R Notebook

Alexia Wells

2024-09-28

- Business Statement
- Exploration Tasks
    - Cleaning the Data
        - How Does the Initial Data Look?
        - What is the Distribution of the Target Variable?
        - Are there Columns with Near-Zero Variance?
        - What is the Scope of missing data in application_{train|test}.csv?
        - Are there String Errors in the Data?
        - How Can I Reorganize the Organization Type Variable?
        - Do the Gender Levels Line Up in Train and Test?
    - Feature Engineering
        - Car Age Binning Variable
        - Housing Info Binning Variable
        - Amount of Children Applicant Has
    - Data Imputation
        - Mean/Mode Strategy
        - Are There More Near Zero Variance Predictors?
        - How to Approach Imbalanced Data?
- Modeling - Which Predictors are Significant?
    - Simple Models
        - Does Car Ownership Predict Default?
        - Does Car Age Predict Default?
        - Does Count Children Bin Predict Default?
    - Complex Model
        - Finalized Complex Logistic Regression Model
- Results
    - Future Plans

Business Statement

Individuals with insufficient credit histories often struggle to receive a loan from trusted lenders. Consequently, many individuals in a similar situation are taken advantage of and ultimately exploited. Home Credit aims to create a safe loaning environment for that population while being profitable for the company. The purpose of this project is to reasonably maintain inclusivity and accessibility while predicting which applicants will repay their loans.

This will be a predictive analytics project. The likelihood of customers repaying their loans will be generated using a classification algorithm. The data contains many columns of customer demographics. For example, the client's gender, whether they have owned a car, how many children they have, their highest level of education received, etc. The target variable is whether customers repay their loans, which is binary (yes or no).

The purpose of this EDA notebook is to take a look at the data and get an initial understanding it. That means variable distributions and relationships will be explored. One of my main goals is to identify issues that may come up in the modeling process and consequently making any necessary fixes.

Exploration Tasks

```
# Load libraries
library(tidyverse)
library(dplyr)
library(DataExplorer)
library(ggplot2)
library(caret)
library(skimr)
library(tidymodels)
library(patchwork)
library(missForest)
library(kableExtra)
library(ROSE)
library(car)
library(ggcorrplot)
```

Cleaning the Data

How Does the Initial Data Look?

I first read in application_train and test csv. The train is a big dataset with 122 columns and 307511 rows. My first observation was the amount of categorical variables that needed to be factored. The good news is that there is no duplicated data in the train set.

```
# Read in data
app_train <- vroom::vroom("application_train.csv")
app_test <- vroom::vroom("application_test.csv")
```

```
# Check for duplicated data
anyDuplicated(app_train)

# Check data type of variables
str(app_train)
```

```
# Change categorical variables to factors
app_train <- app_train |>
  mutate(across(c(TARGET, NAME_CONTRACT_TYPE, CODE_GENDER, FLAG_OWN_CAR,
                  CNT_CHILDREN,NAME_TYPE_SUITE,NAME_INCOME_TYPE,
                  NAME_EDUCATION_TYPE,NAME_FAMILY_STATUS,NAME_HOUSING_TYPE,
                  FLAG_MOBIL, FLAG_EMP_PHONE, FLAG_WORK_PHONE, FLAG_CONT_MOBILE,
                  FLAG_PHONE,FLAG_EMAIL, OCCUPATION_TYPE, CNT_FAM_MEMBERS,
                  REGION_RATING_CLIENT, REGION_RATING_CLIENT_W_CITY,
                  WEEKDAY_APPR_PROCESS_START, HOUR_APPR_PROCESS_START,
                  REG_REGION_NOT_LIVE_REGION, REG_REGION_NOT_WORK_REGION,
                  LIVE_REGION_NOT_WORK_REGION,REG_CITY_NOT_LIVE_CITY,
                  REG_CITY_NOT_WORK_CITY, LIVE_CITY_NOT_WORK_CITY, ORGANIZATION_TYPE,
                  FONDKAPREMONT_MODE, HOUSETYPE_MODE, WALLSMATERIAL_MODE,
                  EMERGENCYSTATE_MODE, OBS_30_CNT_SOCIAL_CIRCLE,
                  OBS_60_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE,
                  DEF_60_CNT_SOCIAL_CIRCLE, FLAG_OWN_REALTY, FLAG_DOCUMENT_2,
                  FLAG_DOCUMENT_3, FLAG_DOCUMENT_4, FLAG_DOCUMENT_5, FLAG_DOCUMENT_6,
                  FLAG_DOCUMENT_7,FLAG_DOCUMENT_8, FLAG_DOCUMENT_9, FLAG_DOCUMENT_10,
                  FLAG_DOCUMENT_11, FLAG_DOCUMENT_12, FLAG_DOCUMENT_13,
                  FLAG_DOCUMENT_14, FLAG_DOCUMENT_15, FLAG_DOCUMENT_16,
                  FLAG_DOCUMENT_17,FLAG_DOCUMENT_18,FLAG_DOCUMENT_19,
                  FLAG_DOCUMENT_20,FLAG_DOCUMENT_21), as.factor))

# Do the same for test
app_test <- app_test |>
    mutate(across(c(NAME_CONTRACT_TYPE, CODE_GENDER, FLAG_OWN_CAR,
                  CNT_CHILDREN,NAME_TYPE_SUITE,NAME_INCOME_TYPE,
                  NAME_EDUCATION_TYPE,NAME_FAMILY_STATUS,NAME_HOUSING_TYPE,
                  FLAG_MOBIL, FLAG_EMP_PHONE, FLAG_WORK_PHONE, FLAG_CONT_MOBILE,
                  FLAG_PHONE,FLAG_EMAIL, OCCUPATION_TYPE, CNT_FAM_MEMBERS,
                  REGION_RATING_CLIENT, REGION_RATING_CLIENT_W_CITY,
                  WEEKDAY_APPR_PROCESS_START, HOUR_APPR_PROCESS_START,
                  REG_REGION_NOT_LIVE_REGION, REG_REGION_NOT_WORK_REGION,
                  LIVE_REGION_NOT_WORK_REGION,REG_CITY_NOT_LIVE_CITY,
                  REG_CITY_NOT_WORK_CITY, LIVE_CITY_NOT_WORK_CITY, ORGANIZATION_TYPE,
                  FONDKAPREMONT_MODE, HOUSETYPE_MODE, WALLSMATERIAL_MODE,
                  EMERGENCYSTATE_MODE, OBS_30_CNT_SOCIAL_CIRCLE,
                  OBS_60_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE,
                  DEF_60_CNT_SOCIAL_CIRCLE, FLAG_OWN_REALTY, FLAG_DOCUMENT_2,
                  FLAG_DOCUMENT_3, FLAG_DOCUMENT_4, FLAG_DOCUMENT_5, FLAG_DOCUMENT_6,
                  FLAG_DOCUMENT_7,FLAG_DOCUMENT_8, FLAG_DOCUMENT_9, FLAG_DOCUMENT_10,
                  FLAG_DOCUMENT_11, FLAG_DOCUMENT_12, FLAG_DOCUMENT_13,
                  FLAG_DOCUMENT_14, FLAG_DOCUMENT_15, FLAG_DOCUMENT_16,
                  FLAG_DOCUMENT_17,FLAG_DOCUMENT_18,FLAG_DOCUMENT_19,
                  FLAG_DOCUMENT_20,FLAG_DOCUMENT_21), as.factor))
```

