Home Credit EDA

Rylan Tribush

2024-04-28

Contents

Introduction

Home Credit Group seeks to provide home loans to currently underserved communities, but it only wants to give loans to potential clients with the ability to repay. It is difficult to predict if someone with a limited credit history will be able to repay a loan. Therefore, the overall purpose of this project is to develop a model to predict whether an applicant will repay, based on historical data. The outcome variable is binary, so this is considered a supervised classification problem. This notebook won't dive into any machine learning algorithms; it's scope is limited to examining and understanding the existing home loan data. The data set and the original case competition on which this business problem is based can be found at Kaggle using this link.

Data

While Home Credit has compiled seven different files of prior customer credit data, we will focus our analysis on the **application_train.csv** file. This csv file is the main data set that contains the target variable that we care about. Here's a brief summary of the other data: The file **application_test.csv** contains the data that we will test the model on. It is the same as the training data, except it doesn't have a column for the target variable. The **bureau.csv** file has data on previous loans that were issued by other companies that reported to the credit bureau and **bureau_balance.csv** has monthly balance data from the bureau. The **previous_application.csv** file contains data from previous loans made by Home Credit. Finally, **POS_CASH_balance.csv**, **installments_payments.csv**, and **credit_card_balance.csv** have monthly data on loan balances, payment, and credit card balances for previous Home Credit customers.

Let's begin by taking a quick look at the data from the main training data set:

```
# import data
train_data <- read_csv("application_train.csv", show_col_types = FALSE)

# size of data set
dim(train_data)
```

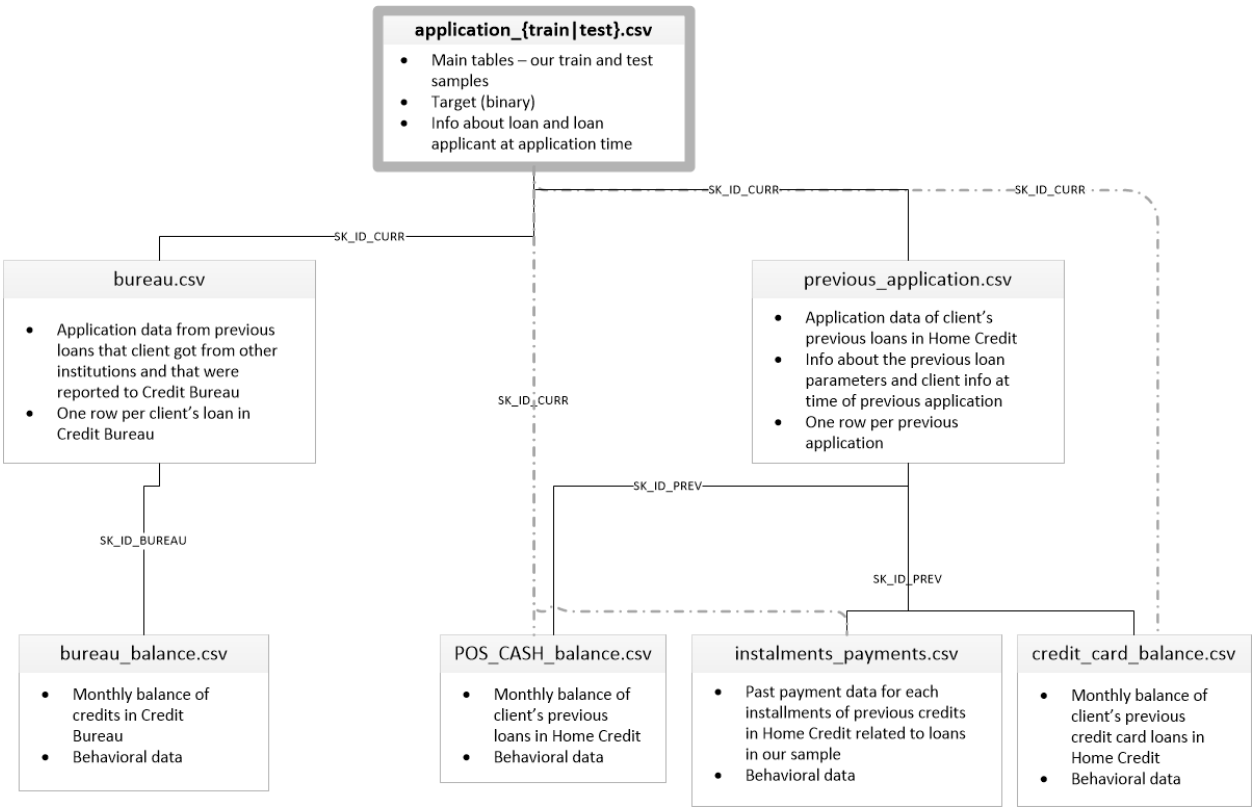


Figure 1: Diagram of Data Files

```
## [1] 307511    122
```

```
# look at first few rows
head(train_data[,1:5])
```

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR
100002	1	Cash loans	M	N
100003	0	Cash loans	F	N
100004	0	Revolving loans	M	Y
100006	0	Cash loans	F	N
100007	0	Cash loans	M	N
100008	0	Cash loans	M	N

The training data set has roughly 310,000 observations, with each row representing a loan. The data set has 122 columns, including our target variable, conveniently named "TARGET."

