PREDICTING CREDIT CARD RISK

Jakub Below

March 2022

INTRODUCTION

This project originates from the kaggle's dataset "Credit Card Approval Prediction" that can be found under the following address: https://www.kaggle.com/rikdifos/credit-card-approval-prediction. The data has been saved in the 'credit_data.rda' file containing credit card history with corresponding records listing personal information along with the status of the credit (e.g. whether the customer defaults and for how long).

The goal of this project is to predict whether the applicant is going to be at risk of default based on provided features. Due to economic hardships, the bank wants to introduce an additional vetting step to limit credit lines to customers who may be at risk.

SUMMARY

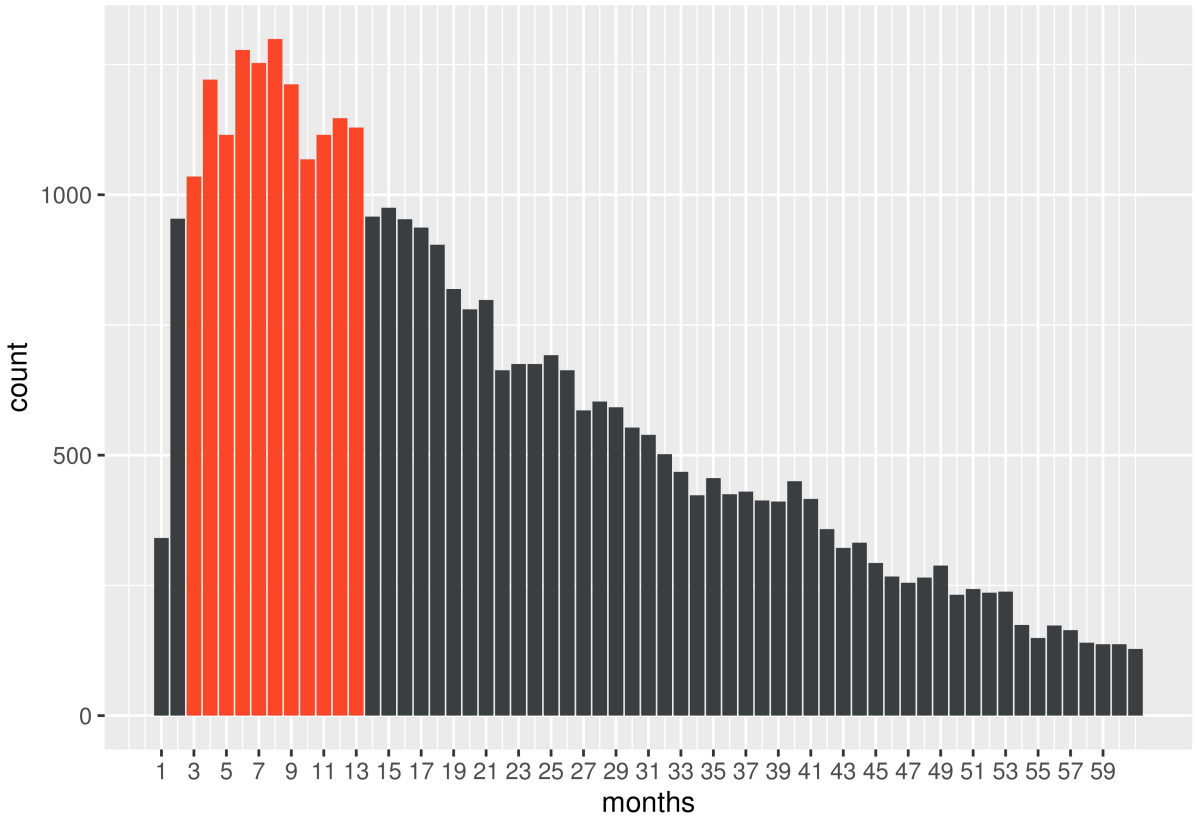
In the following sections of this report, the author has defined business goals and specific measures for the evaluation of prediction models. The author reviewed all the available features and prepared the data set for training. Based on selected criteria of maximizing Balanced Accuracy (the leading indicator) along with F1 and Specificity ratings, the author built and evaluated several models: linear algorithm, Decision Tree, Random Forest with custom weights, and Random Forest with over-sampling technique. The best model has been further evaluated against a separate validation set. Finally, the author has shared his conclusions around project constraints and future works.

OVERVIEW

The given data set has 777,715 rows and 20 columns.

ID	Min. :5008804	Median :5069530	Mean :5078743	Max. :5150487
MONTHS_BALANCE	Min. :-60.00	Median :-17.00	Mean :-19.37	Max. : 0.00
STATUS	Length:777715	Mode :character	NA	NA
CODE_GENDER	Length:777715	Mode :character	NA	NA
FLAG_OWN_CAR	Length:777715	Mode :character	NA	NA
FLAG_OWN_REALTY	Length:777715	Mode :character	NA	NA
CNT_CHILDREN	Min. : 0.0000	Median : 0.0000	Mean : 0.4281	Max. :19.0000
AMT_INCOME_TOTAL	Min. : 27000	Median : 162000	Mean : 188535	Max. :1575000
NAME_INCOME_TYPE	Length:777715	Mode :character	NA	NA
NAME_EDUCATION_TYPE	Length:777715	Mode :character	NA	NA
NAME_FAMILY_STATUS	Length:777715	Mode :character	NA	NA
NAME_HOUSING_TYPE	Length:777715	Mode :character	NA	NA
DAYS_BIRTH	Min. :-25152	Median :-15760	Mean :-16125	Max. : -7489
DAYS_EMPLOYED	Min. :-15713	Median : -1682	Mean : 57776	Max. :365243
FLAG_MOBIL	Min. :1	Median :1	Mean :1	Max. :1
FLAG_WORK_PHONE	Min. :0.0000	Median :0.0000	Mean :0.2318	Max. :1.0000
FLAG_PHONE	Min. :0.000	Median :0.000	Mean :0.301	Max. :1.000
FLAG_EMAIL	Min. :0.00000	Median :0.00000	Mean :0.09168	Max. :1.00000
OCCUPATION_TYPE	Length:777715	Mode :character	NA	NA
CNT_FAM_MEMBERS	Min. : 1.000	Median : 2.000	Mean : 2.209	Max. :20.000