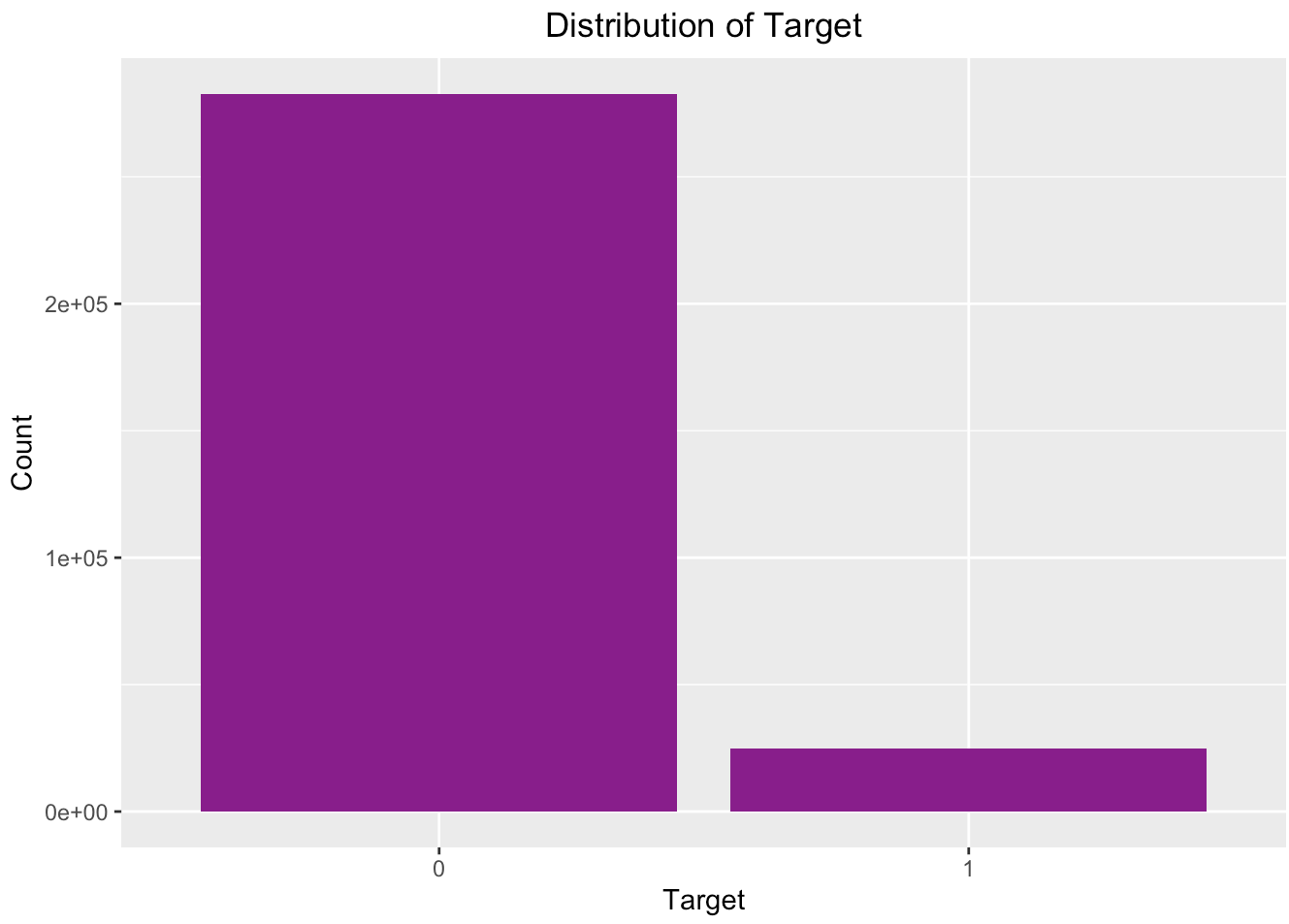
What is the Distribution of the Target Variable?

The target variable is highly unbalanced. The count of 0s is 257,861 observations greater than 1s. This is easy to see in the outputted count table and bar graph.

```
# Checking balance of target variable
app_train |>
  group_by(TARGET) |>
  rename("Target" = TARGET) |>
  summarize(Count = n()) |>
  kbl() |>
  kable_styling()
```

Target	Count
0	282686
1	24825

```
# Bar Graph of Target Variable
ggplot(data = app_train, mapping = aes(x = TARGET)) +
  geom_bar(fill = "magenta4") + xlab("Target") + ylab("Count") +
  ggtitle("Distribution of Target") + theme(plot.title = element_text(hjust = 0.5))
```



Are there Columns with Near-Zero Variance?

There were many predictors with near-zero variance. I used a function that helped me identify exactly which variables fell into that category. I removed them from the data frame, so that there was only 86 columns left. This is nice because it makes the data slightly more approachable.

