Project 1: Determining Financial Risk of Stark Industries Banking Clients

Cam Nguyen, Kalina Gavrilova, Lauren Louie

2022-12-01

Problem Description

Stark Industries is a new banking company that has hired our team to create a machine learning algorithm that can successfully evaluate the risk-level of their potential clients based on data from their existing clients, and whether those existing clients were able to repay their loans to Stark Industries or not.

Objective

Our Objective is to prepare two different KNN models that can be used to predict the risk-level of a set of new clients based on their other data. Then, comparing the accuracy (including the sensitivity and specificity) of the two models to select the best one and use it to generate a prediction from a new set of client data.

Data Description

The data provided to us included 66 possible predictors for whether a client would be able to repay their loan or not- some of these were more explicit financial indicators, like the client's income or the material amount of the loan, while others were more indirect, like the number of dependents the client had, how old the client was, or whether they could be reached by the bank via telephone call. The target variable in this data set was represented as a categorical variable that either had a value of 0 (the client was not high risk) or 1 (the client was high-risk). There were 30,000 total observations in the data set.

Data Preperation

```
# Loading Libraries

library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
#install.packages("janitor", dependencies = TRUE)
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
# Loading Data

credit <- read.csv("credit_fa2022_8.csv", header = TRUE)
colnames(credit)
```

```
##  [1] "X"                            "SK_ID_CURR"
##  [3] "TARGET"                       "NAME_CONTRACT_TYPE"
##  [5] "CODE_GENDER"                  "FLAG_OWN_CAR"
##  [7] "FLAG_OWN_REALTY"              "CNT_CHILDREN"
##  [9] "AMT_INCOME_TOTAL"             "AMT_CREDIT"
## [11] "AMT_ANNUITY"                  "AMT_GOODS_PRICE"
## [13] "NAME_TYPE_SUITE"              "NAME_INCOME_TYPE"
## [15] "NAME_EDUCATION_TYPE"          "NAME_FAMILY_STATUS"
## [17] "NAME_HOUSING_TYPE"            "DAYS_BIRTH"
## [19] "DAYS_EMPLOYED"                "DAYS_REGISTRATION"
## [21] "DAYS_ID_PUBLISH"              "OWN_CAR_AGE"
## [23] "FLAG_MOBIL"                   "FLAG_EMP_PHONE"
## [25] "FLAG_WORK_PHONE"              "FLAG_CONT_MOBILE"
## [27] "FLAG_PHONE"                   "FLAG_EMAIL"
## [29] "OCCUPATION_TYPE"              "CNT_FAM_MEMBERS"
## [31] "REGION_RATING_CLIENT"         "REGION_RATING_CLIENT_W_CITY"
## [33] "WEEKDAY_APPR_PROCESS_START"   "HOUR_APPR_PROCESS_START"
## [35] "REG_REGION_NOT_LIVE_REGION"   "REG_REGION_NOT_WORK_REGION"
## [37] "LIVE_REGION_NOT_WORK_REGION" "REG_CITY_NOT_LIVE_CITY"
## [39] "REG_CITY_NOT_WORK_CITY"       "LIVE_CITY_NOT_WORK_CITY"
## [41] "ORGANIZATION_TYPE"            "DAYS_LAST_PHONE_CHANGE"
## [43] "FLAG_DOCUMENT_2"              "FLAG_DOCUMENT_3"
## [45] "FLAG_DOCUMENT_4"              "FLAG_DOCUMENT_5"
## [47] "FLAG_DOCUMENT_6"              "FLAG_DOCUMENT_7"
## [49] "FLAG_DOCUMENT_8"              "FLAG_DOCUMENT_9"
## [51] "FLAG_DOCUMENT_10"             "FLAG_DOCUMENT_11"
## [53] "FLAG_DOCUMENT_12"             "FLAG_DOCUMENT_13"
## [55] "FLAG_DOCUMENT_14"             "FLAG_DOCUMENT_15"
## [57] "FLAG_DOCUMENT_16"             "FLAG_DOCUMENT_17"
## [59] "FLAG_DOCUMENT_18"             "FLAG_DOCUMENT_19"
## [61] "FLAG_DOCUMENT_20"             "FLAG_DOCUMENT_21"
## [63] "AMT_REQ_CREDIT_BUREAU_HOUR"   "AMT_REQ_CREDIT_BUREAU_DAY"
## [65] "AMT_REQ_CREDIT_BUREAU_WEEK"   "AMT_REQ_CREDIT_BUREAU_MON"
## [67] "AMT_REQ_CREDIT_BUREAU_QRT"    "AMT_REQ_CREDIT_BUREAU_YEAR"
```