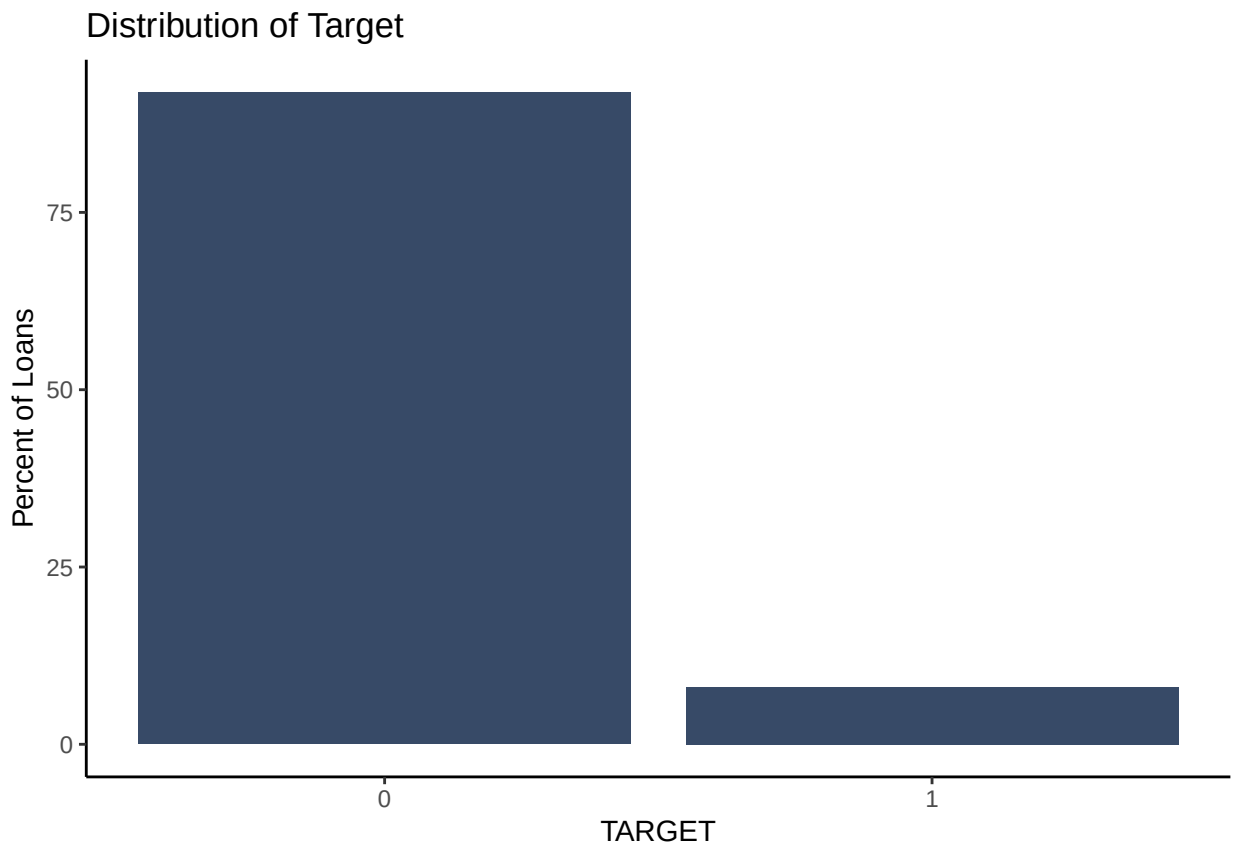
Let's take a look at the distribution of the target variable.

```
# frequency table of TARGET
train_data %>% count(TARGET)
```

TARGET	n
0	282686
1	24825

Most loans were paid back on time, so the distribution of the target variable is imbalanced. We can visualize this with a chart.