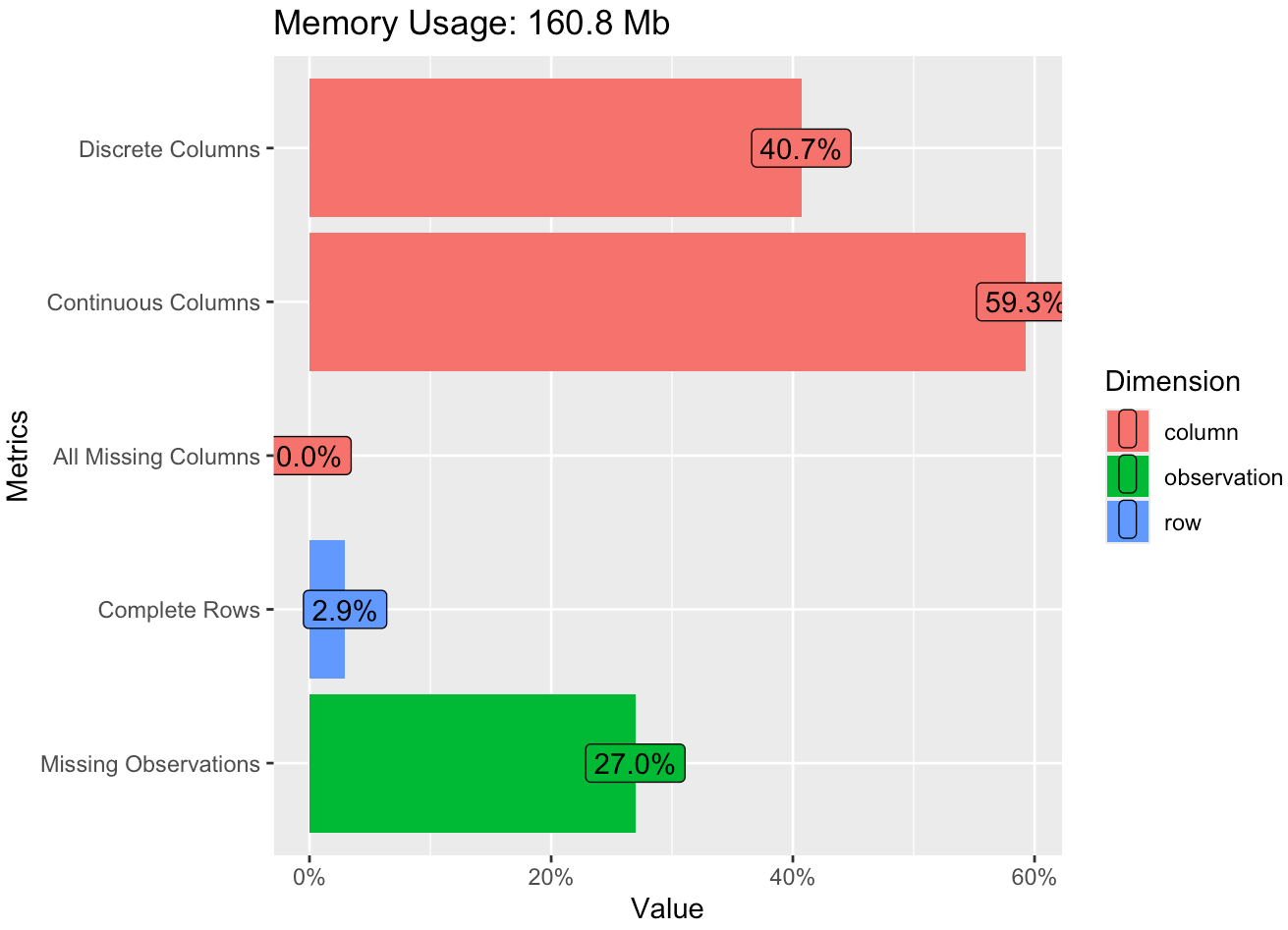
```
# View variables that are near zero variance
nearZeroVar(app_train,saveMetrics = TRUE)
```

```
# Filter out columns with NZvar
filter_train <- app_train |>
  select(-c(DAYS_EMPLOYED, FLAG_MOBIL, FLAG_CONT_MOBILE,
            REG_REGION_NOT_LIVE_REGION, LIVE_REGION_NOT_WORK_REGION,
            BASEMENTAREA_AVG, LANDAREA_AVG, NONLIVINGAREA_AVG,
            BASEMENTAREA_MODE, LANDAREA_MODE, NONLIVINGAREA_MODE,
            BASEMENTAREA_MEDI, LANDAREA_MEDI, NONLIVINGAREA_MEDI,
            HOUSETYPE_MODE, EMERGENCYSTATE_MODE, FLAG_DOCUMENT_2,
            FLAG_DOCUMENT_4, FLAG_DOCUMENT_5,FLAG_DOCUMENT_7, FLAG_DOCUMENT_9,
            FLAG_DOCUMENT_10, FLAG_DOCUMENT_11,FLAG_DOCUMENT_12,
            FLAG_DOCUMENT_13, FLAG_DOCUMENT_14,
            FLAG_DOCUMENT_15,FLAG_DOCUMENT_16, FLAG_DOCUMENT_17,
            FLAG_DOCUMENT_18, FLAG_DOCUMENT_19, FLAG_DOCUMENT_20,
            FLAG_DOCUMENT_21, AMT_REQ_CREDIT_BUREAU_DAY,
            AMT_REQ_CREDIT_BUREAU_HOUR, AMT_REQ_CREDIT_BUREAU_WEEK))

# Make sure test is updated too
app_test <- app_test |>
  select(-c(DAYS_EMPLOYED, FLAG_MOBIL, FLAG_CONT_MOBILE,
            REG_REGION_NOT_LIVE_REGION, LIVE_REGION_NOT_WORK_REGION,
            BASEMENTAREA_AVG, LANDAREA_AVG, NONLIVINGAREA_AVG,
            BASEMENTAREA_MODE, LANDAREA_MODE, NONLIVINGAREA_MODE,
            BASEMENTAREA_MEDI, LANDAREA_MEDI, NONLIVINGAREA_MEDI,
            HOUSETYPE_MODE, EMERGENCYSTATE_MODE, FLAG_DOCUMENT_2,
            FLAG_DOCUMENT_4, FLAG_DOCUMENT_5,FLAG_DOCUMENT_7, FLAG_DOCUMENT_9,
            FLAG_DOCUMENT_10, FLAG_DOCUMENT_11,FLAG_DOCUMENT_12,
            FLAG_DOCUMENT_13, FLAG_DOCUMENT_14,
            FLAG_DOCUMENT_15,FLAG_DOCUMENT_16, FLAG_DOCUMENT_17,
            FLAG_DOCUMENT_18, FLAG_DOCUMENT_19, FLAG_DOCUMENT_20,
            FLAG_DOCUMENT_21, AMT_REQ_CREDIT_BUREAU_DAY,
            AMT_REQ_CREDIT_BUREAU_HOUR, AMT_REQ_CREDIT_BUREAU_WEEK))

# Now we have 86 columns, which is slightly easier to approach
# head(filter_train)
```

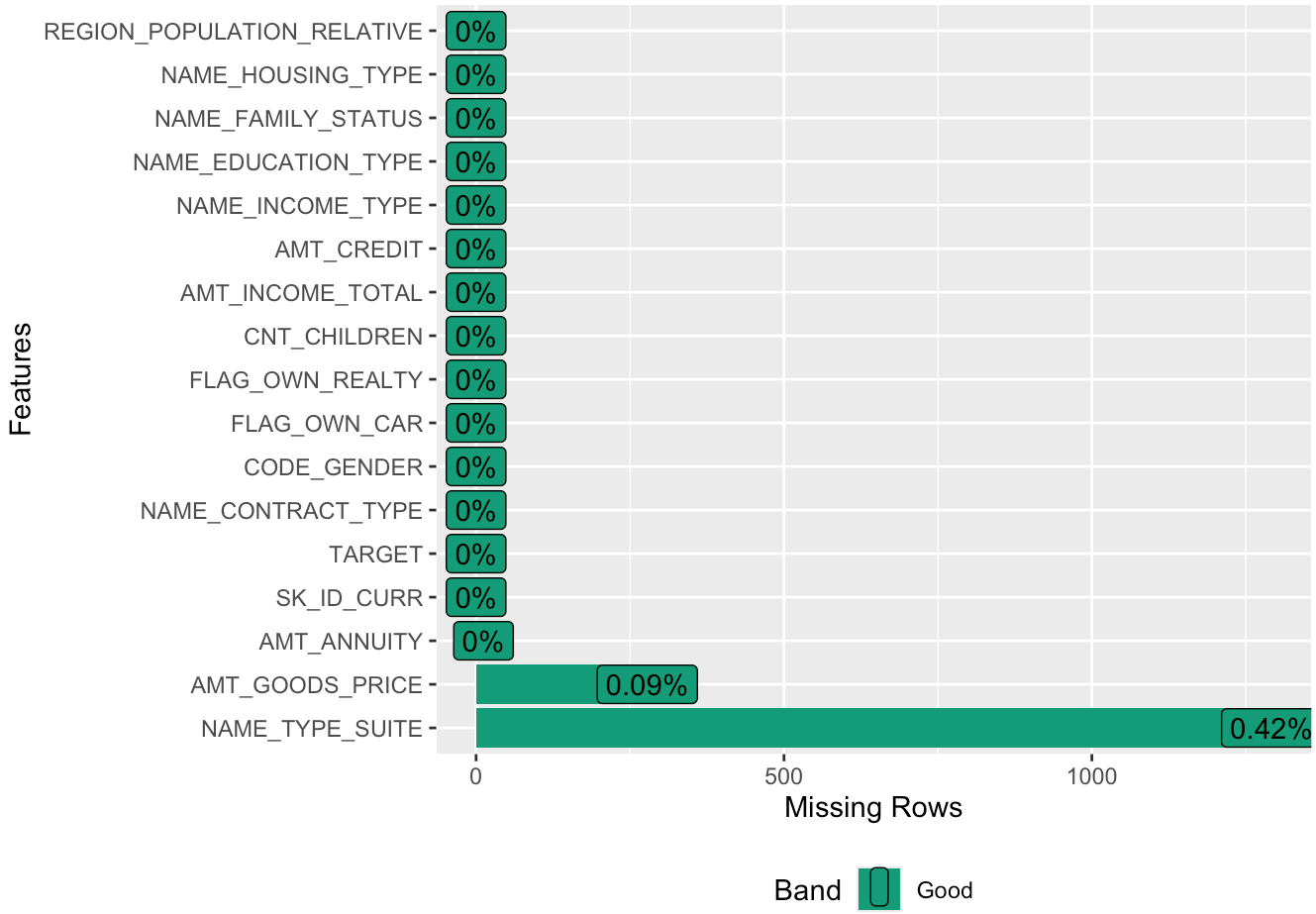
What is the Scope of missing data in application_{train|test}.csv?