Considering that ID is not informative and STATUS and MONTHS_BALANCE are used for rating customers the model will try to predict, we are left with 17 features for the prediction model. The data set contains multiple records for different months recorded per customer. There are 36,457 unique credit holders with most of them having a credit line open between 3 and 13 months.



There are 0 NA values in the data set, although as we will see some customers have no occupation on record.

BUSINESS OBJECTIVE

The data set currently lists multiple entries per customer with identical features across all of them, with the exception of the month the entry has been recorded and credit status. The statuses are as follows:

- 0: 1-29 days past due
- 1: 30-59 days past due
- 2: 60-89 days overdue
- 3: 90-119 days overdue
- 4: 120-149 days overdue
- 5: Overdue or bad debts, write-offs for more than 150 days
- C: paid off that month
- X: No loan for the month

ID	MONTHS_BALANCE	STATUS
5009002	-38	X
5009002	-39	0
5009002	-40	0
5009002	-41	0
5009002	-42	0
5009002	-43	0
5009002	-44	0

For example, the above customer has 7 corresponding entries and had an open credit line for six months but finally paid off the credit card balance (status X) and was never due for longer than 30 days (status 0). This customer was not at risk of default.

As presented earlier, the data set has 771,715 entries for 36,457 unique customers. We do not need to keep all of these records since features are identical across each customer's history and keeping different numbers of duplicated entries would skew the data towards individuals who had an open credit line for a longer period of time (i.e. for the above customer, we would count each feature 7 times). We will therefore group the data set by applicant ID. First, however, we need to define how to rate credit card history in order to divide customers between risky and safe groups.

For the purpose of this project, a risky customer is one that has defaulted for over 30 days at any time (indicating some financial problems).

```
cnames <- colnames(data)[-c(2,3)] # column names without MONTHS and STATUS

scoring_data <- data %>%
  group_by(.dots=cnames) %>%
  summarize(
    RISKY = case_when(            # over 30 days due - mark as risky
      any(STATUS == 2 |
          STATUS == 3 |
          STATUS == 4 |
          STATUS == 5) ~ 1,
      TRUE ~ 0                    # otherwise mark as not risky
    )
  ) %>% as.data.frame()

round(mean(scoring_data$RISKY==1) * 100,2)    # 1.69% risky customers
```

```
## [1] 1.69
```

The prevalence for customers at risk of default is only 1.69%, which makes sense business-wise (because the bank already rejected the most problematic customers applications) but poses serious issues for our predictive models. We will try to mitigate this constraint when evaluating respective algorithms.

DATA SET ANALYSIS

It's time to investigate features, starting with categorical ones.

```
## 'data.frame':    36457 obs. of  19 variables:
##  $ ID              : int  5008804 5008805 ...
##  $ CODE_GENDER     : chr  "M" ...
##  $ FLAG_OWN_CAR    : chr  "Y" ...
##  $ FLAG_OWN_REALTY : chr  "Y" ...
##  $ CNT_CHILDREN    : int  0 0 ...
##  $ AMT_INCOME_TOTAL: num  427500 ...
##  $ INCOME_SRC      : chr  "Working" ...
##  $ EDUCATION       : chr  "Higher education" ...
##  $ FAMILY          : chr  "Civil marriage" ...
##  $ HOUSING         : chr  "Rented apartment" ...
##  $ DAYS_BIRTH      : int  -12005 -12005 ...
```

