Project Creditworthiness

Dung Nguyen

Step 1: Business and Data Understanding

What decisions needs to be made?

Determining whether customers are creditworthy to give a loan to.

What data is needed to inform those decisions?

Outcome: Credit-Application-Result.

What kind of model (Continuous, Binary, Non-Binary, Time-Series) do we need to use to help make these decisions?

Binary model.

Step 2: Building the Training Set

After importing, the data set were formatted as suggested.

```
library(tidyverse)
library(caret)
library(randomForest)
library(bst)
library(cowplot)
#raw <- read.csv("./Creditworthiness/credit-data-training.csv", stringsAsFactors=F)
#validation <- read.csv("./Creditworthiness/customers-to-score.csv", stringsAsFactors=F)
colnames(validation) <- str_replace_all(colnames(validation), '\\.', '_')
colnames(raw) <- str_replace_all(colnames(raw), '\\.', '_')
glimpse(raw)
```

```
## Rows: 500
## Columns: 20
## $ Credit_Application_Result         <chr> "Creditworthy", "Creditworthy", "...
## $ Account_Balance                   <chr> "Some Balance", "Some Balance", "...
## $ Duration_of_Credit_Month          <int> 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, ...
## $ Payment_Status_of_Previous_Credit <chr> "Paid Up", "Paid Up", "No Problem...
## $ Purpose                           <chr> "Other", "Home Related", "Home Re...
## $ Credit_Amount                     <int> 1494, 1494, 1544, 3380, 343, 362,...
## $ Value_Savings_Stocks              <chr> "£100-£1000", "£100-£1000", "None...
## $ Length_of_current_employment      <chr> "< 1yr", "< 1yr", "1-4 yrs", "1-4...
## $ Instalment_per_cent               <int> 1, 1, 2, 1, 4, 4, 4, 3, 3, 2, 3, ...
```

```
## $ Guarantors                          <chr> "None", "None", "None", "None", "...
## $ Duration_in_Current_address         <int> 2, 2, 1, 1, 1, NA, NA, NA, 3, 4, ...
## $ Most_valuable_available_asset       <int> 1, 1, 1, 1, 1, 3, 2, 2, 1, 1, 1, ...
## $ Age_years                           <int> NA, 29, 42, 37, 27, 52, 24, 22, 2...
## $ Concurrent_Credits                  <chr> "Other Banks/Depts", "Other Banks...
## $ Type_of_apartment                   <int> 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, ...
## $ No_of_Credits_at_this_Bank          <chr> "1", "1", "More than 1", "1", "1"...
## $ Occupation                          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ No_of_dependents                    <int> 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, ...
## $ Telephone                           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ...
## $ Foreign_Worker                      <int> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
```

Identifying and imputing missing data

The number of missing values in each field:

```
isna = vector(length = length(raw))
i=1
for(col in raw){
  isna[i] = sum(is.na(col))
  i = i + 1 }
print(data.frame(field = colnames(raw), no_of_missing = isna))
```

```
##                                field no_of_missing
## 1          Credit_Application_Result             0
## 2                    Account_Balance             0
## 3             Duration_of_Credit_Month             0
## 4  Payment_Status_of_Previous_Credit             0
## 5                            Purpose             0
## 6                      Credit_Amount             0
## 7                Value_Savings_Stocks             0
## 8         Length_of_current_employment             0
## 9                  Instalment_per_cent             0
## 10                        Guarantors             0
## 11       Duration_in_Current_address           344
## 12     Most_valuable_available_asset             0
## 13                         Age_years            12
## 14                Concurrent_Credits             0
## 15                 Type_of_apartment             0
## 16        No_of_Credits_at_this_Bank             0
## 17                        Occupation             0
## 18                  No_of_dependents             0
## 19                         Telephone             0
## 20                    Foreign_Worker             0
```

`Duration_in_Current_address` has 344 missing values so I dropped it. `Age_years` has 12 missing values so I imputed it by median.

```
raw = raw[,!(colnames(raw) %in% c('Duration_in_Current_address'))]

median_age <- median(raw$Age_years[!is.na(raw$Age_years)])
raw$Age_years[is.na(raw$Age_years)] <- median_age
mean(raw$Age_years)
```

```
## [1] 35.574
```