There are lots of missing observations in these datasets. The train csv is missing 27% of observations while the test is missing 26.4%. Specifically for train, I wanted to see exactly how much was missing per column. Of course, this encouraged me to start thinking about how I wanted to approach this issue.

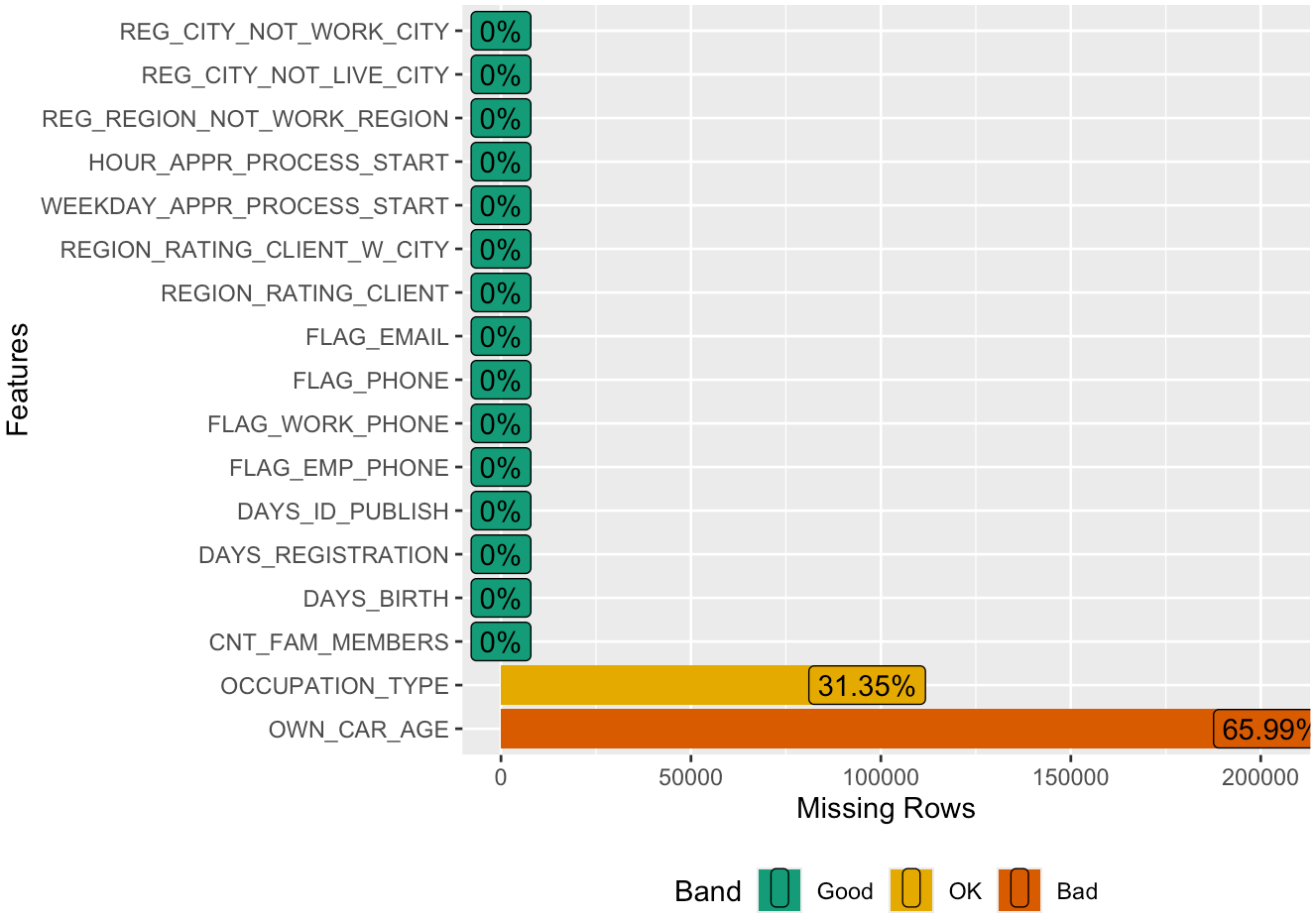
```
# Train set - 27% missing obs
# Visualization of metrics in dataset
filter_train |>
  plot_intro()
```



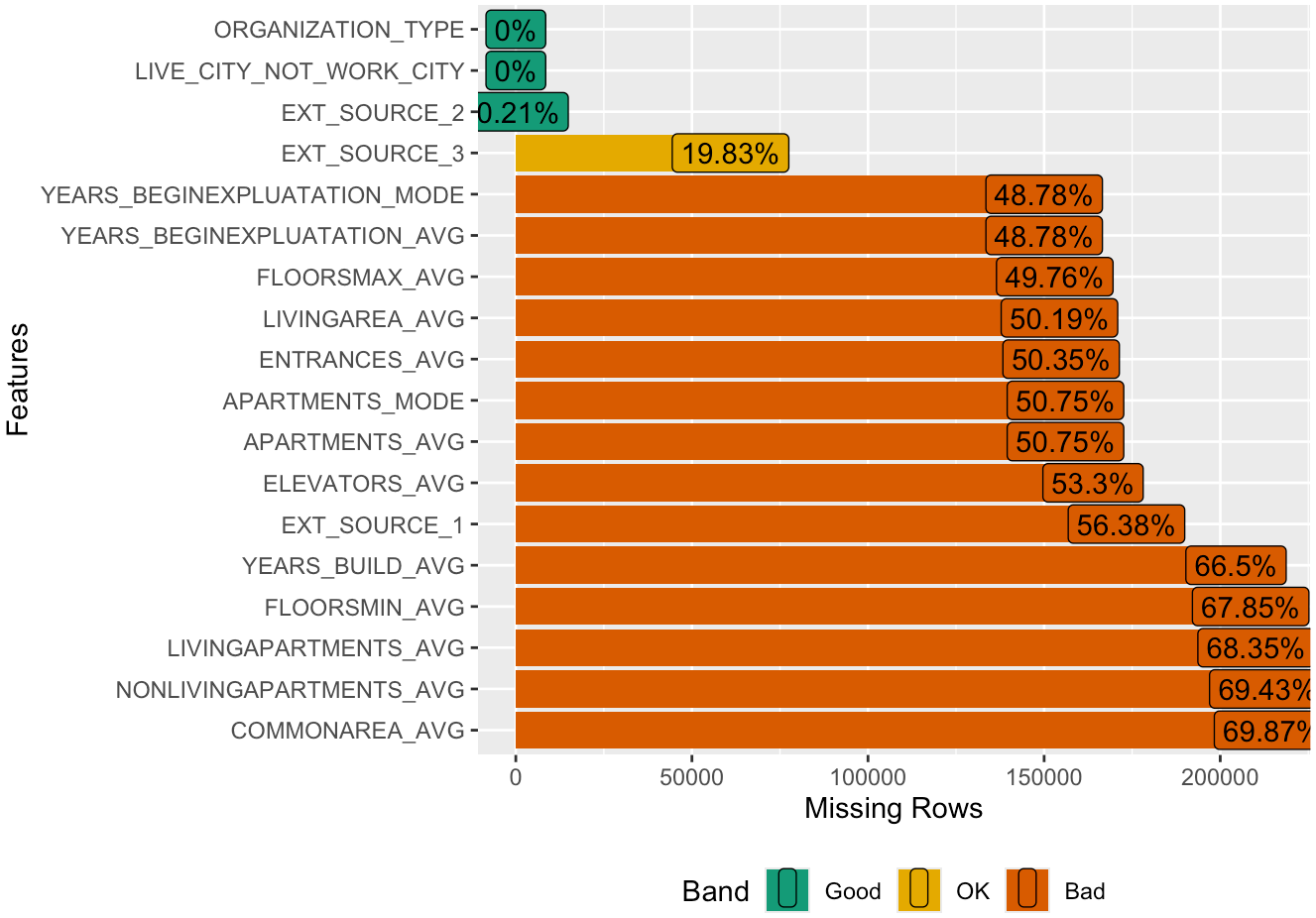

```
# Investigating missing observations per feature
missing1 <- filter_train |>
  select(1:17) |>
  plot_missing()
```