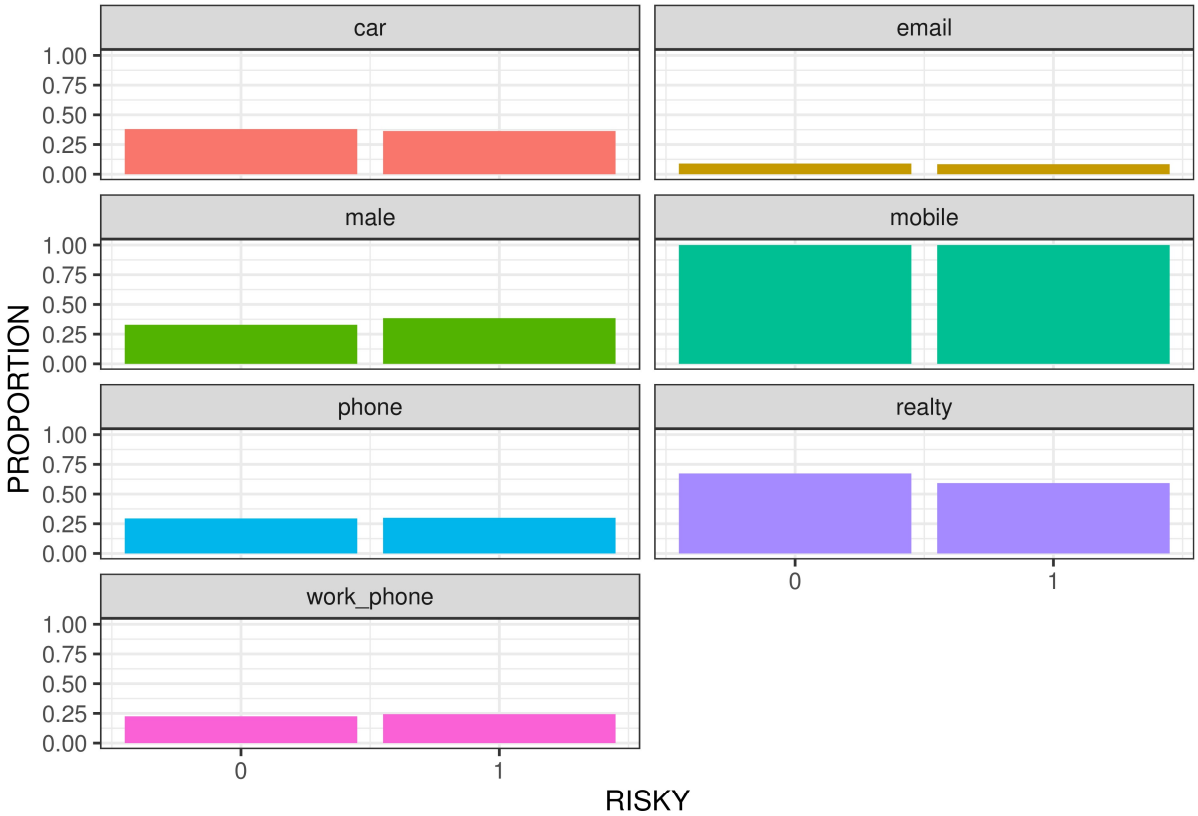
```
##  $ DAYS_EMPLOYED   : int  -4542 -4542 ...
##  $ FLAG_MOBIL      : int  1 1 ...
##  $ FLAG_WORK_PHONE : int  1 1 ...
##  $ FLAG_PHONE      : int  0 0 ...
##  $ FLAG_EMAIL      : int  0 0 ...
##  $ OCCUPATION      : chr  "" ...
##  $ CNT_FAM_MEMBERS : num  2 2 ...
##  $ RISKY           : num  0 0 ...
```

To summarize, we have observed the following features groups to work with:

- 7 binary variables (gender, car ownership, realty ownership, mobile phone, work phone, phone (other), and email)
- 5 categorical variables (income source, education level, marital status, housing conditions, and occupation)
- 5 discrete and continuous numeric variables (family size, number of children, income, days from birth, and days employed)

Binary features

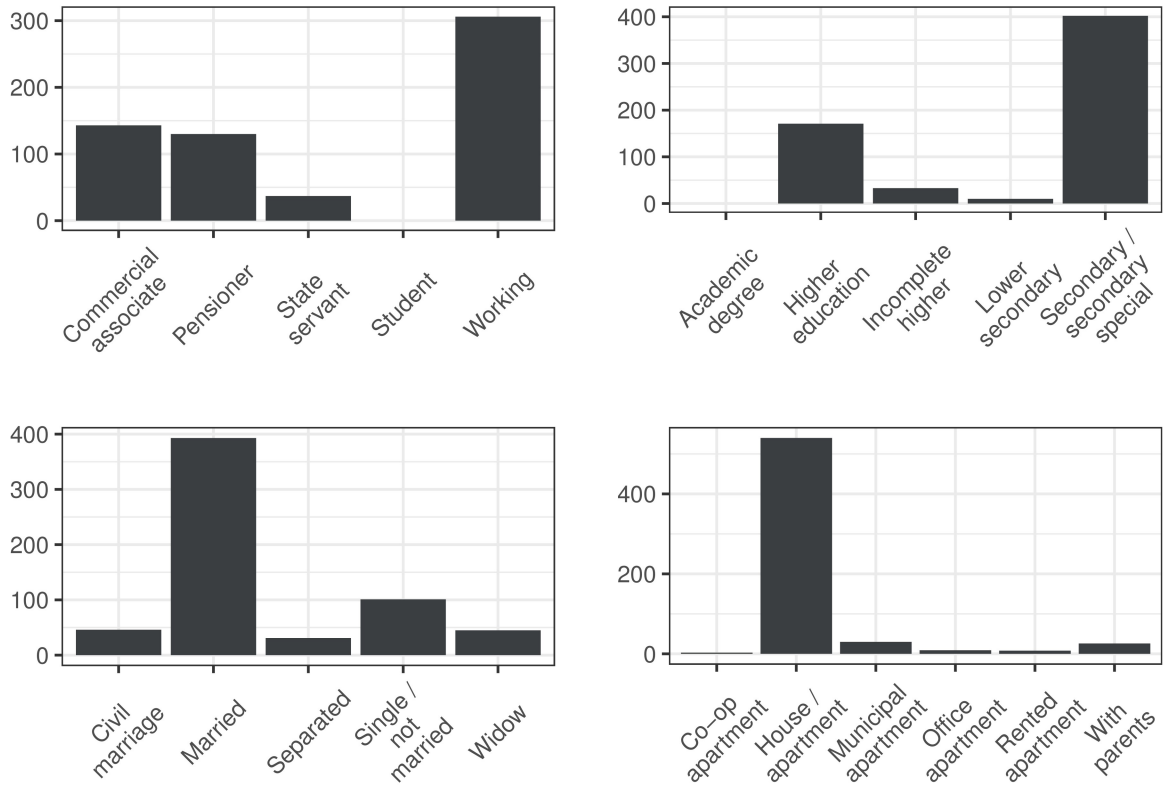
Let's start with a brief overview of binary features.