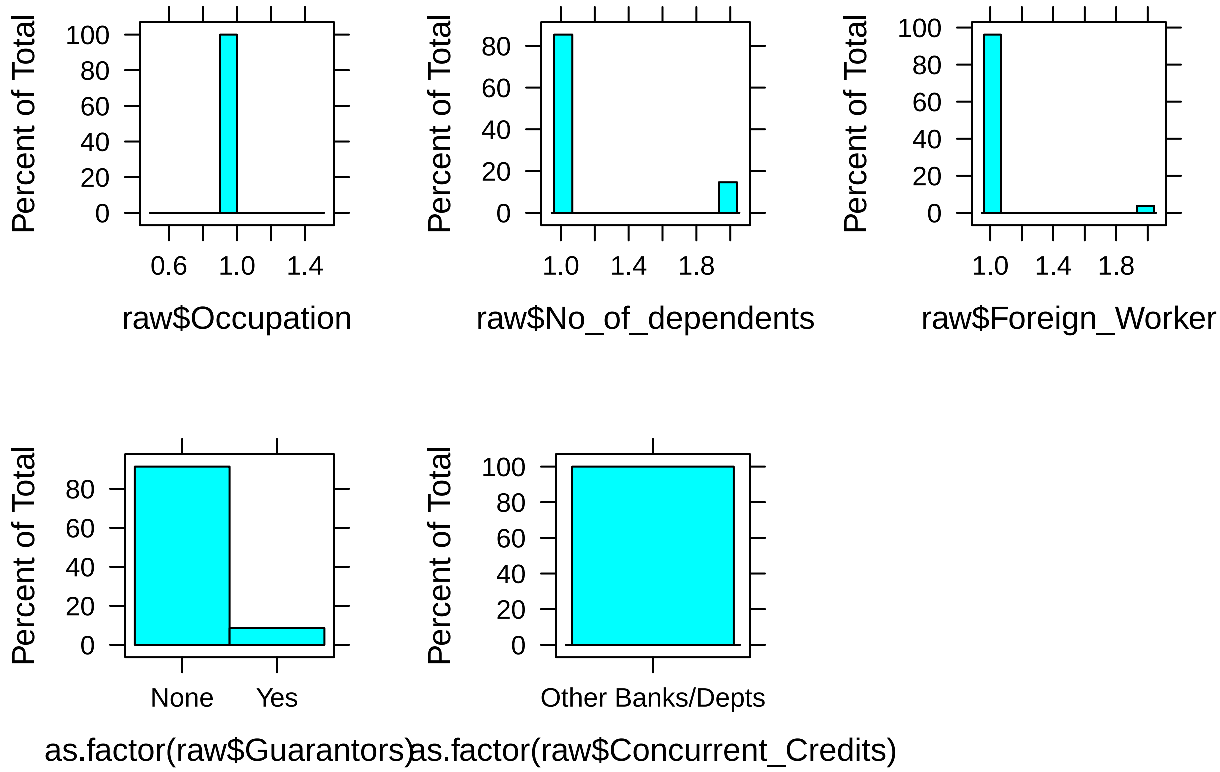
Identifying low-variability fields and removing them

Frequency table and histogram of low-variability fields:

```r
print(list(count(raw, Occupation),
     count(raw, No_of_dependents),
     count(raw, Foreign_Worker),
     count(raw, Guarantors),
     count(raw, Concurrent_Credits)))
```

```
## [[1]]
##   Occupation   n
## 1          1 500
##
## [[2]]
##   No_of_dependents   n
## 1                1 427
## 2                2  73
##
## [[3]]
##   Foreign_Worker   n
## 1              1 481
## 2              2  19
##
## [[4]]
##   Guarantors   n
## 1       None 457
## 2        Yes  43
##
## [[5]]
##   Concurrent_Credits   n
## 1  Other Banks/Depts 500
```

```r
p1 <- histogram(raw$Occupation)
p2 <- histogram(raw$No_of_dependents)
p3 <- histogram(raw$Foreign_Worker)
p4 <- histogram(as.factor(raw$Guarantors))
p5 <- histogram(as.factor(raw$Concurrent_Credits))
plot_grid(p1, p2, p3, p4, p5)
```



After removing those fields plus `Telephone`, no high correlation was detected among numeric fields:

```
numeric <-  c('Duration_of_Credit_Month',
              'Credit_Amount',
              'Age_years')
numeric2 <- c('Instalment_per_cent',
           'Most_valuable_available_asset',
           'Type_of_apartment')
factor <- c("Credit_Application_Result",
             "Account_Balance",
             "Payment_Status_of_Previous_Credit",
             "Purpose",
             "Value_Savings_Stocks",
             "Length_of_current_employment",
             "No_of_Credits_at_this_Bank")

raw <- data.frame(raw[c(numeric, numeric2)],
                     apply(raw[factor],2, as.factor))

print(cor(raw[c(numeric, numeric2)]))
```

```
##                                Duration_of_Credit_Month Credit_Amount
## Duration_of_Credit_Month                     1.00000000    0.57397971
## Credit_Amount                                0.57397971    1.00000000
## Age_years                                   -0.06419695    0.06931589
## Instalment_per_cent                          0.06810553   -0.28885153
```

```
## Most_valuable_available_asset              0.29985487    0.32554538
## Type_of_apartment                          0.15251629    0.17007119
##                                Age_years Instalment_per_cent
## Duration_of_Credit_Month      -0.06419695          0.06810553
## Credit_Amount                  0.06931589         -0.28885153
## Age_years                      1.00000000          0.03926967
## Instalment_per_cent            0.03926967          1.00000000
## Most_valuable_available_asset  0.08623342          0.08149260
## Type_of_apartment              0.32935038          0.07453322
##                               Most_valuable_available_asset Type_of_apartment
## Duration_of_Credit_Month                         0.29985487        0.15251629
## Credit_Amount                                    0.32554538        0.17007119
## Age_years                                        0.08623342        0.32935038
## Instalment_per_cent                              0.08149260        0.07453322
## Most_valuable_available_asset                    1.00000000        0.37310079
## Type_of_apartment                                0.37310079        1.00000000
```