Model 1

```
# Removing Unnecessary Variables and Null Values
credit_m1 <- credit[ , -c(1,2, 5, 11, 13, 15, 20:22, 23:25, 27:28, 31:68)]

credit_m1 <- credit_m1[complete.cases(credit_m1),]

# Factorizing Categorical Variables

# colnames(credit_m1)
# str(credit_m1)
credit_m1[, c(1:4, 9:11, 14:15)] <- lapply(credit_m1[, c(1:4, 9:11, 14:15)], as.factor)


# Training/Validation Split

set.seed(666)

train_index_m1 <- sample(1:nrow(credit_m1), 0.6 * nrow(credit_m1))
valid_index_m1 <- setdiff(1:nrow(credit_m1), train_index_m1)

train_m1 <- credit_m1[train_index_m1, ]
valid_m1 <- credit_m1[valid_index_m1, ]

# nrow(train_m1)
# nrow(valid_m1)

# str(train_m1)
# str(valid_m1)

# Defining New Customers

new_custs <- read.csv("credit_test_fa2022_8.csv", header = TRUE)
# colnames(new_custs)

new_custs_m1 <- new_custs[ , -c(1, 2, 4, 10, 12, 14, 19:20, 21:24, 26:27, 30:67)]
# str(new_custs_m1)
# colnames(new_custs_m1)

new_custs_m1[, c(1:3, 8:10, 13:14)] <- lapply(new_custs_m1[, c(1:3, 8:10, 13:14)], as.fa
ctor)
# str(new_custs_m1)

# Normalising Numerical Variables
train_norm_m1 <- train_m1
valid_norm_m1 <- valid_m1

# str(train_norm_m1)
# colnames(train_norm_m1)
# str(valid_norm_m1)
# colnames(valid_norm_m1)

norm_values_m1 <- preProcess(train_m1[, c(6:8, 12:13)], method = c("center",
"scale"))
```

```
# Then normalise the training and validation sets.

train_norm_m1[, c(6:8, 12:13)] <- predict(norm_values_m1, train_m1[, c(6:8, 12:13)])

valid_norm_m1[, c(6:8, 12:13)] <- predict(norm_values_m1, valid_m1[, c(6:8, 12:13)])

# Predicting Normalised Values for the New Record
new_custs_norm_m1 <- predict(norm_values_m1, new_custs_m1)
new_custs_norm_m1
```

```
##   NAME_CONTRACT_TYPE FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN AMT_INCOME_TOTAL
## 1         Cash loans            N               Y            0       -0.7965008
## 2         Cash loans            N               Y            0        0.1179641
## 3         Cash loans            N               Y            0       -0.7965008
## 4         Cash loans            N               N            0       -0.3392684
## 5         Cash loans            N               Y            0        0.0265176
##   AMT_CREDIT AMT_GOODS_PRICE     NAME_INCOME_TYPE   NAME_FAMILY_STATUS
## 1 -0.7875042      -0.8167662 Commercial associate            Separated
## 2 -0.3938690      -0.4809786              Working            Separated
## 3 -1.0285656      -1.0406246            Pensioner Single / not married
## 4 -0.8016010      -0.7918931        State servant              Married
## 5 -0.2740808      -0.1949373              Working              Married
##   NAME_HOUSING_TYPE DAYS_BIRTH DAYS_EMPLOYED FLAG_CONT_MOBILE
## 1 House / apartment -0.8885148    -0.4527553                1
## 2  Rented apartment  0.3595704    -0.4552035                1
## 3 House / apartment -1.6675582     2.2300132                1
## 4 House / apartment -0.2558162    -0.4542563                1
## 5 House / apartment -1.1763569    -0.4511450                1
##         OCCUPATION_TYPE CNT_FAM_MEMBERS
## 1 Private service staff               1
## 2           Sales staff               1
## 3                                     1
## 4              Managers               2
## 5           Sales staff               2
```

```
compare_df_cols(valid_m1, valid_norm_m1, train_m1, train_norm_m1)
```

```
##           column_name valid_m1 valid_norm_m1 train_m1 train_norm_m1
## 1          AMT_CREDIT  numeric       numeric  numeric       numeric
## 2     AMT_GOODS_PRICE  numeric       numeric  numeric       numeric
## 3    AMT_INCOME_TOTAL  numeric       numeric  numeric       numeric
## 4        CNT_CHILDREN  integer       integer  integer       integer
## 5     CNT_FAM_MEMBERS  integer       integer  integer       integer
## 6          DAYS_BIRTH  integer       numeric  integer       numeric
## 7       DAYS_EMPLOYED  integer       numeric  integer       numeric
## 8    FLAG_CONT_MOBILE   factor        factor   factor        factor
## 9        FLAG_OWN_CAR   factor        factor   factor        factor
## 10    FLAG_OWN_REALTY   factor        factor   factor        factor
## 11 NAME_CONTRACT_TYPE   factor        factor   factor        factor
## 12 NAME_FAMILY_STATUS   factor        factor   factor        factor
## 13  NAME_HOUSING_TYPE   factor        factor   factor        factor
## 14   NAME_INCOME_TYPE   factor        factor   factor        factor
## 15    OCCUPATION_TYPE   factor        factor   factor        factor
## 16             TARGET   factor        factor   factor        factor
```

```
# Train k = 3

knn_model_k3_m1 <- caret::knn3(TARGET ~ ., data = train_norm_m1, k =
3)
knn_model_k3_m1
```

```
## 3-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 14470  3513
```

```
# Predict training set

knn_pred_k3_train_m1 <- predict(knn_model_k3_m1, newdata = train_norm_m1[,
-c(1)], type = "class")
# head(knn_pred_k3_train_m1)

# Evaluate

confusionMatrix(knn_pred_k3_train_m1, as.factor(train_norm_m1[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13876  2106