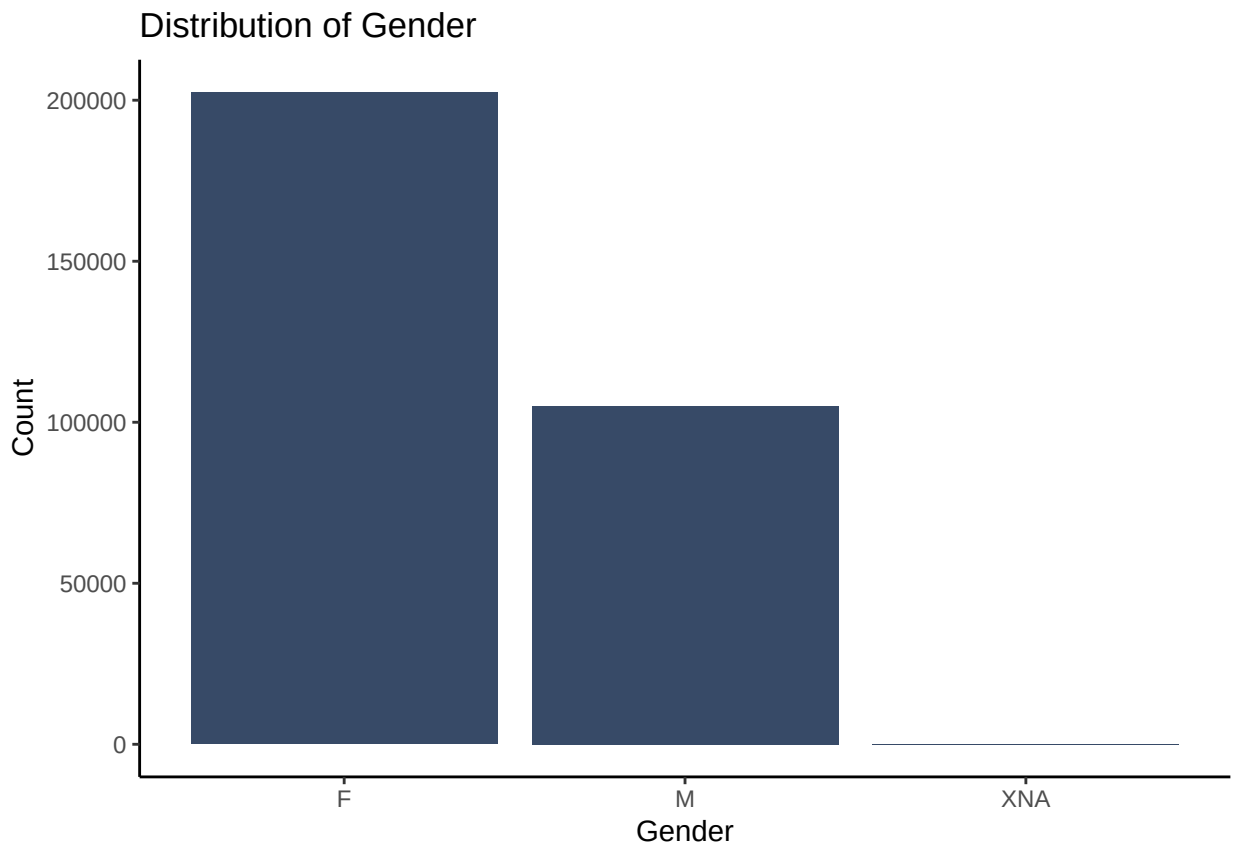
```
# frequency chart of TARGET
ggplot(data = train_data %>% group_by(TARGET) %>%
         summarize(percent = n()/nrow(train_data)),
       aes(y = percent * 100, x = TARGET)) +
  geom_col(fill = "#374A67") +
  theme_classic() +
  scale_x_continuous(breaks = c(0,1)) +
  labs(y = "Percent of Loans", title = "Distribution of Target")
```



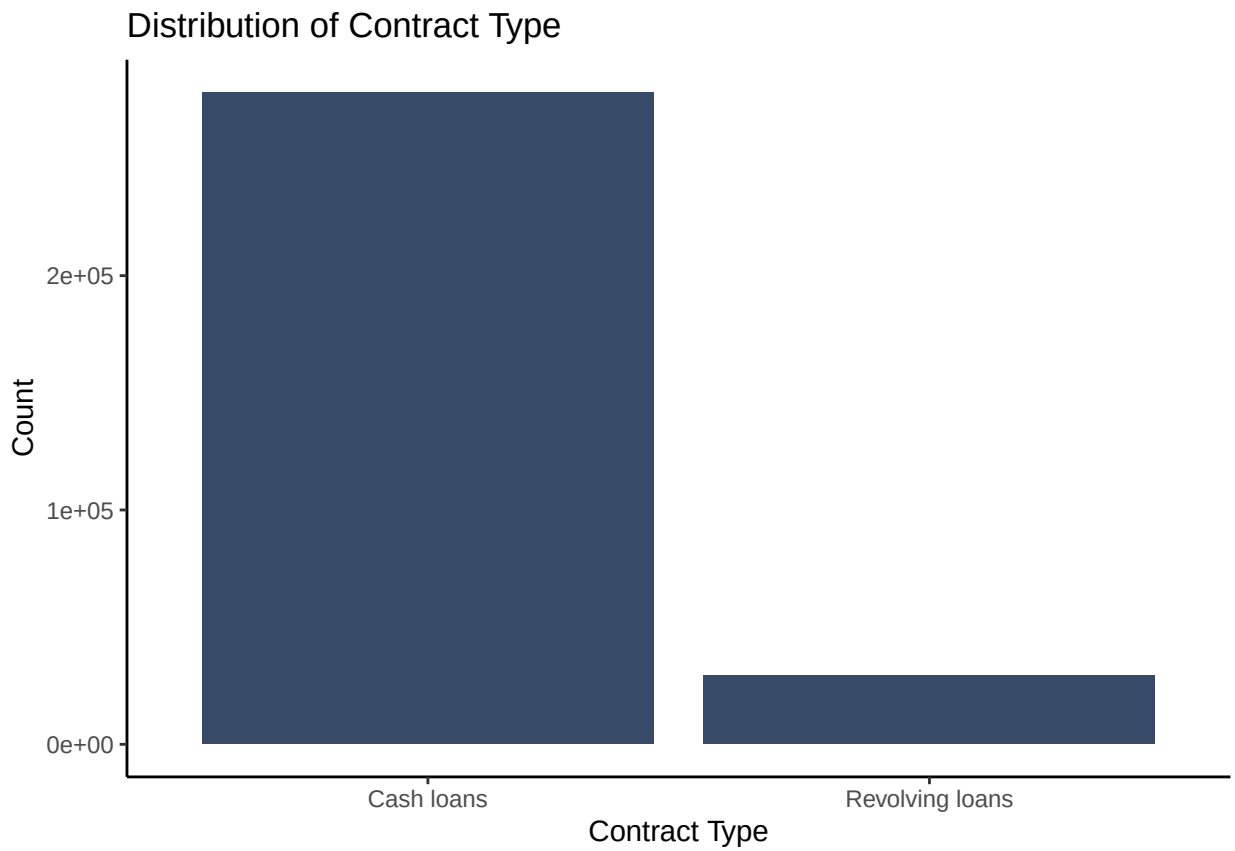
It will be important for us to keep this imbalance in mind when assessing the accuracy of any potential models. Only about 8% of the loans were not paid on time, so a naive model that predicts 0 every time will still have high accuracy. When considering different machine learning algorithms, we will need to research which ones are good at handling imbalances like this.