The final data set has 13 columns

```
raw <- raw %>% mutate(Credit_Application_Result= as.factor(Credit_Application_Result),
                           Account_Balance= as.factor(Account_Balance),
                           Payment_Status_of_Previous_Credit=as.factor(Payment_Status_of_Previous_Credi
                           Purpose=as.factor(Purpose),
                           Value_Savings_Stocks=as.factor(Value_Savings_Stocks),
                           Length_of_current_employment=as.factor(Length_of_current_employment),
                           No_of_Credits_at_this_Bank=as.factor(No_of_Credits_at_this_Bank))
data <- raw
glimpse(data)
```

```
## Rows: 500
## Columns: 13
## $ Duration_of_Credit_Month         <int> 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, ...
## $ Credit_Amount                    <int> 1494, 1494, 1544, 3380, 343, 362,...
## $ Age_years                        <dbl> 33, 29, 42, 37, 27, 52, 24, 22, 2...
## $ Instalment_per_cent              <int> 1, 1, 2, 1, 4, 4, 4, 3, 3, 2, 3, ...
## $ Most_valuable_available_asset    <int> 1, 1, 1, 1, 1, 3, 2, 2, 1, 1, 1, ...
## $ Type_of_apartment                <int> 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, ...
## $ Credit_Application_Result        <fct> Creditworthy, Creditworthy, Credi...
## $ Account_Balance                  <fct> Some Balance, Some Balance, Some ...
## $ Payment_Status_of_Previous_Credit <fct> Paid Up, Paid Up, No Problems (in...
## $ Purpose                          <fct> Other, Home Related, Home Related...
## $ Value_Savings_Stocks             <fct> £100-£1000, £100-£1000, None, Non...
## $ Length_of_current_employment     <fct> < 1yr, < 1yr, 1-4 yrs, 1-4 yrs, <...
## $ No_of_Credits_at_this_Bank       <fct> 1, 1, More than 1, 1, 1, More tha...
```

Step 3: Train your Classification Models

Create Estimation and Validation samples

```
data$worthy <- as.factor(ifelse(data$Credit_Application_Result == 'Creditworthy', 1, 0))
data$nonworthy <- as.factor(ifelse(data$Credit_Application_Result == 'Non-Creditworthy', 1, 0))
```

```r
set.seed(1)
train.index <- createDataPartition(data$Credit_Application_Result, p = .7, list = FALSE)
train <- data[train.index,]
test <- data[-train.index,]
```