##          1   594  1407
##
##                Accuracy : 0.8499
##                  95% CI : (0.8446, 0.855)
##     No Information Rate : 0.8046
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4294
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.40051
##             Specificity : 0.95895
##          Pos Pred Value : 0.70315
##          Neg Pred Value : 0.86823
##              Prevalence : 0.19535
##          Detection Rate : 0.07824
##    Detection Prevalence : 0.11127
##       Balanced Accuracy : 0.67973
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k3_valid_m1 <- predict(knn_model_k3_m1, newdata = valid_norm_m1[,
-c(1)], type = "class")
# head(knn_pred_k3_valid_m1)

# Evaluate

confusionMatrix(knn_pred_k3_valid_m1, as.factor(valid_norm_m1[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8628 2005
##          1 1055  302
##
##                Accuracy : 0.7448
##                  95% CI : (0.7369, 0.7526)
##     No Information Rate : 0.8076
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.026
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.13091
##             Specificity : 0.89105
##          Pos Pred Value : 0.22255
##          Neg Pred Value : 0.81144
##              Prevalence : 0.19241
##          Detection Rate : 0.02519
##    Detection Prevalence : 0.11318
##       Balanced Accuracy : 0.51098
##
##        'Positive' Class : 1
##
```

```
# Train k=5

knn_model_k5_m1 <- caret::knn3(TARGET ~ ., data = train_norm_m1, k =
5)
knn_model_k5_m1
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 14470  3513
```

```
# Predict training set

knn_pred_k5_train_m1 <- predict(knn_model_k5_m1, newdata = train_norm_m1[,
-c(1)], type = "class")
# head(knn_pred_k5_train_m1)

# Evaluate

confusionMatrix(knn_pred_k5_train_m1, as.factor(train_norm_m1[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14006  2692
##          1   464   821
##
##                Accuracy : 0.8245
##                  95% CI : (0.8189, 0.83)
##     No Information Rate : 0.8046
##     P-Value [Acc > NIR] : 5.469e-12
##
##                   Kappa : 0.2654
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.23370
##             Specificity : 0.96793
##          Pos Pred Value : 0.63891
##          Neg Pred Value : 0.83878
##              Prevalence : 0.19535
##          Detection Rate : 0.04565
##    Detection Prevalence : 0.07146
##       Balanced Accuracy : 0.60082
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k5_valid_m1 <- predict(knn_model_k5_m1, newdata = valid_norm_m1[,
-c(1)], type = "class")
# head(knn_pred_k5_valid_m1)

# Evaluate
confusionMatrix(knn_pred_k5_valid_m1, as.factor(valid_norm_m1[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9040 2105
##          1  643  202
##
##                Accuracy : 0.7708
##                  95% CI : (0.7632, 0.7783)
##     No Information Rate : 0.8076
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0279
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.08756
##             Specificity : 0.93359
##          Pos Pred Value : 0.23905
##          Neg Pred Value : 0.81113
##              Prevalence : 0.19241
##          Detection Rate : 0.01685
##    Detection Prevalence : 0.07048
##       Balanced Accuracy : 0.51058
##
##        'Positive' Class : 1
##
```

```
# Train k=7

knn_model_k7_m1 <- caret::knn3(TARGET ~ ., data = train_norm_m1, k =
7)
knn_model_k7_m1
```

```
## 7-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 14470  3513
```

```
# Predict training set

knn_pred_k7_train_m1 <- predict(knn_model_k7_m1, newdata = train_norm_m1[,
-c(1)], type = "class")
# head(knn_pred_k7_train_m1)

# Evaluate

confusionMatrix(knn_pred_k7_train_m1, as.factor(train_norm_m1[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14108  2955
##          1   362   558
##
##                Accuracy : 0.8155
##                  95% CI : (0.8098, 0.8212)
##     No Information Rate : 0.8046
##     P-Value [Acc > NIR] : 0.0001071
##
##                   Kappa : 0.1857
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.15884
##             Specificity : 0.97498
##          Pos Pred Value : 0.60652
##          Neg Pred Value : 0.82682
##              Prevalence : 0.19535
##          Detection Rate : 0.03103
##    Detection Prevalence : 0.05116
##       Balanced Accuracy : 0.56691
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k7_valid_m1 <- predict(knn_model_k7_m1, newdata = valid_norm_m1[,
-c(1)], type = "class")
# head(knn_pred_k7_valid_m1)

# Evaluate
confusionMatrix(knn_pred_k7_valid_m1, as.factor(valid_norm_m1[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9241 2153
##          1  442  154
##
##                Accuracy : 0.7836
##                  95% CI : (0.7761, 0.7909)
##     No Information Rate : 0.8076
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0294
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.06675
##             Specificity : 0.95435
##          Pos Pred Value : 0.25839
##          Neg Pred Value : 0.81104
##              Prevalence : 0.19241
##          Detection Rate : 0.01284
##    Detection Prevalence : 0.04971
##       Balanced Accuracy : 0.51055
##
##        'Positive' Class : 1
##
```