Now let's take a look at some of the feature variables.

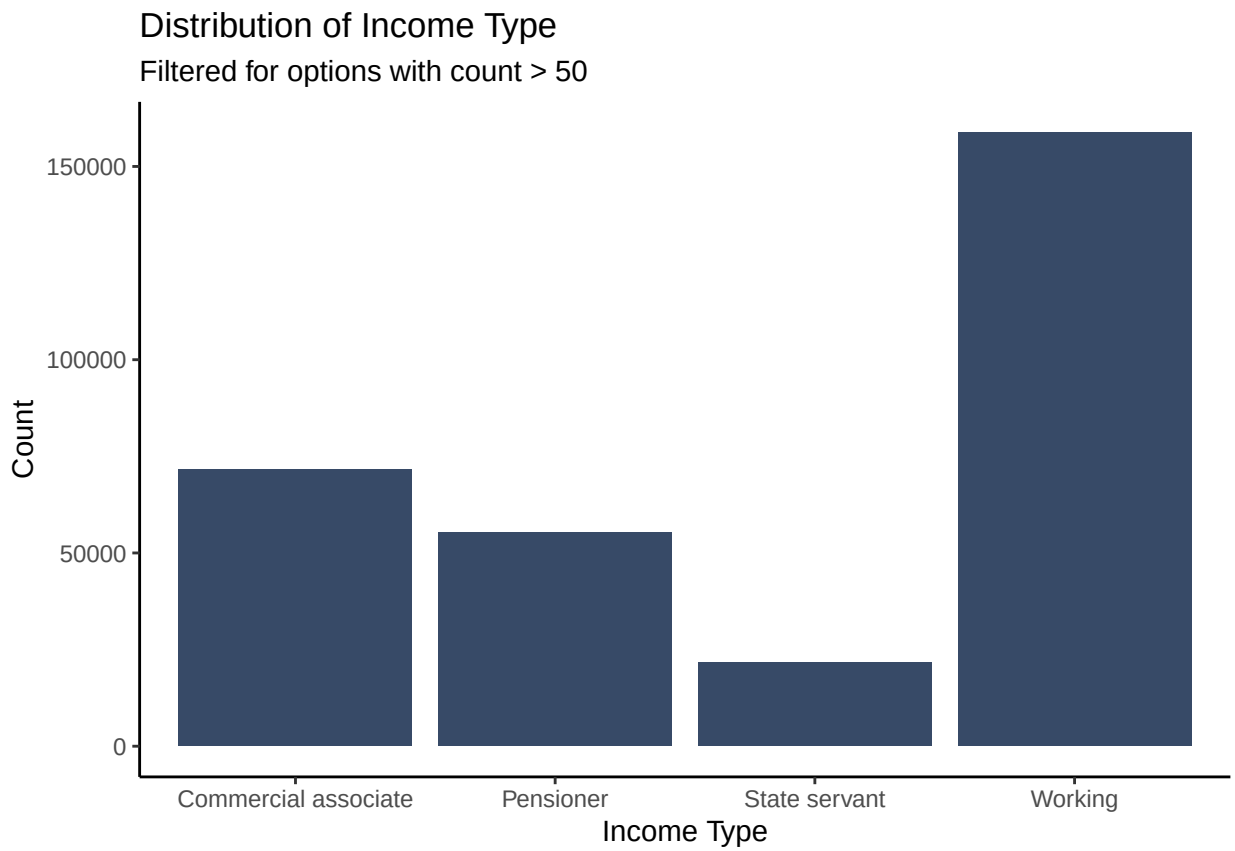
```
# Bar chart of Gender
train_data %>%
  count(CODE_GENDER) %>%
  ggplot(aes(x = CODE_GENDER, y = n)) +
    geom_col(fill = "#374A67") +
    theme_classic() +
    labs(
      x = "Gender",
      y = "Count",
      title = "Distribution of Gender"
    )
```




```
# Bar chart of Contract Type
train_data %>%
  count(NAME_CONTRACT_TYPE) %>%
  ggplot(aes(x = NAME_CONTRACT_TYPE, y = n)) +
    geom_col(fill = "#374A67") +
    theme_classic() +
    labs(
      x = "Contract Type",
      y = "Count",
      title = "Distribution of Contract Type"
    )
```




```
# Bar chart of Income Type
train_data %>%
  count(NAME_INCOME_TYPE) %>%
  filter(n > 50) %>% # remove uncommon options from chart
  ggplot(aes(x = NAME_INCOME_TYPE, y = n)) +
    geom_col(fill = "#374A67") +
    theme_classic() +
    labs(
      x = "Income Type",
      y = "Count",
      title = "Distribution of Income Type",
      subtitle = "Filtered for options with count > 50"
    )
```