Confusion matrices

```r
# Logistic regression
lr <- vector()
model1 <- train(Credit_Application_Result~.,
                data=train[!colnames(train) %in% c('worthy', 'nonworthy')],
                method='glm')  #check
y_hat1 <- predict(model1, test)
y_hat10 <- predict(model1, train)
lr['Accuracy_train'] <- confusionMatrix(data = y_hat10, reference = train$Credit_Application_Result)$ove
lr['Accuracy_test'] <- confusionMatrix(data = y_hat1, reference = test$Credit_Application_Result)$overal

model1w <- train(worthy~.,
                 data=train[!colnames(train) %in% c('Credit_Application_Result', 'nonworthy')],
                 method='glm')
y_hat1w <- predict(model1w, test)
lr['Accyracy_worthy'] <- confusionMatrix(data = y_hat1w, reference = test$worthy)$overall["Accuracy"]

model1n <- train(nonworthy~.,
                 data=train[!colnames(train) %in% c('Credit_Application_Result', 'worthy')],
                 method='glm')
y_hat1n <- predict(model1n, test)
lr['Accuracy_nonworthy'] <- confusionMatrix(data = y_hat1n, reference = test$nonworthy)$overall["Accura

vi1 <- varImp(model1, scale = F)   #pvalue
plot1 <- ggplot(vi1, top = dim(vi1$importance)[1]) +
  ggtitle('Variable Importance - Logistic Regression')

# Decision tree
tree <- vector()
model2 <- train(Credit_Application_Result~.,
                data=train[!colnames(train) %in% c('worthy', 'nonworthy')],
                method='rpart')
y_hat2 <- predict(model2, test)
y_hat20 <- predict(model2, train)
tree['Accuracy_train'] <- confusionMatrix(data = y_hat20, reference = train$Credit_Application_Result)$
tree['Accuracy_test'] <- confusionMatrix(data = y_hat2, reference = test$Credit_Application_Result)$ove

model2w <- train(worthy~.,
                 data=train[!colnames(train) %in% c('Credit_Application_Result', 'nonworthy')],
                 method='rpart')
y_hat2w <- predict(model2w, test)
tree['Accuracy_worthy'] <- confusionMatrix(data = y_hat2w, reference = test$worthy)$overall["Accuracy"]

model2n <- train(nonworthy~.,
                 data=train[!colnames(train) %in% c('Credit_Application_Result', 'worthy')],
```

```r
                 method='rpart')
y_hat2n <- predict(model2n, test)
tree['Accuracy_nonworthy'] <- confusionMatrix(data = y_hat2n, reference = test$nonworthy)$overall["Accu

vi2 <- varImp(model2, scale = F)
plot2 <- ggplot(vi2, top = dim(vi2$importance)[1]) +
  ggtitle('Variable Importance - Decision Tree')

# Forest model
forest <- vector()
set.seed(1)
model3 <- train(Credit_Application_Result~.,
                data=train[!colnames(train) %in% c('worthy', 'nonworthy')],
                method='rf',
                ntree=500)
y_hat3 <- predict(model3, test)
y_hat30 <- predict(model3, train)
forest['Accuracy_train'] <- confusionMatrix(data = y_hat30, reference = train$Credit_Application_Result
forest['Accuracy_test'] <- confusionMatrix(data = y_hat3, reference = test$Credit_Application_Result)$o

set.seed(1)
model3w <- train(worthy~.,
                 data=train[!colnames(train) %in% c('Credit_Application_Result', 'nonworthy')],
                 method='rf',
                 ntree=500)
y_hat3w <- predict(model3w, test)
forest['Accuracy_worthy'] <- confusionMatrix(data = y_hat3w, reference = test$worthy)$overall["Accuracy

set.seed(1)
model3n <- train(nonworthy~.,
                 data=train[!colnames(train) %in% c('Credit_Application_Result', 'worthy')],
                 method='rf',
                 ntree=500)
y_hat3n <- predict(model3n, test)
forest['Accuracy_nonworthy'] <- confusionMatrix(data = y_hat3n, reference = test$nonworthy)$overall["Ac

vi3 <- varImp(model3, scale = F)   #pvalue
plot3 <- ggplot(vi3, top = dim(vi3$importance)[1]) +
  ggtitle('Variable Importance - Random Forest')

# Boosted model - Boosted Classification Trees
boosted <- vector()
# Data have to be re-formatted for bst() function
train4 <- train %>% mutate(#Credit_Application_Result= as.numeric(Credit_Application_Result),
  Account_Balance= as.factor(as.numeric(Account_Balance)),
  Payment_Status_of_Previous_Credit=as.factor(as.numeric(Payment_Status_of_Previous_Credit)),
  Purpose=as.factor(as.numeric(Purpose)),
  Value_Savings_Stocks=as.factor(as.numeric(Value_Savings_Stocks)),
  Length_of_current_employment=as.factor(as.numeric(Length_of_current_employment)),
  No_of_Credits_at_this_Bank=as.factor(as.numeric(No_of_Credits_at_this_Bank)),
  )

test4 <- test %>% mutate(#Credit_Application_Result= as.numeric(Credit_Application_Result),
```

```r
  Account_Balance= as.factor(as.numeric(Account_Balance)),
  Payment_Status_of_Previous_Credit=as.factor(as.numeric(Payment_Status_of_Previous_Credit)),
  Purpose=as.factor(as.numeric(Purpose)),
  Value_Savings_Stocks=as.factor(as.numeric(Value_Savings_Stocks)),
  Length_of_current_employment=as.factor(as.numeric(Length_of_current_employment)),
  No_of_Credits_at_this_Bank=as.factor(as.numeric(No_of_Credits_at_this_Bank)),
  )
control <- trainControl(method = "cv", number = 5)
set.seed(1)
model4 <- train(Credit_Application_Result~.,
                data=train4[!colnames(train) %in% c('worthy', 'nonworthy')],
                method="bstTree",
                trControl = control)
y_hat4 <- predict(model4, test4)
y_hat40 <- predict(model4, train4)
boosted['Accuracy_train'] <- confusionMatrix(data = y_hat40, reference = train4$Credit_Application_Resul
boosted['Accuracy_test'] <- confusionMatrix(data = y_hat4, reference = test4$Credit_Application_Result)$

set.seed(1)
model4w <- train(worthy~.,
                 data=train4[!colnames(train) %in% c('Credit_Application_Result', 'nonworthy')],
                 method='bstTree',
                 trControl = control)
y_hat4w <- predict(model4w, test4)
boosted['Accuracy_worthy'] <- confusionMatrix(data = y_hat4w, reference = test4$worthy)$overall["Accura

set.seed(1)
model4n <- train(nonworthy~.,
                 data=train4[!colnames(train) %in% c('Credit_Application_Result', 'worthy')],
                 method='bstTree',
                 trControl = control)
y_hat4n <- predict(model4n, test4)
boosted['Accuracy_nonworthy'] <- confusionMatrix(data = y_hat4n, reference = test4$nonworthy)$overall["

vi4 <- varImp(model4, scale = F)   #pvalue
plot4 <- ggplot(vi4, top = dim(vi4$importance)[1]) +
  ggtitle('Variable Importance - Boosted Tree')

#Print confusion matrices
print(confusionMatrix(data = y_hat1, reference = test$Credit_Application_Result))
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction           Creditworthy Non-Creditworthy
##   Creditworthy                104               22
##   Non-Creditworthy              3               20
##
##                Accuracy : 0.8322
##                  95% CI : (0.7624, 0.8884)
##     No Information Rate : 0.7181
##     P-Value [Acc > NIR] : 0.0008381
##
```

```
##                  Kappa : 0.5195
##
##  Mcnemar's Test P-Value : 0.0003182
##
##             Sensitivity : 0.9720
##             Specificity : 0.4762
##          Pos Pred Value : 0.8254
##          Neg Pred Value : 0.8696
##              Prevalence : 0.7181
##          Detection Rate : 0.6980
##    Detection Prevalence : 0.8456
##       Balanced Accuracy : 0.7241
##
##        'Positive' Class : Creditworthy
##
```

```r
print(confusionMatrix(data = y_hat2, reference = test$Credit_Application_Result))
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         Creditworthy Non-Creditworthy
##   Creditworthy              103               23
##   Non-Creditworthy            4               19
##
##                Accuracy : 0.8188
##                  95% CI : (0.7474, 0.8771)
##     No Information Rate : 0.7181
##     P-Value [Acc > NIR] : 0.003077
##
##                   Kappa : 0.4811
##
##  Mcnemar's Test P-Value : 0.000532
##
##             Sensitivity : 0.9626
##             Specificity : 0.4524
##          Pos Pred Value : 0.8175
##          Neg Pred Value : 0.8261
##              Prevalence : 0.7181
##          Detection Rate : 0.6913
##    Detection Prevalence : 0.8456
##       Balanced Accuracy : 0.7075
##
##        'Positive' Class : Creditworthy
##
```

```r
print(confusionMatrix(data = y_hat3, reference = test$Credit_Application_Result))
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         Creditworthy Non-Creditworthy
##   Creditworthy              106               32
```

```
##   Non-Creditworthy            1               10
## 
##                Accuracy : 0.7785
##                  95% CI : (0.7033, 0.8424)
##     No Information Rate : 0.7181
##     P-Value [Acc > NIR] : 0.05832
## 
##                   Kappa : 0.2949
## 
##  Mcnemar's Test P-Value : 1.767e-07
## 
##             Sensitivity : 0.9907
##             Specificity : 0.2381
##          Pos Pred Value : 0.7681
##          Neg Pred Value : 0.9091
##              Prevalence : 0.7181
##          Detection Rate : 0.7114
##    Detection Prevalence : 0.9262
##       Balanced Accuracy : 0.6144
## 
##        'Positive' Class : Creditworthy
## 
```

```r
print(confusionMatrix(data = y_hat4, reference = test4$Credit_Application_Result))
```

```
## Confusion Matrix and Statistics
## 
##                   Reference
## Prediction          Creditworthy Non-Creditworthy
##   Creditworthy               102               23
##   Non-Creditworthy             5               19
## 
##                Accuracy : 0.8121
##                  95% CI : (0.74, 0.8713)
##     No Information Rate : 0.7181
##     P-Value [Acc > NIR] : 0.005532
## 
##                   Kappa : 0.4664
## 
##  Mcnemar's Test P-Value : 0.001315
## 
##             Sensitivity : 0.9533
##             Specificity : 0.4524
##          Pos Pred Value : 0.8160
##          Neg Pred Value : 0.7917
##              Prevalence : 0.7181
##          Detection Rate : 0.6846
##    Detection Prevalence : 0.8389
##       Balanced Accuracy : 0.7028
## 
##        'Positive' Class : Creditworthy
## 
```