Considering all the binary features, there are no significant differences between risky and safe customers when it comes to these characteristics. Males and females, car owners and people without a car, renters and realty owners - they all seem to be just as likely to default on their credit card payments with minor exceptions.

Categorical features

Categorical features offer more variability.



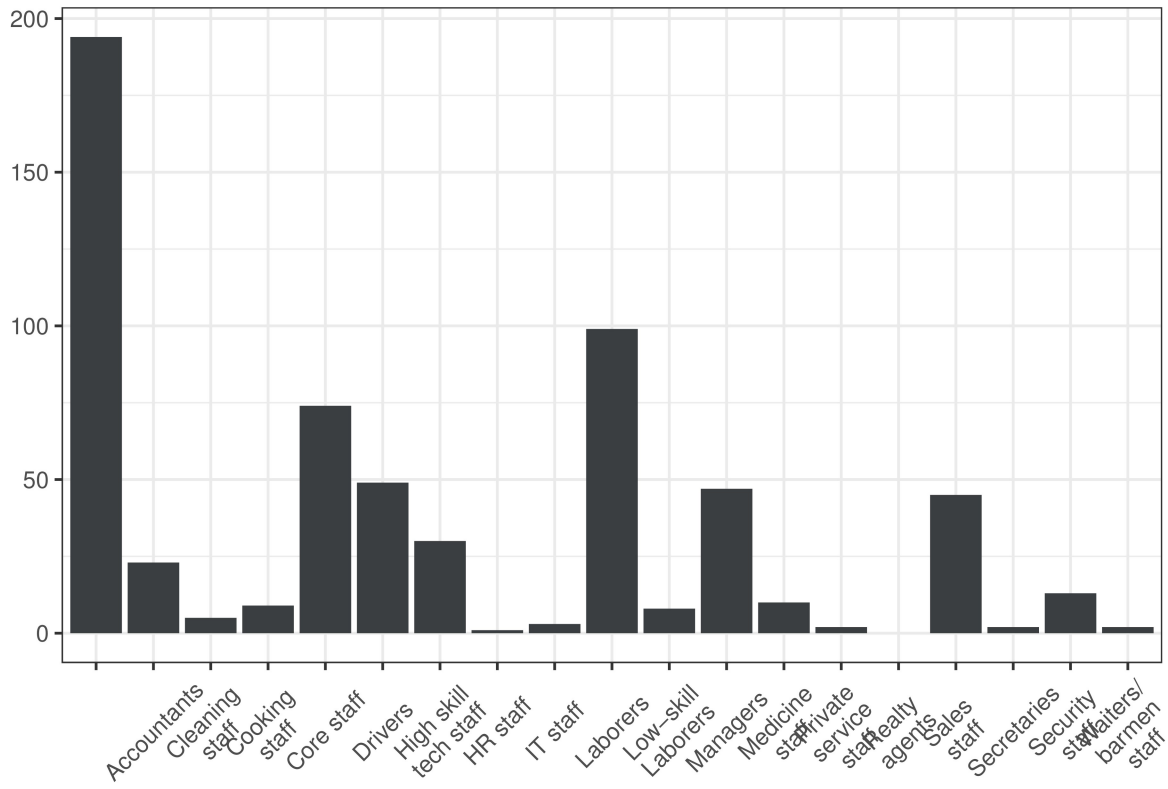
When looking just at the individuals rated as risky clients, there are several groups that stand out. Primarily, working individuals are defaulting more often than students due to more stringent requirements for applicants. Students do not get credit cards.

In fact, these proportions correspond to real proportions observed in the overall population as observed when reviewing the percentage point difference between prevalence in both populations.

name	value	difference
FAMILY	Married	-0.0490690
INCOME_SRC	Pensioner	0.0422922
FAMILY	Single / not married	0.0315036
FAMILY	Widow	0.0310298
EDUCATION	Secondary / secondary special	-0.0270252
INCOME_SRC	State servant	-0.0218123
INCOME_SRC	Working	-0.0194439
HOUSING	Municipal apartment	0.0177607
HOUSING	House / apartment	-0.0161544
EDUCATION	Incomplete higher	0.0148957
HOUSING	Office apartment	0.0074238
FAMILY	Separated	-0.0073597
EDUCATION	Higher education	0.0070321
HOUSING	With parents	-0.0065071
FAMILY	Civil marriage	-0.0061048
EDUCATION	Lower secondary	0.0059751
HOUSING	Rented apartment	-0.0027850
EDUCATION	Academic degree	-0.0008777
INCOME_SRC	Commercial associate	-0.0007342
INCOME_SRC	Student	-0.0003017
HOUSING	Co-op apartment	0.0002620