Discussion of Missing Data

A quick look at the data file shows that a good portion of the columns are missing data. We can get a better sense of the problem by counting the number of NAs in each column.

```
kable(as_tibble(freq.na(train_data), rownames = NA) %>%
  rownames_to_column() %>%
  filter(missing > 0) %>%
  rename("variable" = rowname))
```

variable	missing	%
COMMONAREA_AVG	214865	70
COMMONAREA_MODE	214865	70
COMMONAREA_MEDI	214865	70
NONLIVINGAPARTMENTS_AVG	213514	69
NONLIVINGAPARTMENTS_MODE	213514	69
NONLIVINGAPARTMENTS_MEDI	213514	69
FONDKAPREMONT_MODE	210295	68
LIVINGAPARTMENTS_AVG	210199	68
LIVINGAPARTMENTS_MODE	210199	68
LIVINGAPARTMENTS_MEDI	210199	68
FLOORSMIN_AVG	208642	68
FLOORSMIN_MODE	208642	68
FLOORSMIN_MEDI	208642	68
YEARS_BUILD_AVG	204488	66

variable	missing	%
YEARS_BUILD_MODE	204488	66
YEARS_BUILD_MEDI	204488	66
OWN_CAR_AGE	202929	66
LANDAREA_AVG	182590	59
LANDAREA_MODE	182590	59
LANDAREA_MEDI	182590	59
BASEMENTAREA_AVG	179943	59
BASEMENTAREA_MODE	179943	59
BASEMENTAREA_MEDI	179943	59
EXT_SOURCE_1	173378	56
NONLIVINGAREA_AVG	169682	55
NONLIVINGAREA_MODE	169682	55
NONLIVINGAREA_MEDI	169682	55
ELEVATORS_AVG	163891	53
ELEVATORS_MODE	163891	53
ELEVATORS_MEDI	163891	53
WALLSMATERIAL_MODE	156341	51
APARTMENTS_AVG	156061	51
APARTMENTS_MODE	156061	51
APARTMENTS_MEDI	156061	51
ENTRANCES_AVG	154828	50
ENTRANCES_MODE	154828	50
ENTRANCES_MEDI	154828	50
LIVINGAREA_AVG	154350	50
LIVINGAREA_MODE	154350	50
LIVINGAREA_MEDI	154350	50
HOUSETYPE_MODE	154297	50
FLOORSMAX_AVG	153020	50
FLOORSMAX_MODE	153020	50
FLOORSMAX_MEDI	153020	50
YEARS_BEGINEXPLUATATION_AVG	150007	49
YEARS_BEGINEXPLUATATION_MODE	150007	49
YEARS_BEGINEXPLUATATION_MEDI	150007	49
TOTALAREA_MODE	148431	48
EMERGENCYSTATE_MODE	145755	47
OCCUPATION_TYPE	96391	31
EXT_SOURCE_3	60965	20
AMT_REQ_CREDIT_BUREAU_HOUR	41519	14
AMT_REQ_CREDIT_BUREAU_DAY	41519	14
AMT_REQ_CREDIT_BUREAU_WEEK	41519	14
AMT_REQ_CREDIT_BUREAU_MON	41519	14
AMT_REQ_CREDIT_BUREAU_QRT	41519	14
AMT_REQ_CREDIT_BUREAU_YEAR	41519	14
NAME_TYPE_SUITE	1292	0
OBS_30_CNT_SOCIAL_CIRCLE	1021	0
DEF_30_CNT_SOCIAL_CIRCLE	1021	0
OBS_60_CNT_SOCIAL_CIRCLE	1021	0
DEF_60_CNT_SOCIAL_CIRCLE	1021	0
EXT_SOURCE_2	660	0
AMT_GOODS_PRICE	278	0
AMT_ANNUITY	12	0
CNT_FAM_MEMBERS	2	0

variable	missing	%
DAYS_LAST_PHONE_CHANGE	1	0

This table shows that there is a lot of missing data. 67 of the 122 columns have at least one NA, and 44 of those columns have at least 50% NAs. Since we're only exploring the data right now, not building any models, I'm going to leave the missing data as it is. However, for the modelling process, we have number of different options. The easiest option is to just drop the rows with missing data. This would probably be a mistake since it would leave us with a lot less data to work with and possibly bias the results. Another option is to ignore the columns with missing data and just build models with the other 55 features. Depending on our goals, and how complex we want the model to be, this could be a decent option. 55 features is a lot to work with, but ignoring the other columns seems like a waste of valuable data. A third option would be to use some kind of imputation method to generate values for the missing data. This would probably be the most work, but could improve our results if we are able to come up with a strategy that doesn't introduce bias. During the modelling part of the project, my current plan is to use a combination of the second and third options: I will try use imputation where useful and drop the columns when necessary.