Overall accuracy

Among the 4 models, Logistic regression has the highest overall accuracy agains the Validation set (0.83). The bias can be seen in Logistic regression, Random forest, and Boosted tree, since accuracy against train set is lower than validation test for those models.

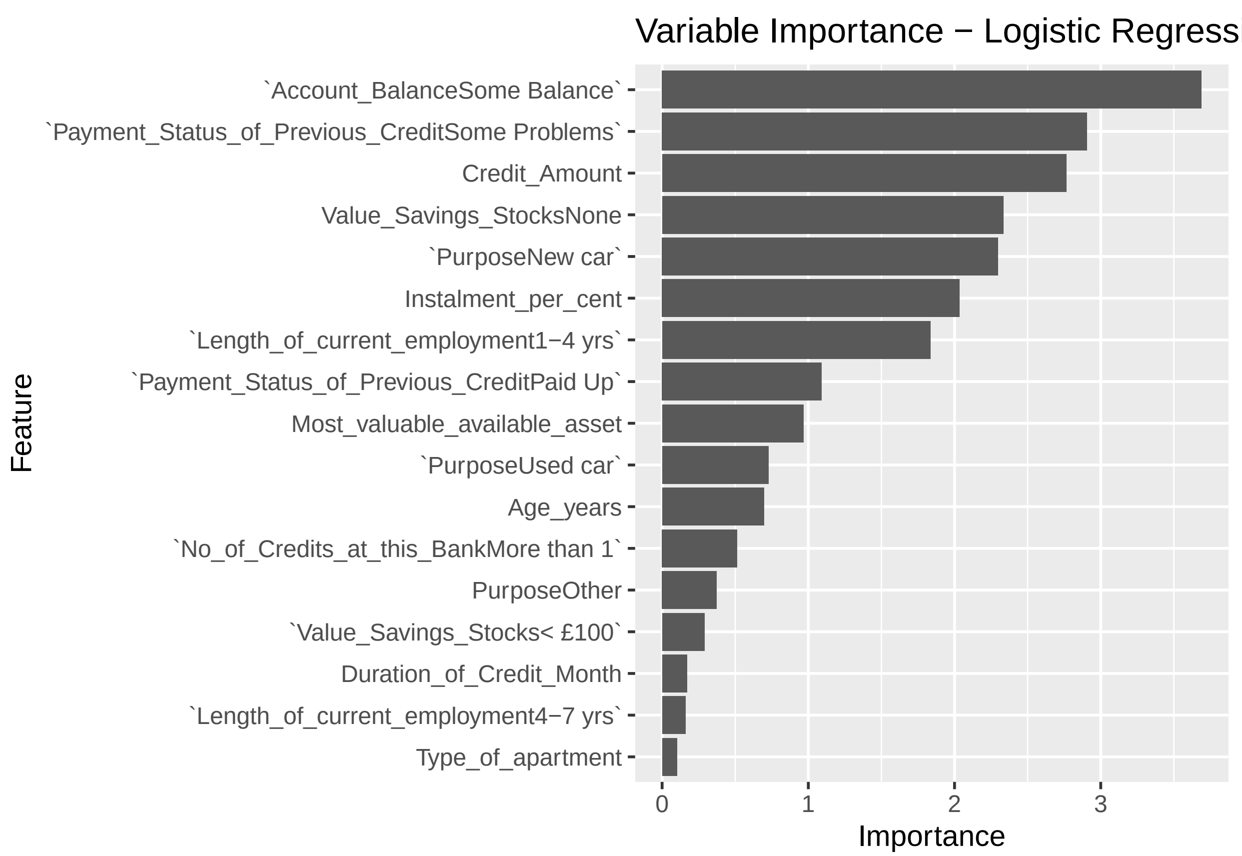
```
print(data.frame(lr, tree, forest, boosted)[c('Accuracy_train','Accuracy_test'),])
```

```
##                       lr      tree    forest   boosted
## Accuracy_train 0.7692308 0.7806268 0.8860399 0.7948718
## Accuracy_test  0.8322148 0.8187919 0.7785235 0.8120805
```