Process and Analysis of Model 1

For this model, we decided to include 15 dependent variables; the type of loan being requested, car ownership, realty ownership, count of children, total income, total credit, price of goods to be purchased with loan, type of income, marriage status, housing type, age, and amount of time employed. We selected these variables based on what we felt would be most strongly correlated with a client's level of risk- for example, a client with a higher income is likely to be lower risk. We felt that clients who own a car or housing might also be lower risk as they have some assets. We were interested in the impacts of family circumstances on risk level, as perhaps clients with many dependents are higher risk due to their increased expenses, or clients who are married could be lower risk due to presumably being from two-income households.

We decided to deal with missing values in the data for this model by simply removing them- this is because null values made up a very small portion of the data (just 27 observations out of 30,000) which would not impact the results of our model significantly. We also decided to load in the new client data so we could organize, factorize, and normalise it along with the corresponding model data frame.

Overall, the accuracy of the k=3 version of this model was quite good- 85% in the training set and 74% for the validation set. The issue with the model, however, was the sensitivity, which is quite low in the validation set, only 13%. The sensitivity measures the true positive rate, meaning that while our model is technically accurate, it is not so good at predicting when clients had difficulty paying back their loans- of all of the high-risk clients, the algorithm was only able to successfully identify them 13% of the time. This is a problem, because this is the prediction we are interested in.

To combat this issue, we tried k values of 5 and 7 as well, but these did not fare much better in terms of accuracy, and actually performed even worse in terms of sensitivity. This told us that for our next model, we should try different dependent variables to predict the clients' risk-level, and also that we are working with an unbalanced data set. An unbalanced data set occurs when the target variable results are very skewed toward one value or category- in this case, the vast majority of clients being low-risk customers. As a result, the algorithm has a hard time learning when clients are high-risk, since there are so few examples of that case relative to the entire set of data.