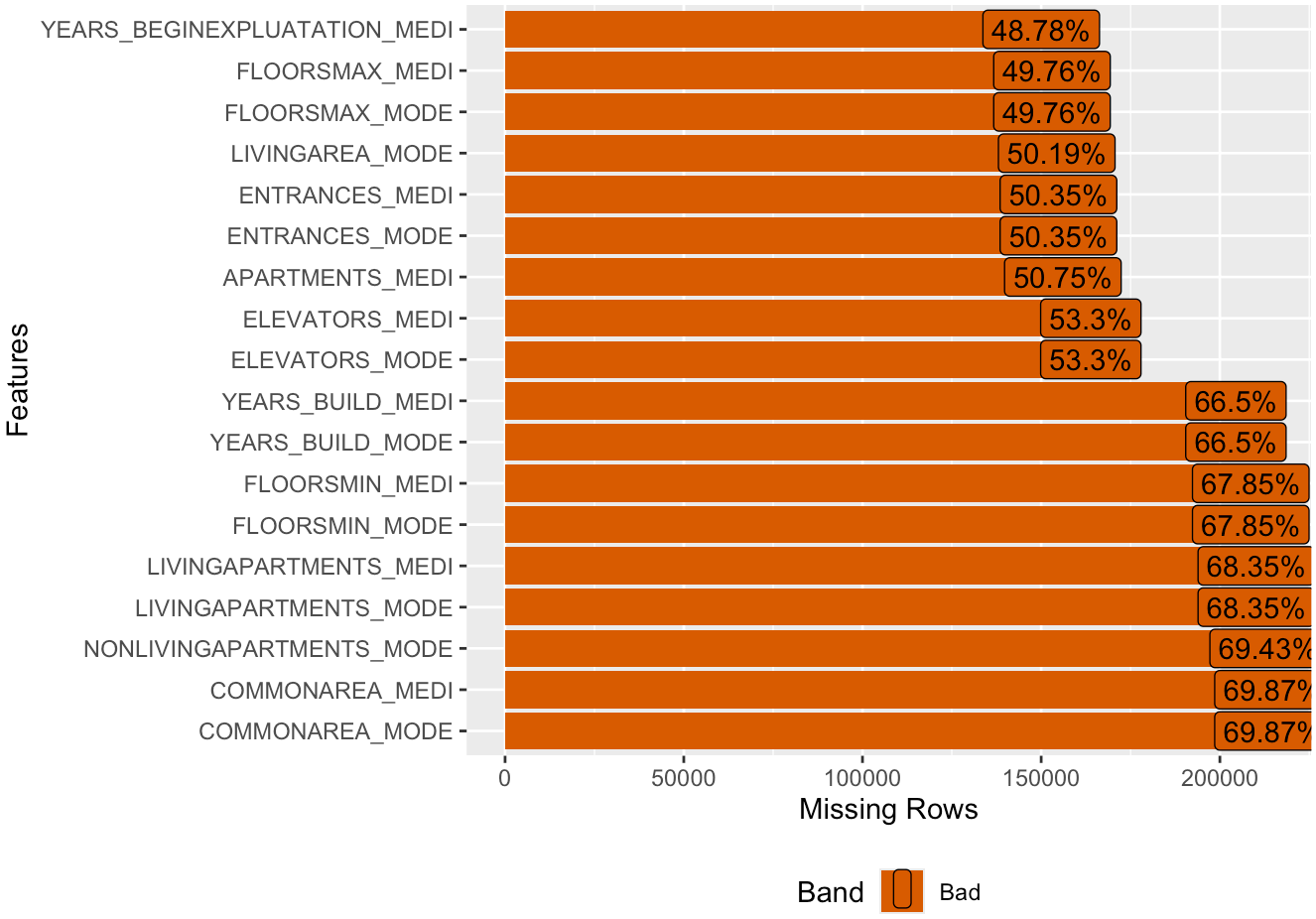

```
missing2 <- filter_train |>
  select(18:34) |>
  plot_missing()
```