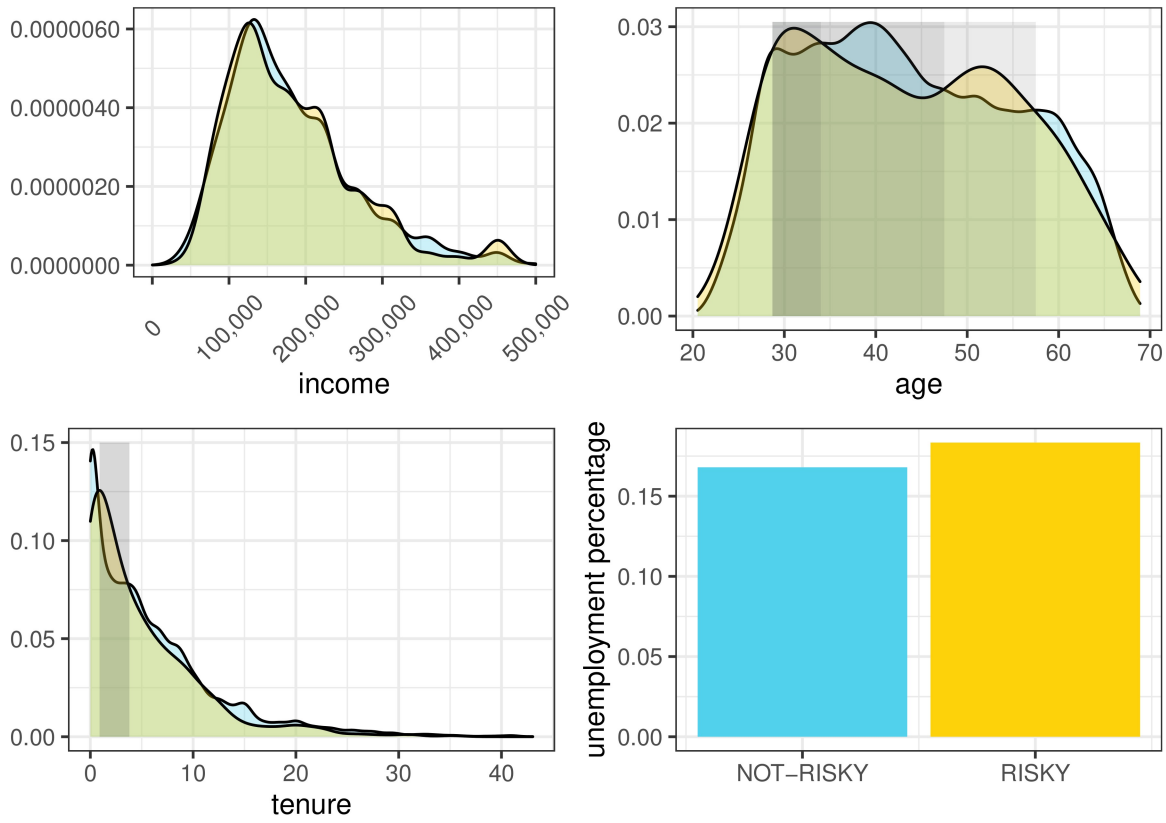
This, unfortunately, makes it very difficult to make any predictions based on given characteristics. The biggest difference between the group of risky customers and the overall population pertains to pensioners and married people. The percentage of these categories of people in the risky group is over 4 percentage points lower than in the overall population. This, however, does not allow for any significant inferences about those groups.

Some differences can also be observed when comparing occupation, with the most prevalent group being people without their occupation on record.



Discrete numeric features

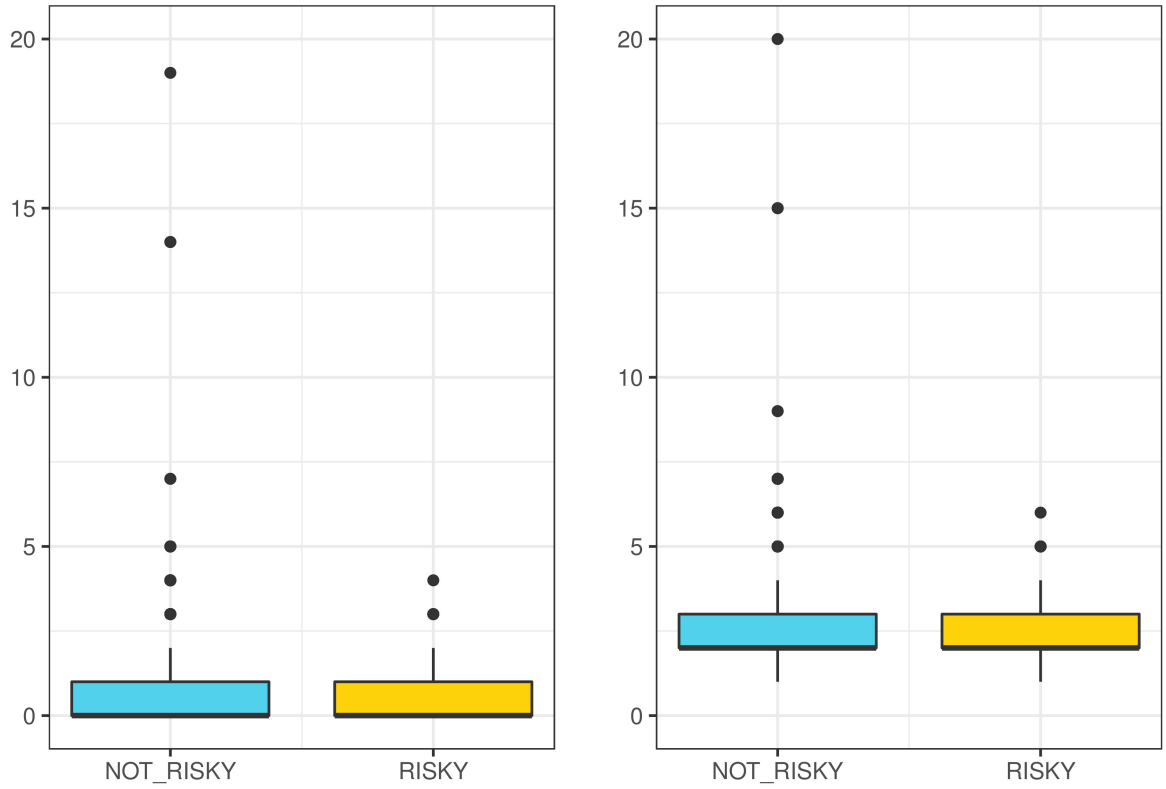
The difference between groups is also not conspicuous when looking at age, work tenure, or income data.