Model 1.1

```
# Creating Model with Same Variables, but Weighted

credit_m1_w <- credit_m1

# Factorizing Categorical Variables

# colnames(credit_m1_w)
# str(credit_m1_w)
credit_m1_w[, c(1:4, 9:11, 14:15)] <- lapply(credit_m1_w[, c(1:4, 9:11, 14:15)], as.fact
or)

# Training/Validation Split

set.seed(666)

train_index_m1_w <- sample(1:nrow(credit_m1_w), 0.7 * nrow(credit_m1_w))
valid_index_m1_w <- setdiff(1:nrow(credit_m1_w), train_index_m1_w)

train_m1_w <- credit_m1_w[train_index_m1_w, ]
valid_m1_w <- credit_m1_w[valid_index_m1_w, ]

# nrow(train_m1_w)
# nrow(valid_m1_w)

# str(train_m1_w)
# str(valid_m1_w)



#weighted sampling
train_rose_m1 <- ROSE(TARGET ~.,
                      data = train_m1_w, seed = 666)$data

# Defining new customers

new_custs_m1_w <- new_custs_m1
# str(new_custs_m1_w)
# colnames(new_custs_m1_w)

new_custs_m1_w[, c(1:3, 8:10, 13:14)] <- lapply(new_custs_m1_w[, c(1:3, 8:10, 13:14)], a
s.factor)
# str(new_custs_m1_w)


# Normalising Numerical Variables
train_norm_m1_w <- train_rose_m1
valid_norm_m1_w <- valid_m1_w

# str(train_norm_m1_w)
# colnames(train_norm_m1_w)
# str(valid_norm_m1_w)
# colnames(valid_norm_m1_w)
```

```
norm_values_m1_w <- preProcess(train_m1_w[, c(6:8, 12:13)], method = c("center",
"scale"))


# Then normalise the training and validation sets.

train_norm_m1_w[, c(6:8, 12:13)] <- predict(norm_values_m1_w, train_m1_w[, c(6:8, 12:1
3)])

valid_norm_m1_w[, c(6:8, 12:13)] <- predict(norm_values_m1_w, valid_m1_w[, c(6:8, 12:1
3)])

# Predicting Normalised Values for the New Record
new_custs_norm_m1_w <- predict(norm_values_m1_w, new_custs_m1_w)
new_custs_norm_m1_w
```

```
##   NAME_CONTRACT_TYPE FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN AMT_INCOME_TOTAL
## 1         Cash loans            N               Y            0      -0.79829965
## 2         Cash loans            N               Y            0       0.12255037
## 3         Cash loans            N               Y            0      -0.79829965
## 4         Cash loans            N               N            0      -0.33787464
## 5         Cash loans            N               Y            0       0.03046537
##   AMT_CREDIT AMT_GOODS_PRICE     NAME_INCOME_TYPE   NAME_FAMILY_STATUS
## 1 -0.7854209      -0.8146010 Commercial associate            Separated
## 2 -0.3925867      -0.4796103              Working            Separated
## 3 -1.0259917      -1.0379281            Pensioner Single / not married
## 4 -0.7994890      -0.7897869        State servant              Married
## 5 -0.2730423      -0.1942478              Working              Married
##   NAME_HOUSING_TYPE DAYS_BIRTH DAYS_EMPLOYED FLAG_CONT_MOBILE
## 1 House / apartment -0.8835681    -0.4545543                1
## 2  Rented apartment  0.3618861    -0.4569957                1
## 3 House / apartment -1.6609691     2.2208411                1
## 4 House / apartment -0.2522032    -0.4560511                1
## 5 House / apartment -1.1708034    -0.4529484                1
##         OCCUPATION_TYPE CNT_FAM_MEMBERS
## 1 Private service staff               1
## 2           Sales staff               1
## 3                                     1
## 4              Managers               2
## 5           Sales staff               2
```

```
compare_df_cols(valid_m1_w, valid_norm_m1_w, train_m1_w, train_norm_m1_w)
```

```
##           column_name valid_m1_w valid_norm_m1_w train_m1_w train_norm_m1_w
## 1          AMT_CREDIT    numeric         numeric    numeric         numeric
## 2     AMT_GOODS_PRICE    numeric         numeric    numeric         numeric
## 3    AMT_INCOME_TOTAL    numeric         numeric    numeric         numeric
## 4        CNT_CHILDREN    integer         integer    integer         numeric
## 5     CNT_FAM_MEMBERS    integer         integer    integer         numeric
## 6          DAYS_BIRTH    integer         numeric    integer         numeric
## 7       DAYS_EMPLOYED    integer         numeric    integer         numeric
## 8    FLAG_CONT_MOBILE     factor          factor     factor          factor
## 9        FLAG_OWN_CAR     factor          factor     factor          factor
## 10    FLAG_OWN_REALTY     factor          factor     factor          factor
## 11 NAME_CONTRACT_TYPE     factor          factor     factor          factor
## 12 NAME_FAMILY_STATUS     factor          factor     factor          factor
## 13  NAME_HOUSING_TYPE     factor          factor     factor          factor
## 14   NAME_INCOME_TYPE     factor          factor     factor          factor
## 15    OCCUPATION_TYPE     factor          factor     factor          factor
## 16             TARGET     factor          factor     factor          factor
```

```
# Train k = 3

knn_model_k3_m1_w <- caret::knn3(TARGET ~ ., data = train_norm_m1_w, k =
3)
knn_model_k3_m1_w
```

```
## 3-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 10339 10642
```

```
# Predict training set

knn_pred_k3_train_m1_w <- predict(knn_model_k3_m1_w, newdata = train_norm_m1_w[,
-c(1)], type = "class")
# head(knn_pred_k3_train_m1_w)

# Evaluate

confusionMatrix(knn_pred_k3_train_m1_w, as.factor(train_norm_m1_w[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7992 2250