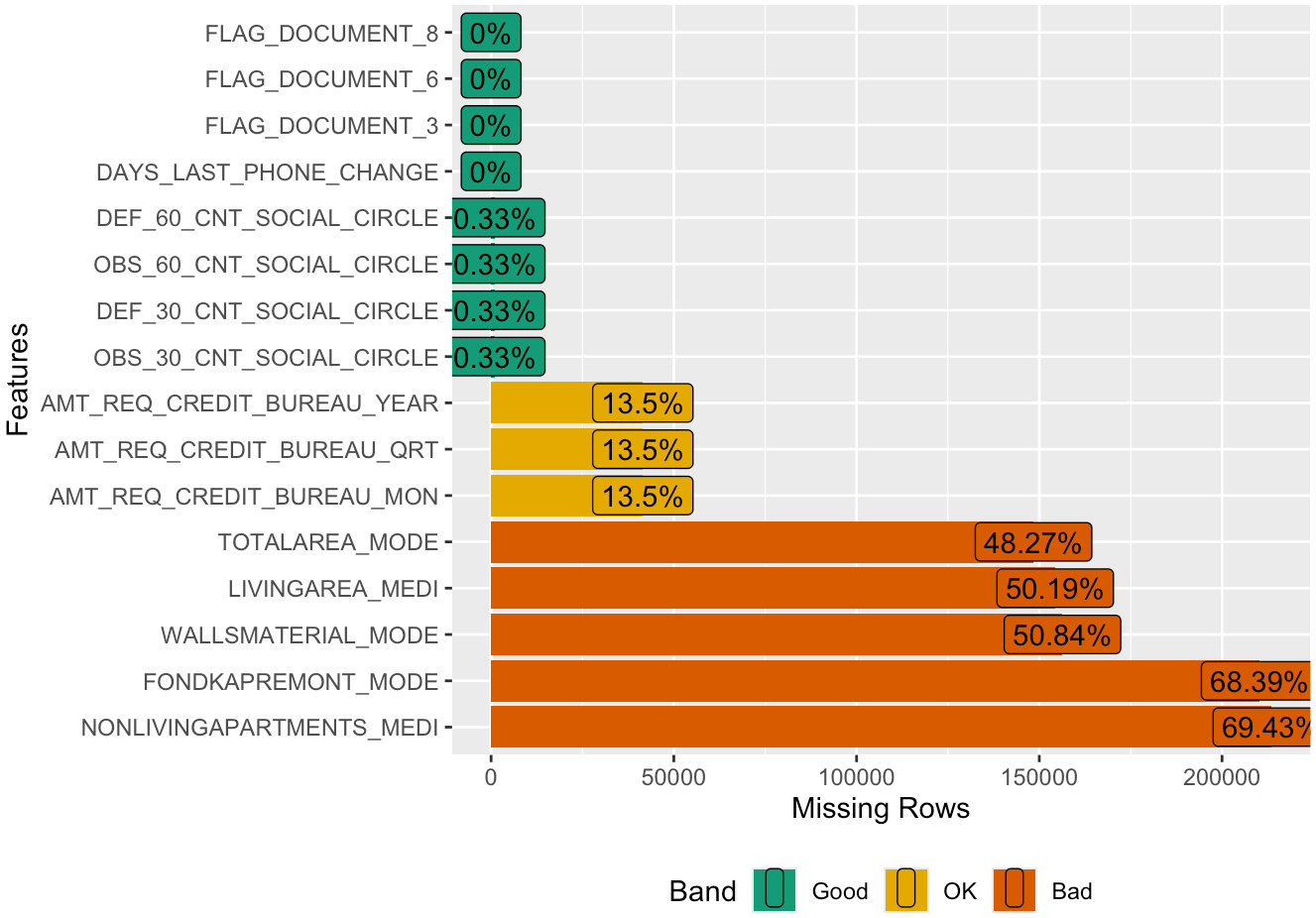

```
missing3 <- filter_train |>
  select(35:52) |>
  plot_missing()
```




```
missing4 <- filter_train |>
  select(53:70) |>
  plot_missing()
```

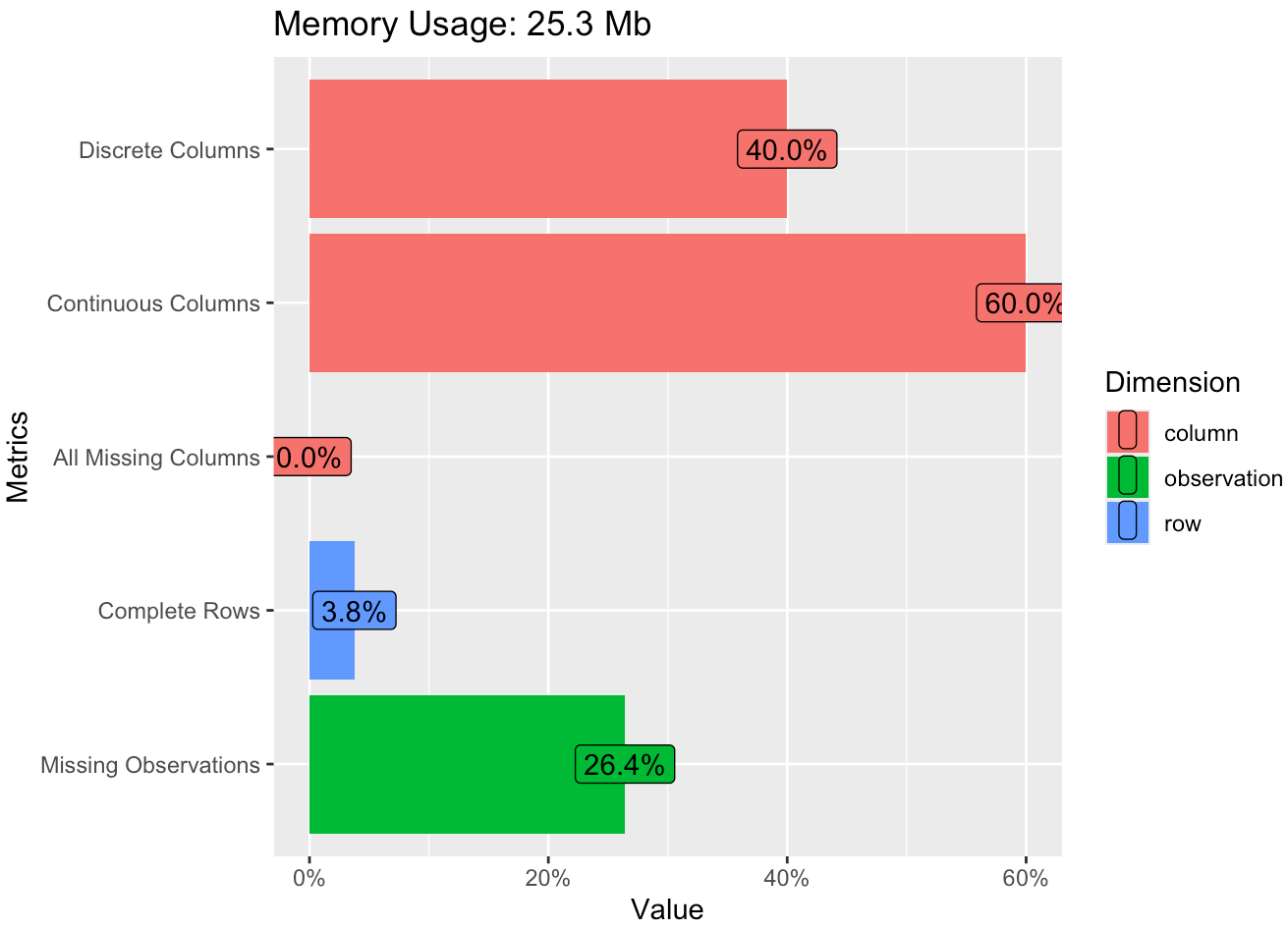



```
missing5 <- filter_train |>
  select(71:86) |>
  plot_missing()
```




```
# Full plot of how much data missing per column
# missing1 + missing2 + missing3 + missing4 + missing5
```

```
# Test set - 26.4% of missing observations
# Visualization of metrics in dataset
app_test |>
  plot_intro()
```



Are there String Errors in the Data?

To check for string errors, like misspelled values, I used the unique function. There is probably a more efficient way, but this is what I thought of.

```
# Check for string errors
unique(filter_train$NAME_TYPE_SUITE)
unique(filter_train$NAME_INCOME_TYPE)
unique(filter_train$NAME_EDUCATION_TYPE)
unique(filter_train$NAME_INCOME_TYPE)
unique(filter_train$NAME_FAMILY_STATUS)
unique(filter_train$NAME_HOUSING_TYPE)
unique(filter_train$OCCUPATION_TYPE)
unique(filter_train$WEEKDAY_APPR_PROCESS_START)
```

How Can I Reorganize the Organization Type Variable?