The yellow area shows the density of risky customers population with the blue area showing the population density for not-risky customers.

There is no linear difference, although the distributions are not overlapping perfectly. There are pockets of differences (e.g. shaded area for work tenure shows that people between 1 and 4 years at work tend to default more often) but they are few in numbers. Age seems to be more promising with three visible areas of difference (29-34, 34-47, and 47-57). Nonetheless, it's difficult to make any easy distinction without employing any mathematical model.

Finally, there are also characteristics corresponding to family sizes and numbers of children (naturally, correlated since children are considered family members).



Again, it's nigh impossible to make any definite decisions based on these features.

In conclusion, both groups of customers (risky vs not-risky) are very similar demographic-wise.

DATA PREPARATION

In order to construct a successful model, the data set has to be slightly transformed. First, binary categories stored as characters should be converted to Boolean factors ($\{0, 1\}$). We won't do it here because those will be dropped as they're not informative for the prediction model. Ultimately, the marginal utility from multiple features is decreasing as some of them are correlated or show low deviation across classes.

We do, however, want to employ a technique called one-hot encoding for categorical features. Since we cannot declare that housing situation or occupation can be ordered on a scale, and even if we could somehow order them (e.g. education level), they wouldn't necessarily have the same distance from each other (i.e. an academic degree is not equal twice as much as a high-school diploma). Those features have to be split between columns holding a binary value for each category (i.e. for education level each column for a different type of education will hold 0 except for one specific to the given customer).

```
# add one-hot encoding features for each categorical column
dummy <- dummyVars(" ~ FAMILY + EDUCATION + HOUSING + OCCUPATION + INCOME_SRC",
                   data=scoring_data)
m <- data.frame(predict(dummy, newdata=scoring_data))
clean_dataset <- bind_cols(m, scoring_data)

# get rid of unnecessary columns
clean_dataset <- clean_dataset %>%
```

```
  select(-c(
                        # remove categorical features
    "INCOME_SRC", "EDUCATION",
    "FAMILY", "HOUSING",
    "OCCUPATION", "ID",
    "DAYS_BIRTH", "DAYS_EMPLOYED",
    "CNT_CHILDREN", "CNT_FAM_MEMBERS",
                        # remove binary feature that do not add any value
    "FLAG_OWN_CAR", "FLAG_OWN_REALTY",
    "CODE_GENDER", "FLAG_MOBIL", "FLAG_WORK_PHONE",
    "FLAG_PHONE", "FLAG_EMAIL", "employed"
           ))
```

Many machine learning algorithms are sensitive to differences in scales, especially those based on calculating distances. For example, some models will be biased towards high values (e.g. $100,000 annual salary) at the expense of binary features (e.g. 0 or 1 denoting owning a real estate). To mitigate this effect, we can transform high values while keeping their relative distribution. There are multiple methods to achieve this goal but we will use the **min-max normalization** technique here.

$$\hat{x} = \frac{x - min(x)}{max(x) - min(x)}$$

```
# Normalize with a min-max function. Decision trees are pretty insensitive
# to scales of features, unlike linear models
clean_dataset <- clean_dataset %>%
  mutate(AMT_INCOME_TOTAL = (AMT_INCOME_TOTAL - mean(AMT_INCOME_TOTAL)) /
                             sd(AMT_INCOME_TOTAL),
         years_old = (years_old - mean(years_old)) /
                      sd(years_old),
         years_employed = (years_employed - mean(years_employed)) /
                           sd(years_employed))
```

The data set has to be split in order to avoid overfitting. The absolute minimum is to split the data between train set and test set, however, we will create three sets, the last one being a validation set. It will be used to evaluate the final model and confirm that the outcome was not obtained by pure chance. We will end up with three data set:

- train set - consisting of 80% of all observations - used for model training
- test set - consisting of 10% of all observations - used for model evaluation
- validation set - consisting of 10% of all observations - used for final model validation

With the above split, we will retain enough observations to successfully train the data set while keeping the ability to make a reliable comparison and decision.

```
# ensure reproducibility
set.seed(777, sample.kind="Rounding")
# train index for 80% of observations
train_index <- createDataPartition(clean_dataset$RISKY, p=0.8, list=FALSE)
# temporary set to split in half
temp_set <- slice(clean_dataset, -train_index)
# 10% test index
test_index <- createDataPartition(temp_set$RISKY, p=0.5, list=FALSE)
```

```
train_set <- slice(clean_dataset, train_index) # 80% train data set
test_set <- slice(temp_set, test_index)        # 10% test set
validation_set <- slice(temp_set, -test_index) # 10% validation set
```

The question remains, on what basis our models should be evaluated? Because the data set is imbalanced (i.e. one class makes up under 2% of the population), overall accuracy cannot be used. Otherwise, it is tempting to assign all predictions to the majority class, automatically yielding over 98% accuracy. This approach would naturally defeat the purpose of creating any machine learning models. Therefore, we will look at three different measures:

- F1 - the harmonic mean of precision and recall. It is often used with imbalanced data sets.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

- Specificity - because we know that the positive class denotes risky customers (the minority class), some models might be biased towards predicting most negative classes as positives when increasing recall. Therefore, we will also monitor specificity.