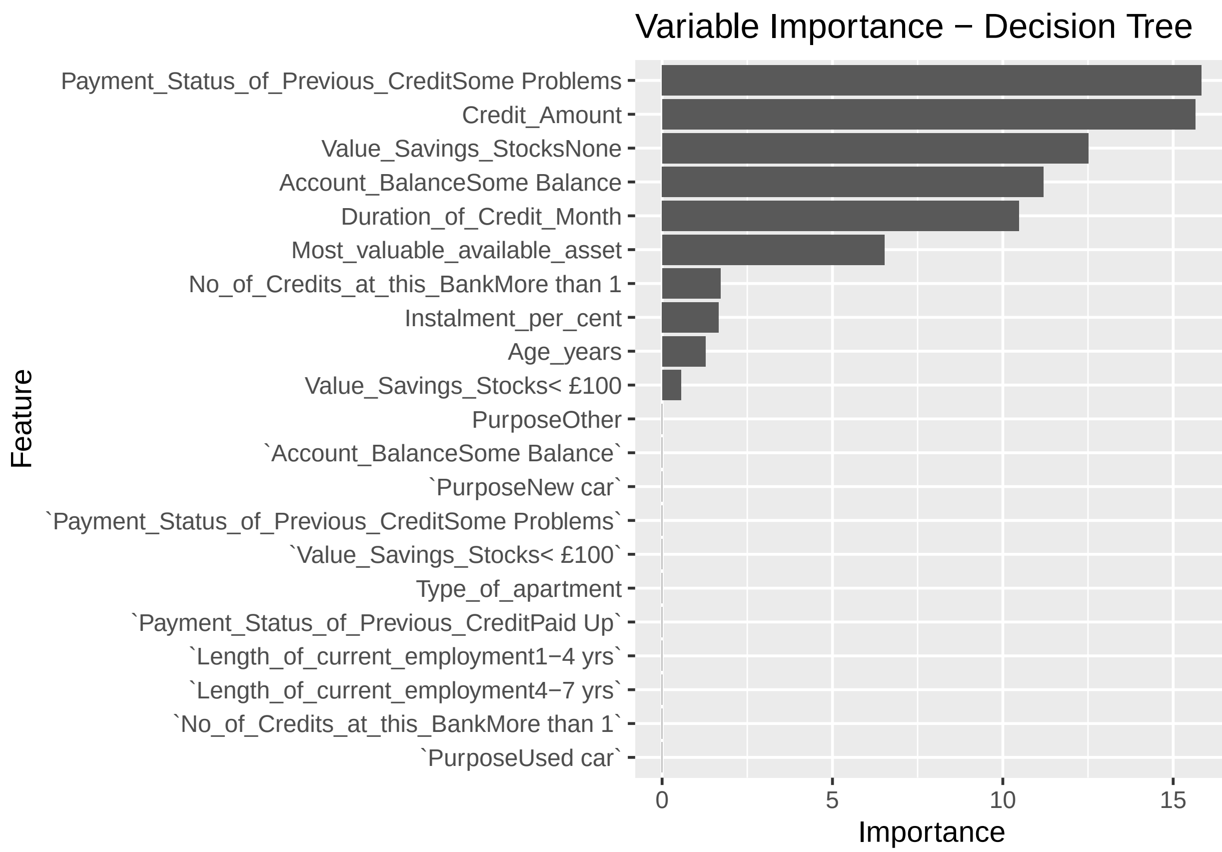
Plot variable importance

Most important predictors: - For Logistic regression: Account_Balance, Payment_Status_of_Previous_Credit, Credit_Amoun, Value_Savings_Stocks, Purpose, Instalment_per_cent. - For Decision tree: Payment_Status_of_Previous_Credit, Credit_Amount, Value_Savings_Stocks, Account_Balance, Duration_of_Credit_Month. - For Random forest: Credit_Amount, Age_years, Duration_of_Credit_Month. - For Boosted tree: Account_Balance, Value_Savings_Stocks, Paymen_Status_of_Previous_Credit, and Duration_of_Credit_Month are the 4 most important variables. The difference is not remarkable among the fields.

```
plot1
```