##          1 2347 8392
##
##                Accuracy : 0.7809
##                  95% CI : (0.7752, 0.7865)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5616
##
##  Mcnemar's Test P-Value : 0.1568
##
##             Sensitivity : 0.7886
##             Specificity : 0.7730
##          Pos Pred Value : 0.7815
##          Neg Pred Value : 0.7803
##              Prevalence : 0.5072
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.5118
##       Balanced Accuracy : 0.7808
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k3_valid_m1_w <- predict(knn_model_k3_m1_w, newdata = valid_norm_m1_w[,
-c(1)], type = "class")
# head(knn_pred_k3_valid_m1_w)

# Evaluate

confusionMatrix(knn_pred_k3_valid_m1_w, as.factor(valid_norm_m1_w[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3896  877
##          1 3365  854
##
##                Accuracy : 0.5282
##                  95% CI : (0.5179, 0.5386)
##     No Information Rate : 0.8075
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0193
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.49336
##             Specificity : 0.53657
##          Pos Pred Value : 0.20242
##          Neg Pred Value : 0.81626
##              Prevalence : 0.19250
##          Detection Rate : 0.09497
##    Detection Prevalence : 0.46919
##       Balanced Accuracy : 0.51496
##
##        'Positive' Class : 1
##
```

```
# Train k=5

knn_model_k5_m1_w <- caret::knn3(TARGET ~ ., data = train_norm_m1_w, k =
5)
knn_model_k5_m1_w
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 10339 10642
```

```
# Predict training set

knn_pred_k5_train_m1_w <- predict(knn_model_k5_m1_w, newdata = train_norm_m1_w[,
-c(1)], type = "class")
# head(knn_pred_k5_train_m1_w)

# Evaluate

confusionMatrix(knn_pred_k5_train_m1_w, as.factor(train_norm_m1_w[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7390 2838
##          1 2949 7804
##
##                Accuracy : 0.7242
##                  95% CI : (0.7181, 0.7302)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4482
##
##  Mcnemar's Test P-Value : 0.1482
##
##             Sensitivity : 0.7333
##             Specificity : 0.7148
##          Pos Pred Value : 0.7258
##          Neg Pred Value : 0.7225
##              Prevalence : 0.5072
##          Detection Rate : 0.3720
##    Detection Prevalence : 0.5125
##       Balanced Accuracy : 0.7240
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k5_valid_m1_w <- predict(knn_model_k5_m1_w, newdata = valid_norm_m1_w[,
-c(1)], type = "class")
# head(knn_pred_k5_valid_m1_w)

# Evaluate
confusionMatrix(knn_pred_k5_valid_m1_w, as.factor(valid_norm_m1_w[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3931  884
##          1 3330  847
##
##                Accuracy : 0.5314
##                  95% CI : (0.521, 0.5417)
##     No Information Rate : 0.8075
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.02
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.48931
##             Specificity : 0.54139
##          Pos Pred Value : 0.20278
##          Neg Pred Value : 0.81641
##              Prevalence : 0.19250
##          Detection Rate : 0.09419
##    Detection Prevalence : 0.46452
##       Balanced Accuracy : 0.51535
##
##        'Positive' Class : 1
##
```

```
# Train k=7

knn_model_k7_m1_w <- caret::knn3(TARGET ~ ., data = train_norm_m1_w, k =
7)
knn_model_k7_m1_w
```

```
## 7-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 10339 10642
```

```
# Predict training set

knn_pred_k7_train_m1_w <- predict(knn_model_k7_m1_w, newdata = train_norm_m1_w[,
-c(1)], type = "class")
# head(knn_pred_k7_train_m1_w)

# Evaluate

confusionMatrix(knn_pred_k7_train_m1_w, as.factor(train_norm_m1_w[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7142 3167
##          1 3197 7475
##
##                Accuracy : 0.6967
##                  95% CI : (0.6904, 0.7029)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.3932
##
##  Mcnemar's Test P-Value : 0.7162
##
##             Sensitivity : 0.7024
##             Specificity : 0.6908
##          Pos Pred Value : 0.7004
##          Neg Pred Value : 0.6928
##              Prevalence : 0.5072
##          Detection Rate : 0.3563
##    Detection Prevalence : 0.5087
##       Balanced Accuracy : 0.6966
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k7_valid_m1_w <- predict(knn_model_k7_m1_w, newdata = valid_norm_m1_w[,
-c(1)], type = "class")
# head(knn_pred_k7_valid_m1_w)

# Evaluate
confusionMatrix(knn_pred_k7_valid_m1_w, as.factor(valid_norm_m1_w[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3971  847
##          1 3290  884
##
##                Accuracy : 0.5399
##                  95% CI : (0.5296, 0.5503)
##     No Information Rate : 0.8075
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0375
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.51069
##             Specificity : 0.54689
##          Pos Pred Value : 0.21179
##          Neg Pred Value : 0.82420
##              Prevalence : 0.19250
##          Detection Rate : 0.09831
##    Detection Prevalence : 0.46419
##       Balanced Accuracy : 0.52879
##
##        'Positive' Class : 1
##
```