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

- Balanced accuracy - as mentioned above, the accuracy metric is not viable for this task, thus we will replace it with balanced accuracy which performs better with imbalanced data.

$$BalancedAccuracy = \frac{sensitivity + specificity}{2}$$

The best model will optimally increase all of the above metrics with the leading role of **Balanced Accuracy**.

METHODS

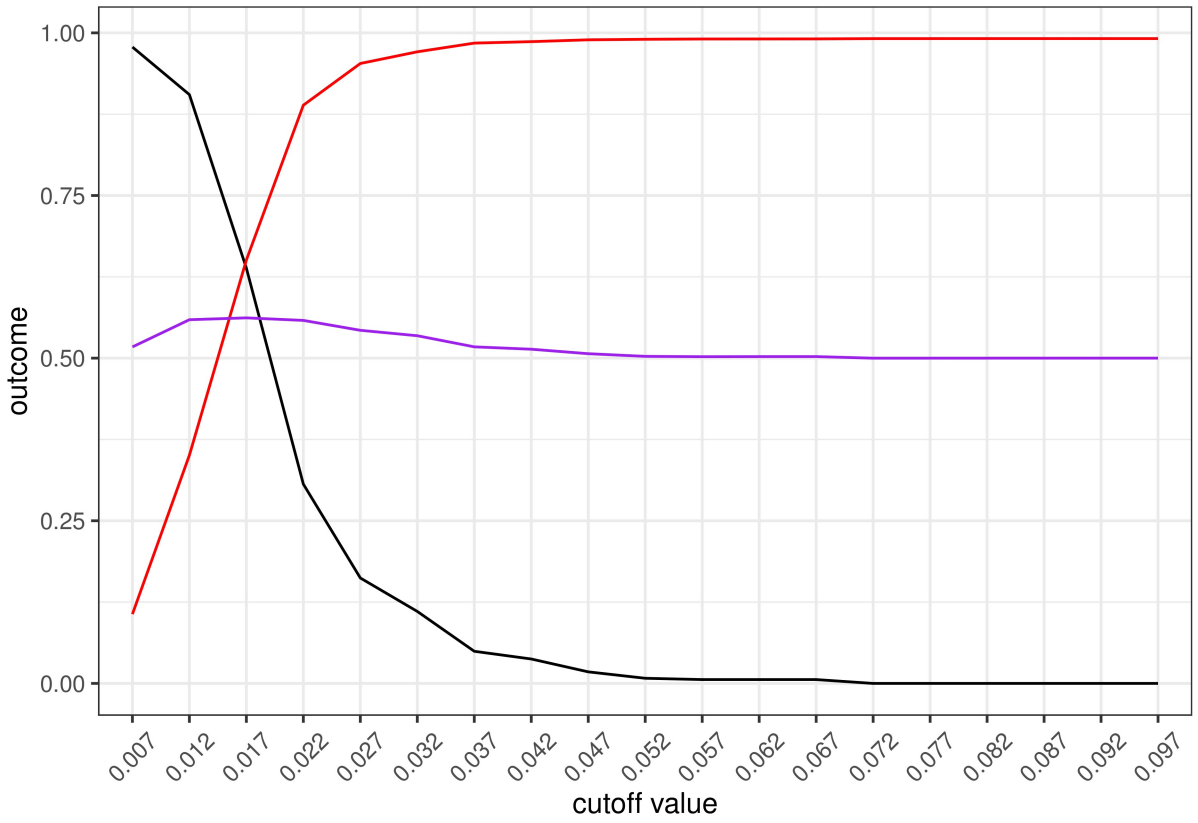
Linear Model

Linear models often prove to be very efficient and effective for many machine learning problems, therefore this will be the first model to train. Once the model is trained, we need to test different cutoff values for the model to decide on the class (i.e. outcome higher than x will be assigned to the positive class - risky customers).

```
fit_glm <- train(RISKY~.,                    # train linear model
                 data=train_set,
                 method="glm")

# Tune the cutoff to maximize F1, Specificity, and Balanced Accuracy
tune <- seq(0.007, 0.1, by=0.005)            # define tuning parameters
pred_glm <- predict(fit_glm, train_set)      # create predictions for the train set
```

```
# apply different cutoff values to predictions to find the best one
tune_glm <- sapply(tune, function(tune){
  cm <- confusionMatrix(as.factor(if_else(pred_glm >= tune, 1, 0)),
                        as.factor(train_set$RISKY),
                        mode="everything")
  cm$byClass[c("Specificity", "F1", "Balanced Accuracy")]
}) %>% as.data.frame()
```



The above chart shows the best cutoff value for the linear model is 0.017. Lower or higher cutoffs skew predictions towards one class or another. Higher cutoff lowers specificity (black line) while simultaneously increasing balanced accuracy (red line) because the model tends to assign more customers to the negative class (not a risky customer).

How does the model perform against this cutoff?

```
# test the performance after tuning
pred_glm <- predict(fit_glm, test_set)           # create predictions for the test set
cm_glm <- confusionMatrix(as.factor(if_else(pred_glm > 0.017, 1, 0)),
                as.factor(test_set$RISKY),
                mode="everything")

cm_glm$byClass[c("Specificity", "F1", "Balanced Accuracy")]
```

```
##       Specificity                F1 Balanced Accuracy
##         0.7551020         0.6551146         0.6219213
```

	0	1
0	1758	12
1	1839	37

Unfortunately, this model assigns too many observations to a positive class (False Positives) rendering it fairly useless.