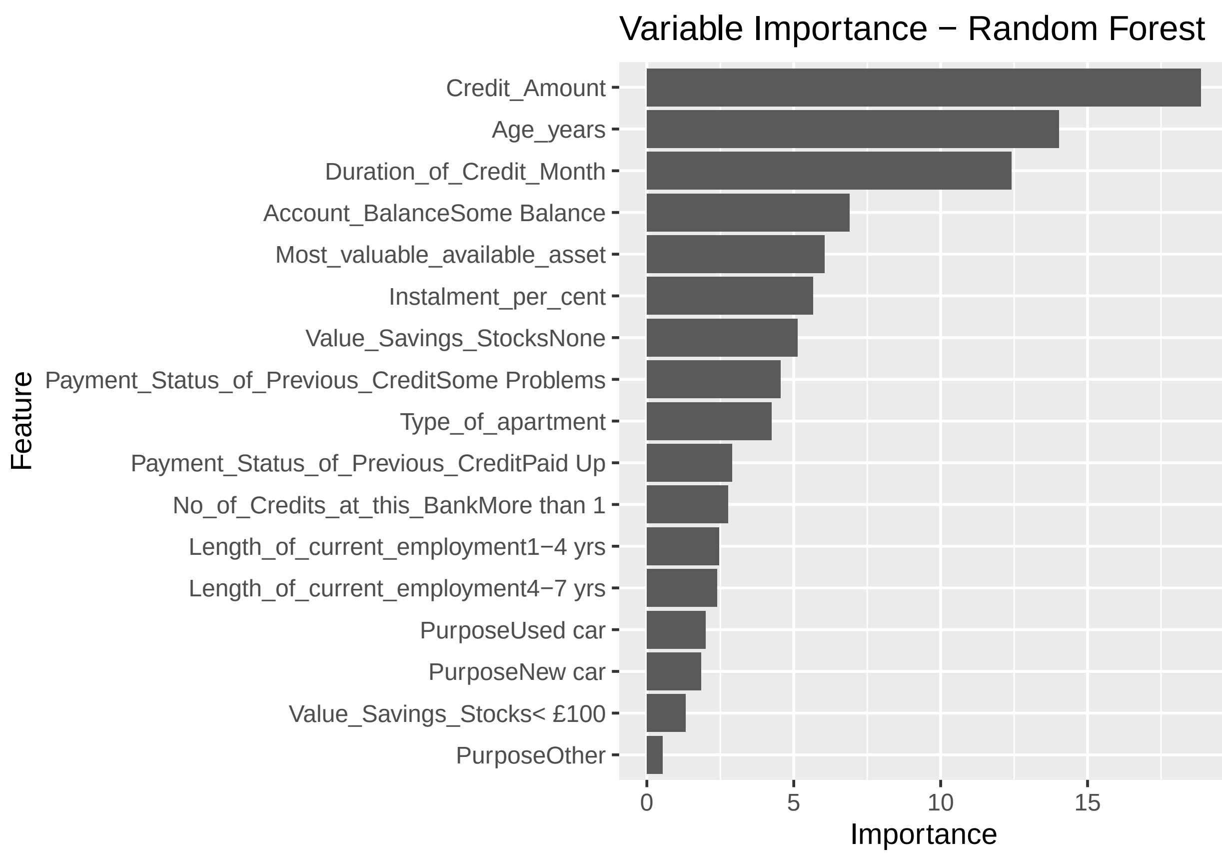


```
plot2
```

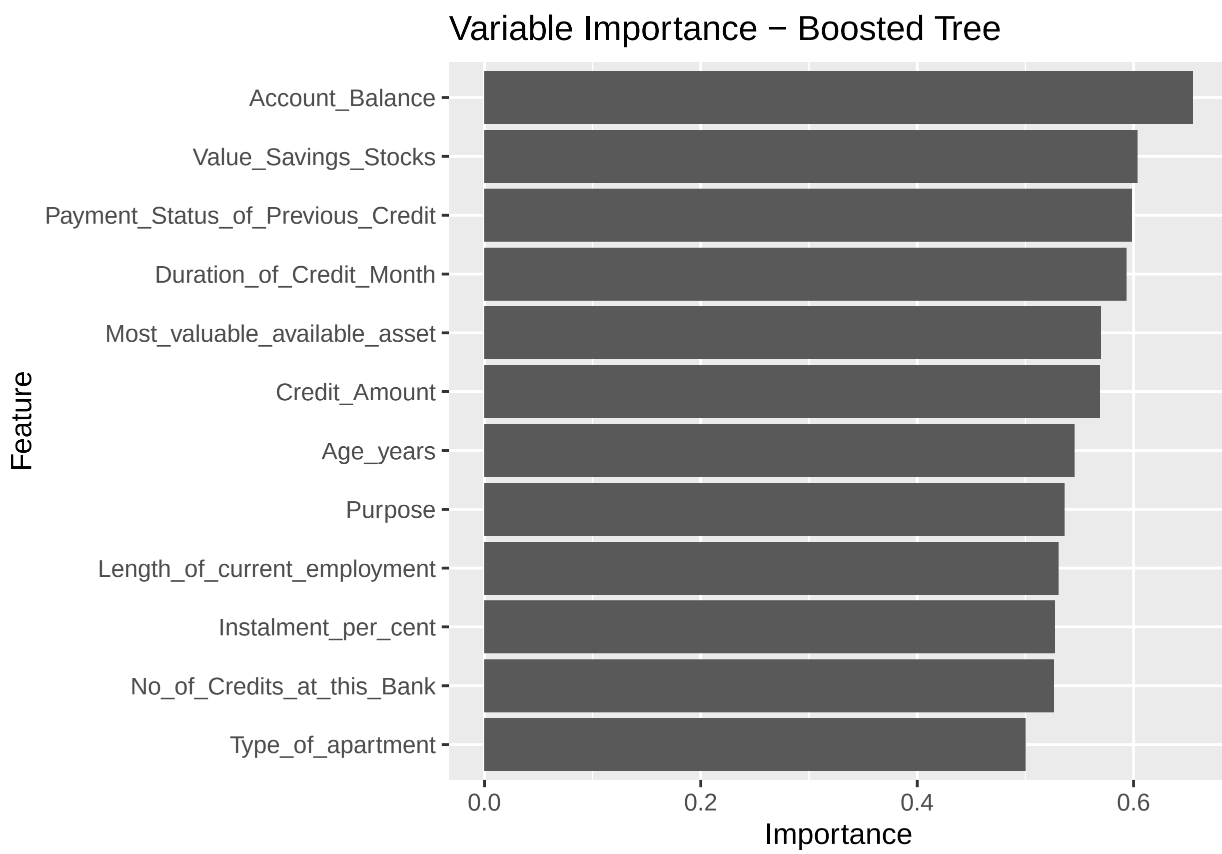


Variable Importance − Decision Tree
Payment_Status_of_Previous_CreditSome Problems
Credit_Amount
Value_Savings_StocksNone
Account_BalanceSome Balance
Duration_of_Credit_Month
Most_valuable_available_asset
No_of_Credits_at_this_BankMore than 1
Instalment_per_cent
Age_years
Value_Savings_Stocks< £100
PurposeOther
`Account_BalanceSome Balance`
`PurposeNew car`
`Payment_Status_of_Previous_CreditSome Problems`
`Value_Savings_Stocks< £100`
Type_of_apartment
`Payment_Status_of_Previous_CreditPaid Up`
`Length_of_current_employment1−4 yrs`
`Length_of_current_employment4−7 yrs`
`No_of_Credits_at_this_BankMore than 1`
`PurposeUsed car`
Feature
0
5
10
15
Importance


```
plot3
```


Variable Importance − Random Forest
Credit_Amount
Age_years
Duration_of_Credit_Month
Account_BalanceSome Balance
Most_valuable_available_asset
Instalment_per_cent
Value_Savings_StocksNone
Payment_Status_of_Previous_CreditSome Problems
Type_of_apartment
Payment_Status_of_Previous_CreditPaid Up
No_of_Credits_at_this_BankMore than 1
Length_of_current_employment1−4 yrs
Length_of_current_employment4−7 yrs
PurposeUsed car
PurposeNew car
Value_Savings_Stocks< £100
PurposeOther
Feature
0
5
10
15
Importance

```
plot4
```



Step 4: Writeup

Accuracy table

```
print(data.frame(lr, tree, forest, boosted))
```

```
##                           lr      tree    forest   boosted
## Accuracy_train     0.7692308 0.7806268 0.8860399 0.7948718
## Accuracy_test      0.8322148 0.8187919 0.7785235 0.8120805
## Accyracy_worthy    0.8322148 0.8187919 0.7718121 0.8120805
## Accuracy_nonworthy 0.8322148 0.7181208 0.7785235 0.8120805
```

Plot ROCs

```
par(pty = "s")
roc(as.numeric(test$Credit_Application_Result),
    as.numeric(y_hat1), plot=TRUE,
    legacy.axes=TRUE, percent=TRUE,
    xlab="False Positive Percentage",
    ylab="True Postive Percentage",
    print.auc=TRUE,
    print.auc.x=45,
    )
```

```
## Error in roc(as.numeric(test$Credit_Application_Result), as.numeric(y_hat1), : could not find functio
```

```r
legend("bottomright", legend=c("Logisitic Regression"))
```

```