The ORGANIZATION_TYPE variable had 58 levels and I wanted to see if I could combine any. After looking at the levels, I created a “Public Saftey and Governance” level that contained military, police, security personnel. The other one I created was “Real Estate and Construction.” These changes helped me lower the amount of levels in the ORGANIZATION_TYPE variable to 52.

```
# This has 58 levels now
unique(filter_train$ORGANIZATION_TYPE)
levels(filter_train$ORGANIZATION_TYPE)
```

```
# Let's lower the levels for train and test
# Combining similar categories
new_train <- filter_train |>
  mutate(ORGANIZATION_TYPE = as.factor(case_when(
    ORGANIZATION_TYPE %in% c("Military", "Police", "Security", "Security Ministries",
                             "Government") ~ "Public Safety and Governance",
    ORGANIZATION_TYPE %in% c("Construction", "Housing") ~ "Real Estate and
    Construction",
    # Keep the rest of the values unchanged
    TRUE ~ ORGANIZATION_TYPE  # This will keep all other values as they are
  )))

app_test <- app_test |>
  mutate(ORGANIZATION_TYPE = as.factor(case_when(
    ORGANIZATION_TYPE %in% c("Military", "Police", "Security", "Security Ministries",
                             "Government") ~ "Public Safety and Governance",
    ORGANIZATION_TYPE %in% c("Construction", "Housing") ~ "Real Estate and
    Construction",
    # Keep the rest of the values unchanged
    TRUE ~ ORGANIZATION_TYPE  # This will keep all other values as they are
  )))
```

Do the Gender Levels Line Up in Train and Test?

```
# Levels of gender don't match up
levels(app_train$CODE_GENDER)
levels(app_test$CODE_GENDER)
```

```
# Add the third level for ease to test data
app_test$CODE_GENDER <- factor(app_test$CODE_GENDER, levels = c("F", "M", "XNA"))
```

Feature Engineering

Car Age Binning Variable

The OWN_CAR_AGE variable represents the age of a client's car, a numerical variable. This column has 200912 missing values. In this case, there is informative missingness. Applicants may have never owned a car, so they left the age of their nonexistent car, empty. For that reason, I have decided to transform this variable to a factor. There will be 4 categories - Missing, Low, Medium, and High.

It was difficult to decided what ages to consider low, medium, and high. As of 2024, S&P Global states that most vehicles are 12.6 years old on average. Google suggested that a "sweet spot" for cars is between 2-5 years old. I also read that cars in the 15-20 year age tend to be nearing the end of their services. For that reason, I am considering Low from 0-5, Medium from 6-15, and High from 16-100.

I also removed the old variable that way there isn't redundant information in our data frame.

```
# Figure out how many n's are missing
skim(new_train$OWN_CAR_AGE)

# Checking the unique values to decide how I want to separate the bins
unique(new_train$OWN_CAR_AGE)
```

```
# Will create 4 bins: Missing, low, medium, high and make sure column is a factor
new_train <- new_train |>
  mutate(OWN_CAR_AGE_BIN = as.factor(case_when(
    is.na(OWN_CAR_AGE) ~ "Missing",
    OWN_CAR_AGE >= 0 & OWN_CAR_AGE <= 5 ~ "Low",
    OWN_CAR_AGE >= 6 & OWN_CAR_AGE <= 15 ~ "Medium",
    OWN_CAR_AGE >= 16 & OWN_CAR_AGE <= 100 ~ "High"
  )))

# Do same for test
app_test <- app_test |>
  mutate(OWN_CAR_AGE_BIN = as.factor(case_when(
    is.na(OWN_CAR_AGE) ~ "Missing",
    OWN_CAR_AGE >= 0 & OWN_CAR_AGE <= 5 ~ "Low",
    OWN_CAR_AGE >= 6 & OWN_CAR_AGE <= 15 ~ "Medium",
    OWN_CAR_AGE >= 16 & OWN_CAR_AGE <= 100 ~ "High"
  )))

# Remove old column
new_train <- new_train |>
  select(-OWN_CAR_AGE)

app_test <- app_test |>
  select(-OWN_CAR_AGE)
```

Housing Info Binning Variable

There are many columns representing information that has to do with housing, about 25. I wanted to filter through them and decide whether applicants had all the columns filled, some, or few. This is another binning variable, few included 0-2, some had greater than 2, and all meant each column was filled with a value. My code was written to avoid considering NAs as a "value."

```
# Combine housing variables
housing_variables <-  c("LIVINGAREA_AVG", "NONLIVINGAPARTMENTS_AVG", "APARTMENTS_MODE",
         "YEARS_BEGINEXPLUATATION_MODE",
         "YEARS_BUILD_MODE", "COMMONAREA_MODE", "ENTRANCES_MODE",
         "FLOORSMAX_MODE", "FLOORSMIN_MODE", "LIVINGAPARTMENTS_MODE",
         "NONLIVINGAPARTMENTS_MODE", "APARTMENTS_MEDI",
         "YEARS_BEGINEXPLUATATION_MEDI", "YEARS_BUILD_MEDI", "COMMONAREA_MEDI",
         "ELEVATORS_MEDI", "ENTRANCES_MEDI", "FLOORSMAX_MEDI", "FLOORSMIN_MEDI",
         "LIVINGAPARTMENTS_MEDI", "LIVINGAREA_MEDI",
         "NONLIVINGAPARTMENTS_MEDI", "FONDKAPREMONT_MODE",
         "TOTALAREA_MODE", "WALLSMATERIAL_MODE")

# Feature engineering for the new variable - should prevent NAs from being considered as
values?
new_train <- new_train %>%
  mutate(HOUSING_INFO = as.factor(case_when(
    # Few values entered
    rowSums(!is.na(select(., all_of(housing_variables)))) <= 2 ~ "Few",
    # All values entered
    rowSums(!is.na(select(., all_of(housing_variables)))) == length(housing_variables)
    ~ "All",
    # Two or more values entered
    rowSums(!is.na(select(., all_of(housing_variables)))) > 2 ~ "Some"
)))

# See count at the different levels
new_train |>
  rename("Housing Info" = HOUSING_INFO) |>
  count(`Housing Info`, name = "Count") |>
  kbl() |>
  kable_styling()
```

Housing Info	Count
All	82425
Few	148859
Some	76227

```
# Do same for test data
app_test <- app_test %>%
  mutate(HOUSING_INFO = as.factor(case_when(
    # Few values entered
    rowSums(!is.na(select(., all_of(housing_variables)))) <= 2 ~ "Few",
    # All values entered
    rowSums(!is.na(select(., all_of(housing_variables)))) == length(housing_variables) ~
"All",
    # Two or more values entered
    rowSums(!is.na(select(., all_of(housing_variables)))) > 2 ~ "Some"
)))
```

Amount of Children Applicant Has

I decided I wanted to bin CNT_CHILDREN for both train and test set. This was a good solution for me because I ran into issues with my test csv having new numeric variables that weren't recognized because of how I had trained my data. The categories are 0, 1-3, and 4+ children.

```
# Make sure cnt_children is numeric in train and test
new_train <- new_train |>
  mutate(CNT_CHILDREN = as.numeric(CNT_CHILDREN))

app_test <- app_test |>
  mutate(CNT_CHILDREN = as.numeric(CNT_CHILDREN))

# Make change to train
new_train <- new_train |>
  mutate(CNT_CHILDREN_BIN = case_when(
            CNT_CHILDREN == 0 ~ "Zero",
            CNT_CHILDREN > 0 & CNT_CHILDREN <= 3 ~ "One-Three",
            CNT_CHILDREN > 3 ~ "Greater than 4")) |>
  mutate(CNT_CHILDREN_BIN = as.factor(CNT_CHILDREN_BIN))  # Convert the new bins to a fa
ctor

# Make change to test
app_test <- app_test %>%
  mutate(CNT_CHILDREN_BIN = case_when(
            CNT_CHILDREN == 0 ~ "Zero",
            CNT_CHILDREN > 0 & CNT_CHILDREN <= 3 ~ "One-Three",
            CNT_CHILDREN > 3 ~ "Greater than 4")) %>%
  mutate(CNT_CHILDREN_BIN = as.factor(CNT_CHILDREN_BIN))  # Convert the new bins to a fa
ctor

# Remove old CNT from train and test to avoid redundancy
new_train <- new_train %>%
  select(-CNT_CHILDREN)

app_test <- app_test %>%
  select(-CNT_CHILDREN)
```