Process and Analysis of Model 1.1

The process for creating this model was exactly the same as our initial model, but with weighted training data, and a slightly larger subset of the data being devoted to the training set in an attempt to reduce over-fitting in our model. Over-fitting occurs when the algorithm is tailored so specifically to the data it was trained with, that it is less accurate when it is given new information as a result- we decided to increase the scope of the training data in hopes of making the algorithm better at predicting new records.

This version of the model improved the sensitivity, which is good, but it decreased the overall accuracy of the model. The model with the highest overall accuracy and sensitivity was the k=7 version of the model, which had an accuracy of 54% and a sensitivity of 48%- this means that while the algorithm was technically worse at accurately predicting risk among the clients overall, it was much better at accurately identifying the high-risk clients (48% of the time as opposed to just 13%).

Hoping to improve the accuracy even further, we decided to try out different variables in our algorithm.

Model 2

```
# New Variables, weighted

credit_m2 <- credit[ , c(3, 6:10, 18)]

credit_m2 <- credit_m2[complete.cases(credit_m2),]

# str(credit_m2)

# Factorizing
credit_m2[, c(1:3)] <- lapply(credit_m2[, c(1:3)], as.factor)

# Training Validation Split

set.seed(666)

train_index_m2 <- sample(1:nrow(credit_m2), 0.7 * nrow(credit_m2))
valid_index_m2 <- setdiff(1:nrow(credit_m2), train_index_m2)

train_m2 <- credit_m2[train_index_m2, ]
valid_m2 <- credit_m2[valid_index_m2, ]

# nrow(train_m2)
# nrow(valid_m2)

# str(train_m2)
# str(valid_m2)

# Defining New Customers

new_custs_m2 <- new_custs[ , c(5:9, 17)]
# str(new_custs_m2)
# colnames(new_custs_m2)

new_custs_m2[, c(1:2)] <- lapply(new_custs_m2[, c(1:2)], as.factor)
#str(new_custs)

#weighted sampling
train_rose_m2 <- ROSE(TARGET ~.,
                      data = train_m2, seed = 666)$data

# Normalisation, only for numerical variables

train_norm_m2 <- train_rose_m2
valid_norm_m2 <- valid_m2

# str(train_norm_m2)
# colnames(train_norm_m2)
# str(valid_norm_m2)
# colnames(valid_norm_m2)

norm_values_m2 <- preProcess(train_m2[, c(4:7)], method = c("center",
"scale"))
```

```
# Then normalise the training and validation sets.

train_norm_m2[, c(4:7)] <- predict(norm_values_m2, train_m2[, c(4:7)])

valid_norm_m2[, c(4:7)] <- predict(norm_values_m2, valid_m2[, c(4:7)])

#predicting normalized values for the new record
new_custs_norm_m2 <- predict(norm_values_m2, new_custs_m2)
new_custs_norm_m2
```

```
##   FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN AMT_INCOME_TOTAL AMT_CREDIT
## 1            N               Y   -0.5799568      -0.80442809 -0.7797901
## 2            N               Y   -0.5799568       0.12630397 -0.3892350
## 3            N               Y   -0.5799568      -0.80442809 -1.0189652
## 4            N               N   -0.5799568      -0.33906206 -0.7937766
## 5            N               Y   -0.5799568       0.03323076 -0.2703842
##   DAYS_BIRTH
## 1 -0.8816690
## 2  0.3628701
## 3 -1.6584990
## 4 -0.2507680
## 5 -1.1686933
```

```
compare_df_cols(valid_m2, valid_norm_m2, train_m2, train_norm_m2)
```

```
##        column_name valid_m2 valid_norm_m2 train_m2 train_norm_m2
## 1       AMT_CREDIT  numeric       numeric  numeric       numeric
## 2 AMT_INCOME_TOTAL  numeric       numeric  numeric       numeric
## 3     CNT_CHILDREN  integer       numeric  integer       numeric
## 4       DAYS_BIRTH  integer       numeric  integer       numeric
## 5     FLAG_OWN_CAR   factor        factor   factor        factor
## 6  FLAG_OWN_REALTY   factor        factor   factor        factor
## 7           TARGET   factor        factor   factor        factor
```

```
# Train k=3

knn_model_k3_m2 <- caret::knn3(TARGET ~ ., data = train_norm_m2, k =
3)
knn_model_k3_m2
```

```
## 3-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 10347 10653
```

```
# Predict training set

knn_pred_k3_train_m2 <- predict(knn_model_k3_m2, newdata = train_norm_m2[,
-c(1)], type = "class")
# head(knn_pred_k3_train_m2)

# Evaluate

confusionMatrix(knn_pred_k3_train_m2, as.factor(train_norm_m2[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7733 2573
##          1 2614 8080
##
##                Accuracy : 0.753
##                  95% CI : (0.7471, 0.7588)
##     No Information Rate : 0.5073
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5059
##
##  Mcnemar's Test P-Value : 0.5786
##
##             Sensitivity : 0.7585
##             Specificity : 0.7474
##          Pos Pred Value : 0.7556
##          Neg Pred Value : 0.7503
##              Prevalence : 0.5073
##          Detection Rate : 0.3848
##    Detection Prevalence : 0.5092
##       Balanced Accuracy : 0.7529
##
##        'Positive' Class : 1
##
```

```
# Predict validation set
knn_pred_k3_valid_m2 <- predict(knn_model_k3_m2, newdata = valid_norm_m2[,
-c(1)], type = "class")
# head(knn_pred_k3_valid_m2)

# Evaluate

confusionMatrix(knn_pred_k3_valid_m2, as.factor(valid_norm_m2[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3567  857
##          1 3634  942
##
##                Accuracy : 0.501
##                  95% CI : (0.4906, 0.5114)
##     No Information Rate : 0.8001
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.012
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5236
##             Specificity : 0.4953
##          Pos Pred Value : 0.2059
##          Neg Pred Value : 0.8063
##              Prevalence : 0.1999
##          Detection Rate : 0.1047
##    Detection Prevalence : 0.5084
##       Balanced Accuracy : 0.5095
##
##        'Positive' Class : 1
##
```

```
# Train k = 5

knn_model_k5_m2 <- caret::knn3(TARGET ~ ., data = train_norm_m2, k =
5)
knn_model_k5_m2
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 10347 10653
```

```
# Predict training set

knn_pred_k5_train_m2 <- predict(knn_model_k5_m2, newdata = train_norm_m2[,
-c(1)], type = "class")
# head(knn_pred_k5_train_m2)

# Evaluate

confusionMatrix(knn_pred_k5_train_m2, as.factor(train_norm_m2[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7039 3263
##          1 3308 7390
##
##                Accuracy : 0.6871
##                  95% CI : (0.6808, 0.6934)
##     No Information Rate : 0.5073
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.374
##
##  Mcnemar's Test P-Value : 0.5873
##
##             Sensitivity : 0.6937
##             Specificity : 0.6803
##          Pos Pred Value : 0.6908
##          Neg Pred Value : 0.6833
##              Prevalence : 0.5073
##          Detection Rate : 0.3519
##    Detection Prevalence : 0.5094
##       Balanced Accuracy : 0.6870
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k5_valid_m2 <- predict(knn_model_k5_m2, newdata = valid_norm_m2[,
-c(1)], type = "class")
# head(knn_pred_k5_valid_m2)

# Evaluate
confusionMatrix(knn_pred_k5_valid_m2, as.factor(valid_norm_m2[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3522  860
##          1 3679  939
##
##                Accuracy : 0.4957
##                  95% CI : (0.4853, 0.5061)
##     No Information Rate : 0.8001
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.007
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5220
##             Specificity : 0.4891
##          Pos Pred Value : 0.2033
##          Neg Pred Value : 0.8037
##              Prevalence : 0.1999
##          Detection Rate : 0.1043
##    Detection Prevalence : 0.5131
##       Balanced Accuracy : 0.5055
##
##        'Positive' Class : 1
##
```

```
# Train k=7

knn_model_k7_m2 <- caret::knn3(TARGET ~ ., data = train_norm_m2, k =
7)
knn_model_k7_m2
```

```
## 7-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 10347 10653
```

```
# Predict training set

knn_pred_k7_train_m2 <- predict(knn_model_k7_m2, newdata = train_norm_m2[,
-c(1)], type = "class")
# head(knn_pred_k7_train_m2)

# Evaluate

confusionMatrix(knn_pred_k7_train_m2, as.factor(train_norm_m2[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6685 3582
##          1 3662 7071
##
##                Accuracy : 0.655
##                  95% CI : (0.6486, 0.6615)
##     No Information Rate : 0.5073
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.3099
##
##  Mcnemar's Test P-Value : 0.3533
##
##             Sensitivity : 0.6638
##             Specificity : 0.6461
##          Pos Pred Value : 0.6588
##          Neg Pred Value : 0.6511
##              Prevalence : 0.5073
##          Detection Rate : 0.3367
##    Detection Prevalence : 0.5111
##       Balanced Accuracy : 0.6549
##
##        'Positive' Class : 1
##
```

```
# Predict validation set

knn_pred_k7_valid_m2 <- predict(knn_model_k7_m2, newdata = valid_norm_m2[,
-c(1)], type = "class")
# head(knn_pred_k7_valid_m2)

# Evaluate
confusionMatrix(knn_pred_k7_valid_m2, as.factor(valid_norm_m2[, 1]), positive
= "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3447  859
##          1 3754  940
##
##                Accuracy : 0.4874
##                  95% CI : (0.4771, 0.4978)
##     No Information Rate : 0.8001
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 7e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5225
##             Specificity : 0.4787
##          Pos Pred Value : 0.2003
##          Neg Pred Value : 0.8005
##              Prevalence : 0.1999
##          Detection Rate : 0.1044
##    Detection Prevalence : 0.5216
##       Balanced Accuracy : 0.5006
##
##        'Positive' Class : 1
##
```