## Error in strwidth(legend, units = "user", cex = cex, font = text.font): plot.new has not been called
```

```r
plot.roc(as.numeric(test$Credit_Application_Result),
         as.numeric(y_hat2),
         legacy.axes=TRUE, percent=TRUE,
         xlab="False Positive Percentage",
         ylab="True Postive Percentage",
         print.auc=TRUE,
         print.auc.x=45,
)
```

```
## Error in plot.roc(as.numeric(test$Credit_Application_Result), as.numeric(y_hat2), : could not find fu
```

```r
legend("bottomright", legend=c("Decision Tree"))
```

```
## Error in strwidth(legend, units = "user", cex = cex, font = text.font): plot.new has not been called
```

```r
plot.roc(as.numeric(test$Credit_Application_Result),
         as.numeric(y_hat3),
         legacy.axes=TRUE, percent=TRUE,
         xlab="False Positive Percentage",
         ylab="True Postive Percentage",
         print.auc=TRUE,
         print.auc.x=45,
)
```

```
## Error in plot.roc(as.numeric(test$Credit_Application_Result), as.numeric(y_hat3), : could not find fu
```

```r
legend("bottomright", legend=c("Random Forest"))
```

```
## Error in strwidth(legend, units = "user", cex = cex, font = text.font): plot.new has not been called
```

```r
plot.roc(as.numeric(test$Credit_Application_Result),
         as.numeric(y_hat4),
         legacy.axes=TRUE, percent=TRUE,
         xlab="False Positive Percentage",
         ylab="True Postive Percentage",
         print.auc=TRUE,
         print.auc.x=45,
)
```

```
## Error in plot.roc(as.numeric(test$Credit_Application_Result), as.numeric(y_hat4), : could not find fu
```

```r
legend("bottomright", legend=c("Boosted Tree"))
```

```
## Error in strwidth(legend, units = "user", cex = cex, font = text.font): plot.new has not been called
```

Logistic regresson model has the highest overall accuracy against the Validation set, as well as the highest accuracy within segments. It is also has the most optimal ROC. Therefore, I choose to use logistic regression model. Applying the chosen model, the number of individuals are creditworthy is 413.

```r
yhat_final <- predict(model1, validation)
print(sum(yhat_final=='Creditworthy'))
```

```
## [1] 413
```