be “credit_mix”, “payment_behaviour”, and “num_of_loan”.
- We see that the random forest model is the most effective at predicting credit score with an accuracy of approximately 0.89.
- We see that the decision tree model had an accuracy of approximately 0.87 and the random forest model had an accuracy of approximately 0.89, which are extremely close and are the two best models to use for predicting credit scores.