Process and Analysis of Model 2

For this model, we decided to select different variables, but continue with the same process of weighted training data. The variables we considered in Model 2 were car ownership, realty ownership, children, income, credit, and age. Our reasoning for selecting these was similar in our first model, in that based on domain knowledge, we considered these to be likely predictors of a client's ability to afford their loan. For example, an older client might be lower-risk as they have accumulated more assets throughout their life, and a client who has been employed for longer may have more resources saved up that would help them to repay their loan.

Overall, none of the k values tried for this model significantly impacted our results from version 1.1 of the model.

Model Selection

Based on the outcomes of all of the models, the model we would choose is the weighted version of our first model, called 'Model 1.1' with k = 3 in this document. Model 1.1 is more applicable than Model 1 because it predicts late loan repayments and on time payments. Model 1 does not use weighted data so it is barely trained to predict late repayments. Model 2 has the lower accuracy even though it uses weighted data. # Predicting New Outcome

```
new_cust_predict <- predict(knn_model_k7_m1_w, newdata = new_custs_norm_m1_w, type
= "class")

new_cust_predict
```

```
## [1] 0 1 0 0 1
## Levels: 0 1
```

Analysis

These results tell us that of these 5 new clients, 2 are predicted to be at a high financial risk for not repaying their loans to Stark Industries.

Summary

Improving the sensitivity of our model was difficult- this is because the vast majority of clients in the available data were not high-risk, making it difficult to train the algorithm to recognize when someone would be high-risk. Acknowledging this limitation is important, as in this case, ‘accuracy’ alone is not as significant as being able to successfully identify when a client is at a higher risk of being unable to pay back their loan, which is what Stark Industries is ultimately interested in learning.

Sometimes, there is a trade-off between the overall accuracy of the model, and it’s true-positive rate- deciding which is more important for the model depends on the context of the scenario, and the specific objective of the analyst(s). Acknowledging the limitations of a model is very important, as it determines how the company will use the model. When Stark Industries uses our algorithm to make predictions about the risk-level of their potential clients, they should be aware that of the clients identified as high-risk, there is a 48% liklihood that the model will identify them- Stark Industries could use this as a supplemental tool, in that case, to flag these individuals in their system and take special care to check in with these clients, for example, but should not simply exclude them from loan eligibility altogether based on their predicted risk-level alone. Stark Industries could use this tool as a helpful initial screening process- this would improve their efficiency in sorting through potential clients, and could prevent the harmful effects of personal bias on the part of bank employees in their decision to accept or deny loans to some degree.