Decision Tree

Hopefully, the decision tree model will do better. Due to their nature, decision trees are generally not sensitive to different scales of features but they can be pretty sensitive to imbalanced data. We will, therefore, attempt to apply a penalty to the classes so that the model doesn't treat them the same way. Since the minority class accounts for less than 1/50 of the data set (under 2%), I will apply a weight of 50.

```
# define weights so that the model doesn't predict the majority class for all cases
loss_mtx <- matrix(c(0,50,1,0), 2)

# The weight matrix
#     [,1] [,2]
#[1,]   0    1    POSITIVE
#[2,]   50   0    NEGATIVE
#       T    F

# fit the model
fit_tree <- train(as.factor(RISKY)~.,
                  data=train_set,
                  method="rpart",
                  parms=list(loss=loss_mtx)       # apply weights
                  )

# evaluate against the test set
cm_tree <- confusionMatrix(data=predict(fit_tree, test_set),
                reference=as.factor(test_set$RISKY),
                mode="everything")

# show relevant performance metrics
cm_tree$byClass[c("Specificity", "F1", "Balanced Accuracy")]
```

```
##       Specificity                F1 Balanced Accuracy
##         0.5918367         0.9100497         0.7157126
```

All three parameters score better with the decision tree model. How about the confusion matrix?

	0	1
0	3020	20
1	577	29

The Decision Tree model is significantly less biased towards the positive class (not risky customers), with fewer False Positive outcomes.

Random Forest

One way to further refine the Decision Tree model is to employ multiple decision trees, therefore the next model will be a Random Forest algorithm.

```
# fit random forest
fit_rf <- train(as.factor(RISKY)~.,
                data=train_set,
                method="rf",
                classwt=c(1,50),          # apply the same weights
                ntree=5)

cm_rf <- confusionMatrix(data=predict(fit_rf, test_set),
                reference=as.factor(test_set$RISKY),
                mode="everything")

cm_rf$byClass[c("Specificity", "F1", "Balanced Accuracy")]
```

```
##       Specificity                F1 Balanced Accuracy
##         0.4489796         0.9810207         0.7094773
```

Again, the outcome is slightly better. We have achieved a good increase in Specificity at a cost of slightly decreased F1 and Balanced Accuracy.

	0	1
0	3489	27
1	108	22

We can see that there are significantly fewer False Positive predictions (0s predicted as 1s). Alternatively to the previous Random Forest model with weights, we can mitigate the data imbalance with another technique, namely the **Synthetic Minority Oversampling Technique**. It is based on picking a random sample of the minority class and then oversampling the minority class so that the proportions are similar, thus removing the imbalance factor.

```
# over-sample the minority class
smote <- SMOTE(train_set, train_set$RISKY)
oversampled_set <- smote$data %>% select(-class)

# fit the random forest model again with the updated data set
fit_rf <- train(as.factor(RISKY)~.,
                data=oversampled_set,
                # keep the slightly lower weights because we want
                # to identify risky customers above else
                classwt=c(1,15),
                ntree=5,      # limit number of trees to decrease the run time
                method="rf"
                )

# evaluate on the test set
smote_rf_cm <- confusionMatrix(data=predict(fit_rf, test_set),
                reference=as.factor(test_set$RISKY),
                mode="everything")

smote_rf_cm$byClass[c("Specificity", "F1", "Balanced Accuracy")]
```

```
##       Specificity                F1 Balanced Accuracy
##         0.5510204         0.9795573         0.7584126
```

	0	1
0	3474	22
1	123	27

This approach seems to yield the best results so far.

RESULTS

Now, we can answer the fundamental question: Which model performs the best given the goal defined at the beginning? We can answer this question with a quick overview of all the outcomes.

	Specificity	F1	Balanced.Accuracy
Random Forest with oversampling	0.5510204	0.9795573	0.7584126
Random Forest with weights	0.4489796	0.9810207	0.7094773
Decision Tree	0.5918367	0.9100497	0.7157126
Linear Model	0.7551020	0.6551146	0.6219213

Clearly, the best model is the random forest with over-sampled data set. The last remaining step is to evaluate this model against the validation set to see how it performs and if the outcome is not a product of pure chance.

```
final_cm <- confusionMatrix(data=predict(fit_rf, validation_set),
                reference=as.factor(validation_set$RISKY),
                mode="everything")
final_cm$byClass[c("Specificity", "F1", "Balanced Accuracy")]
```

```
##       Specificity                F1 Balanced Accuracy
##         0.4918033         0.9826931         0.7330668
```

	0	1
0	3492	31
1	92	30

The outcome is within expectations, proving that the model is not overfitting nor that the outcome has been reached by chance.

CONCLUSIONS

The project's goal was to predict which customers are more likely to default on their debt at some point in the future based on the given data set. The business goal has been defined as predicting customers likely to default on their credit card payments for longer than 60 days. Specifically, the goal was to construct a model that optimally maximizes the Specificity, F1, and most importantly Balanced Accuracy metrics.

The best model for this task proved to be the Random Forest with an oversampled data set. It is, however, not ideal and could be further refined in the future, primarily by improving the data set.

The main constraint for the project was the data imbalance, which led to classifiers being biased towards the majority class. In some extreme cases, the classifier may assign all test cases to the majority class.

Another limitation was due to the low predictive value of given customers' characteristics. The two groups of customers (those at risk of default and the others) share similar characteristics.

The model can be future refined based on the business objective and the organization's risk appetite (e.g. due to macroeconomic situation). Customers predicted as at risk of default do not have to necessarily be rejected but may be routed for further manual screening and possibly lowering their available credit balance.

Another way to improve the outcome is to work with other institutions to gather more data on defaulting customers or even consider asking for additional personal information to gather more insightful features.