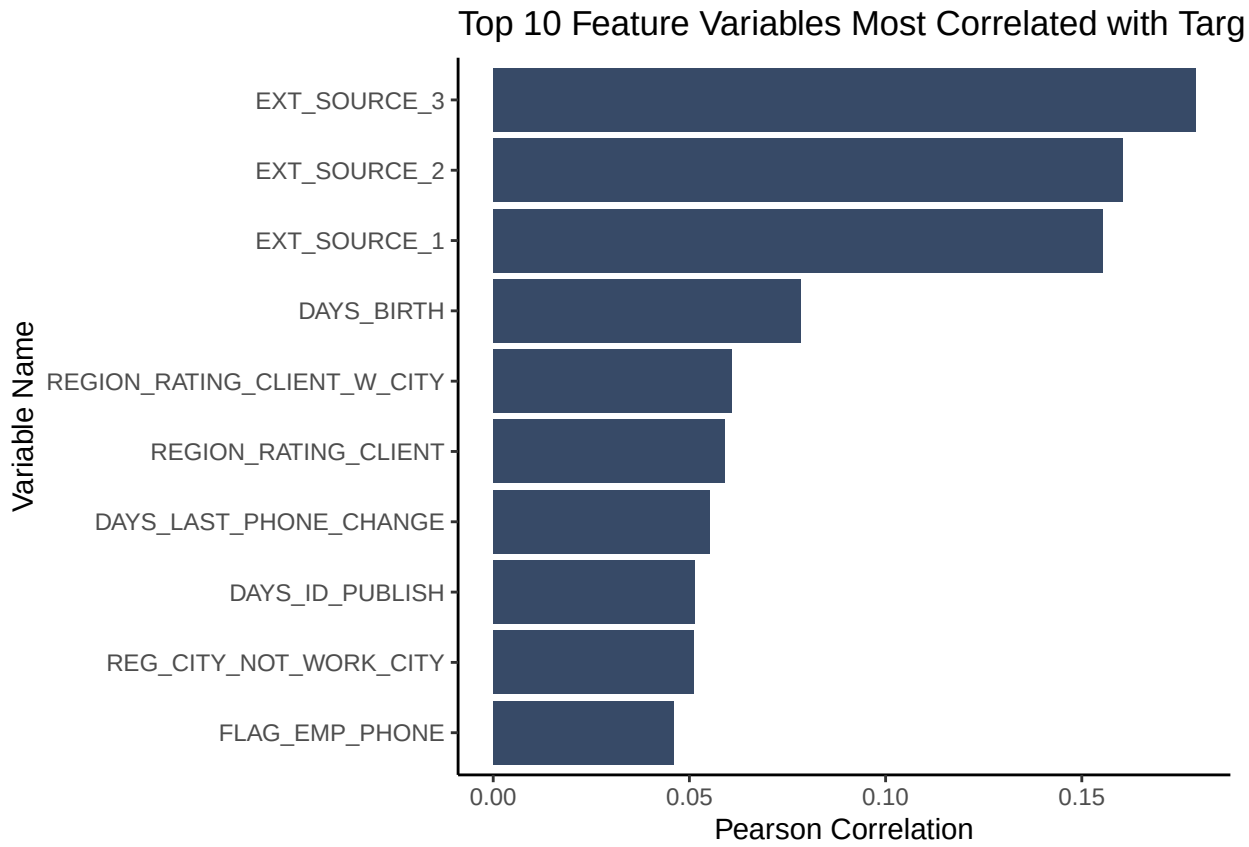
Data Exploratoration

To get a better sense of which variables might be important, we can take a look at the correlation between TARGET and the numeric variables.

```
# generate correlation table
cors <- train_data %>%
  correlate(quiet = TRUE) %>%
  focus(TARGET) %>%
  rename("variable" = term) %>%
  rename("correlation" = TARGET) %>%
  mutate(magnitude_of_correlation = abs(correlation)) %>%
  arrange(0 - magnitude_of_correlation)
```

```
## Warning in stats::cor(x = x, y = y, use = use, method = method): the standard
## deviation is zero
```

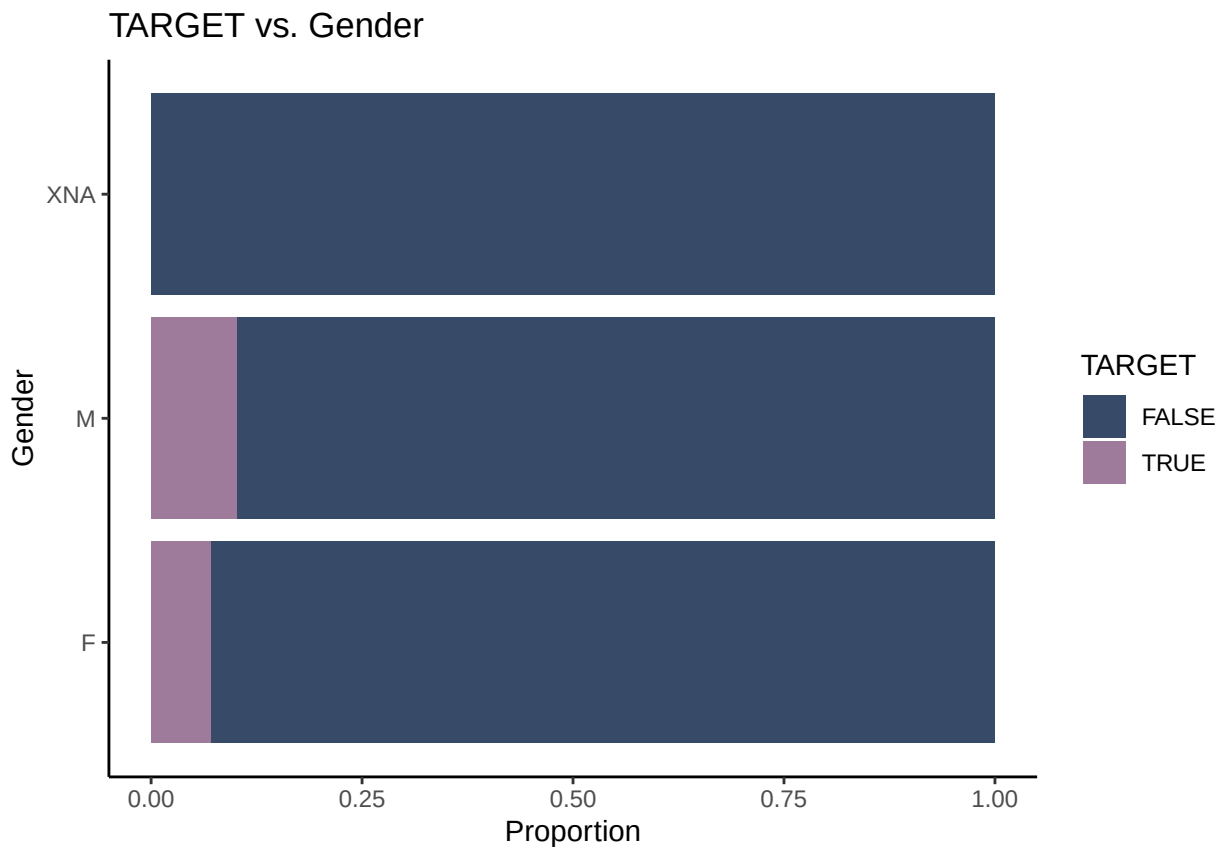
```
ggplot(data = head(cors, 10),
       aes(y = reorder(variable, +magnitude_of_correlation),
           x = magnitude_of_correlation)) +
  geom_bar(stat = "identity", fill = "#374A67") +
  theme_classic() +
  labs(title = "Top 10 Feature Variables Most Correlated with Target",
       x = "Pearson Correlation",
       y = "Variable Name")
```



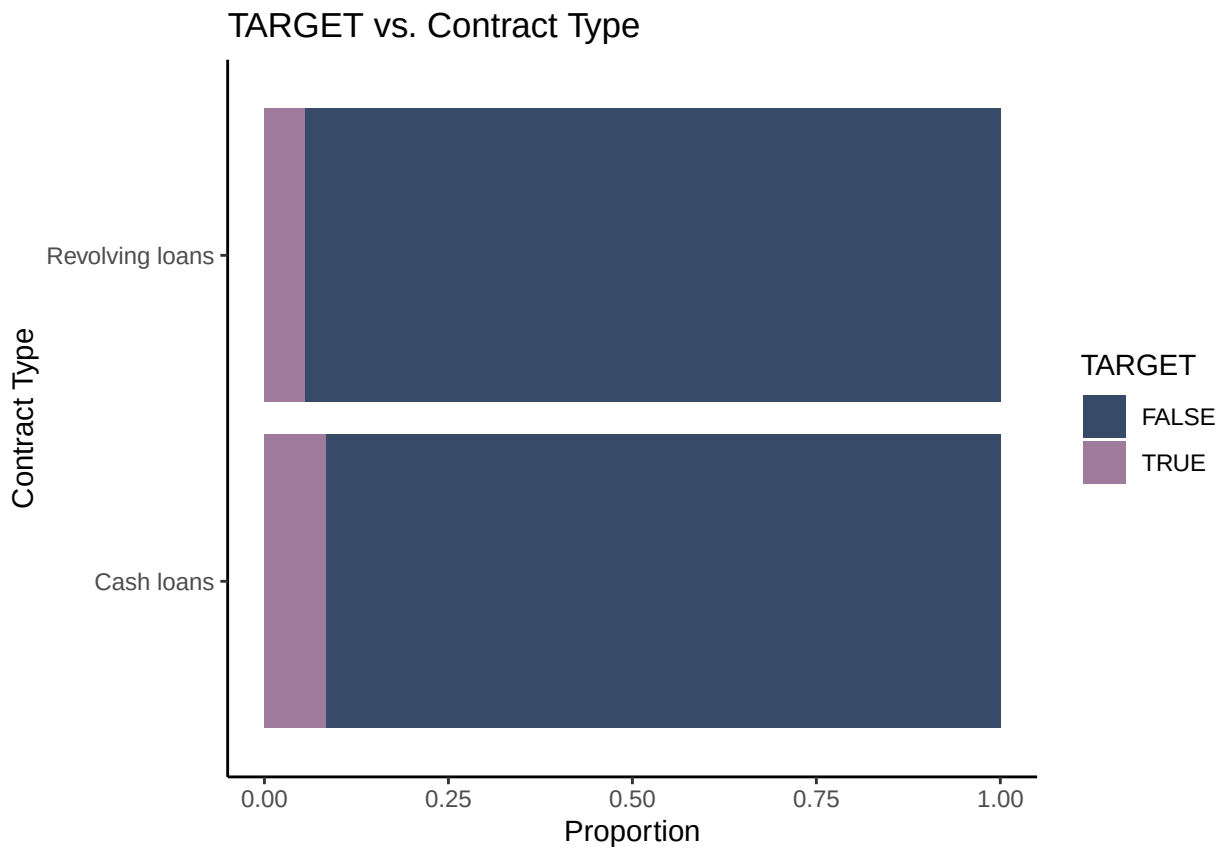
It looks like the strongest correlations are with EXT_SOURCE_3, EXT_SOURCE_2, and EXT_SOURCE_1, which are all normalized scores from external data sources (credit bureaus?). Another promising variable is DAYS_BIRTH, which is the client's age in days at the time of application.

We can also take a look at the relationship between TARGET and some of the categorical variables.

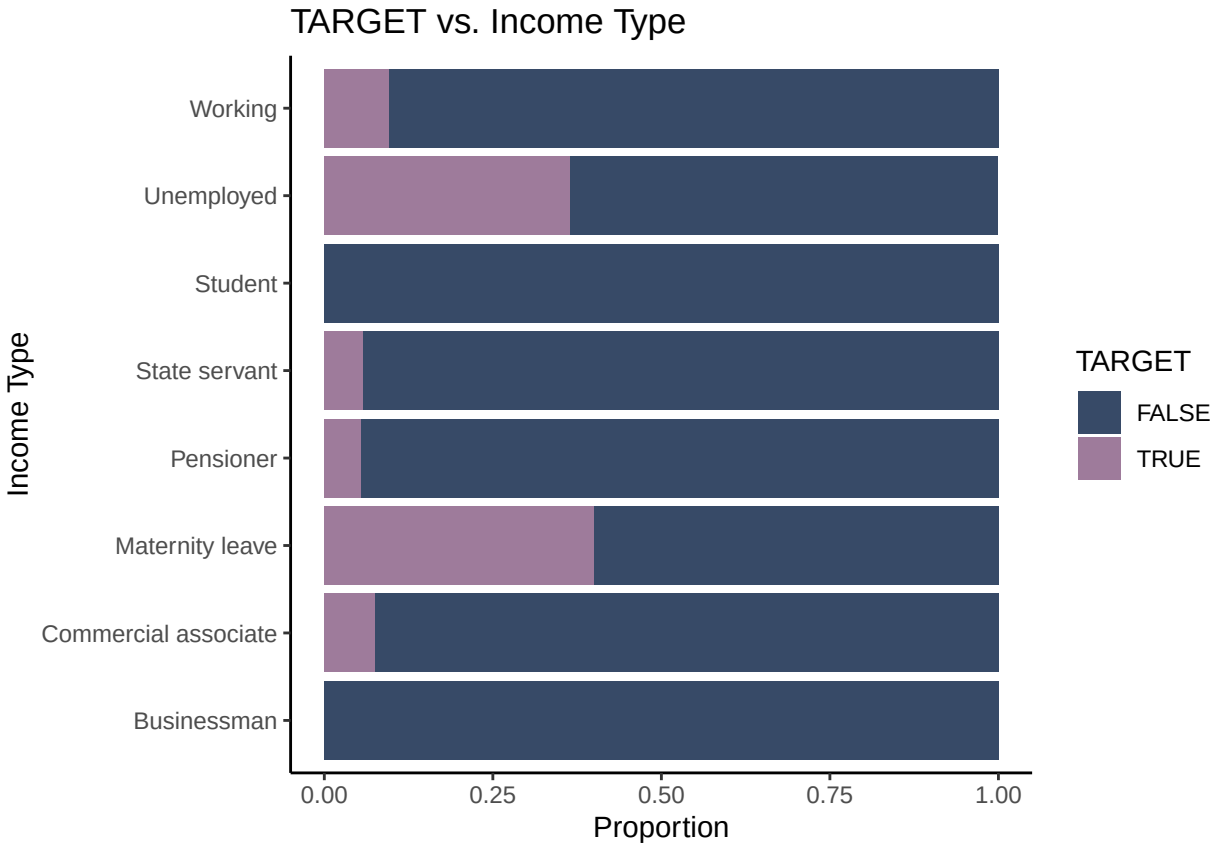
```
# plot of TARGET vs Gender
ggplot(data = train_data, aes(y = CODE_GENDER,
                              fill = if_else(TARGET == 0, F, T))) +
  geom_bar(position = "fill") +
  theme_classic() +
  scale_fill_manual(values = c("#374A67", "#9E7B9B")) +
  labs(
    y = "Gender",
    x = "Proportion",
    fill = "TARGET",
    title = "TARGET vs. Gender"
  )
```




```
# plot of TARGET vs Contract Type
ggplot(data = train_data, aes(y = NAME_CONTRACT_TYPE,
                              fill = if_else(TARGET == 0, F, T))) +
  geom_bar(position = "fill") +
  theme_classic() +
  scale_fill_manual(values = c("#374A67", "#9E7B9B")) +
  labs(
    y = "Contract Type",
    x = "Proportion",
    fill = "TARGET",
    title = "TARGET vs. Contract Type"
  )
```




```
# plot of TARGET vs Income Type
ggplot(data = train_data, aes(y = NAME_INCOME_TYPE,
                              fill = if_else(TARGET == 0, F, T))) +
  geom_bar(position = "fill") +
  theme_classic() +
  scale_fill_manual(values = c("#374A67", "#9E7B9B")) +
  labs(
    y = "Income Type",
    x = "Proportion",
    fill = "TARGET",
    title = "TARGET vs. Income Type"
  )
```



Out of the categorical variables that we looked at here, it looks like Income Type will probably be the most useful, since people who are Unemployed or on Maternity leave have a noticeably higher rate of problems with their loans.

Results

In summary, this notebook provided a brief exploration of the Home Credit Default data. We got a sense of the structure of the various csv files, and got a better understanding of the layout of the primary training data. We visualized the distribution of the target variable and some of the feature variables. Additionally, we examined the extent of the missing data and discussed some possible approaches to use during the modeling process. Finally, we concluded the exploratory data analysis by looking at the relationships between the target variable and the feature variables and identified the ones that have a stronger correlation.

Resources Used

R for Data Science (2e) https://r4ds.hadley.nz

Kaggle Public Notebooks https://www.kaggle.com/c/home-credit-default-risk/code

Coolors https://coolors.co/0e1116-374a67-616283-9e7b9b-cb9cf2