Data Imputation

Mean/Mode Strategy

I was having a really hard time getting the data to impute. I start off with missForest, but it took hours to run given the method. Instead, for numeric variables, I decided to impute using the mean, whereas for categorical/factor variables with the mode. This was a good solution because it was much faster. Since I imputed the train data, the best practice is to do the same to the test. I had to make sure that all the levels in the data line up too.

```
# Impute data
my_recipe <- recipe(TARGET ~ ., data = new_train) %>%
  step_impute_mean(all_numeric(), -all_outcomes()) %>%  # Impute Numeric w/ mean
  step_impute_mode(all_nominal(), -all_outcomes())   # Impute categorical w/ mode

# Prep and bake
prepped_recipe <- prep(my_recipe)
imputed_data <- bake(prepped_recipe, new_data = new_train)

# Yay! No more NAs in the data!
#any(is.na(imputed_data))

# This method actually leaves out the target, so we need to add the target back to our d
ata frame.
imputed_data <- imputed_data |>
   left_join(select(app_train, SK_ID_CURR, TARGET), by="SK_ID_CURR")

finalized_imputed_data <- imputed_data |>
  select(-TARGET.x) |>
  rename(TARGET = TARGET.y)
```

```
# Make sure everything is lined up between train and test
for (col in names(app_train)) {
  if (is.factor(app_train[[col]])) {
    # Check if the column exists in the test set
    if (col %in% names(app_test)) {
      # Align levels between train and test for factors
      app_test[[col]] <- factor(app_test[[col]], levels = levels(app_train[[col]]))
    }
  }
}

for (col in names(app_train)) {
  if (is.factor(app_train[[col]])) {
    if (col %in% names(app_test)) {
      # Align levels and preserve missing levels
      app_test[[col]] <- factor(app_test[[col]], levels = levels(app_train[[col]]))

      # Fill in any missing levels with NA
      app_test[[col]][is.na(app_test[[col]])] <- NA
    }
  }
}

# Impute data
test_my_recipe <- recipe(TARGET ~ ., data = new_train) %>%
  step_impute_mean(all_numeric(), -all_outcomes()) %>%  # Impute Numeric w/ mean
  step_impute_mode(all_nominal(), -all_outcomes())   # Impute categorical w/ mode

# Prep and bake
test_prepped_recipe <- prep(test_my_recipe)
test_imputed_data <- bake(test_prepped_recipe, new_data = app_test)

# Yay! No more NAs in the data!
#any(is.na(test_imputed_data))
```

Are There More Near Zero Variance Predictors?

Now that the data is imputed, I checked if there were any more near zero variance predictors.

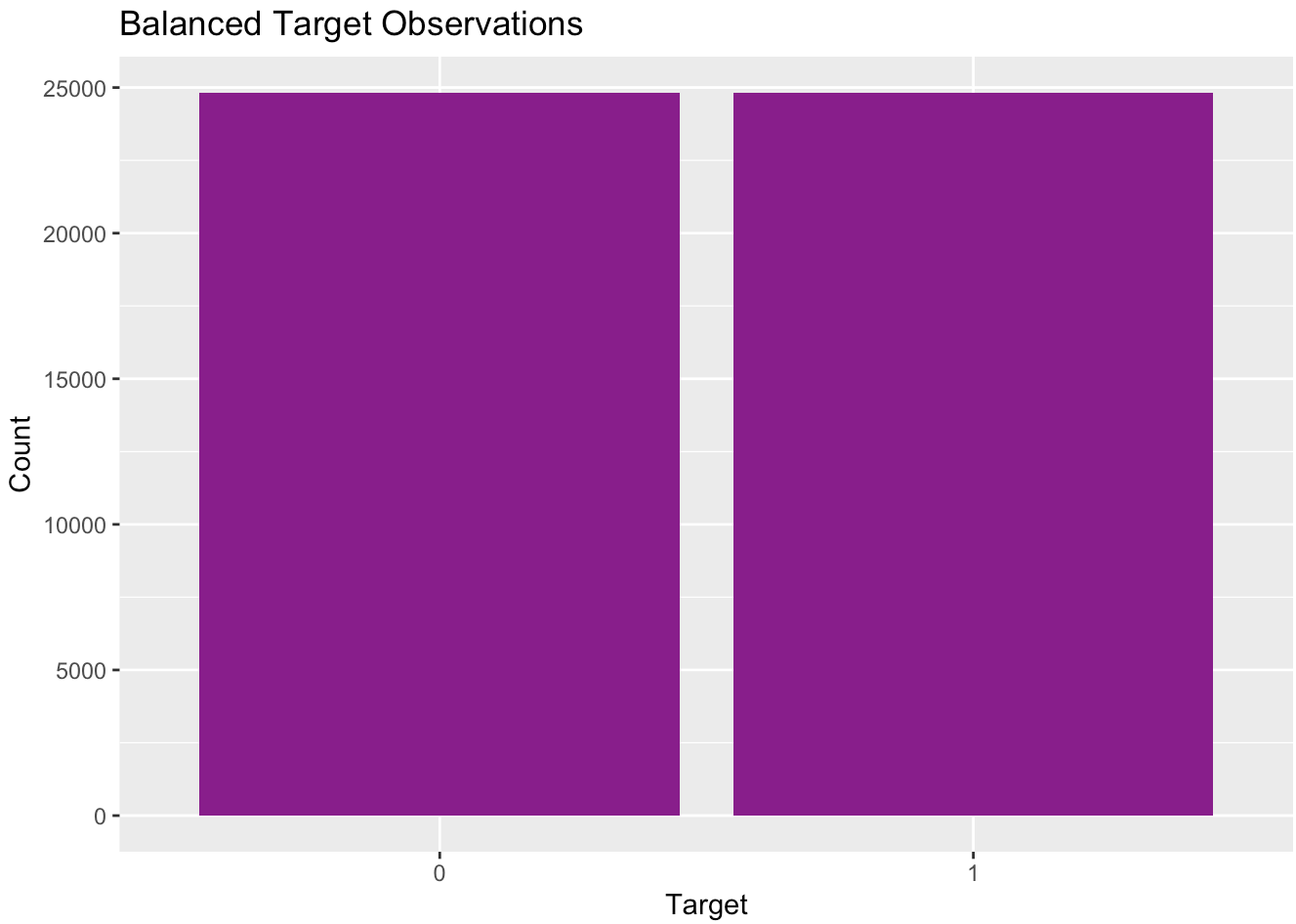
```
# Now that data is imputed, filter them out again
nearZeroVar(finalized_imputed_data,saveMetrics = TRUE)
```

```
filtered_imputed <- finalized_imputed_data |>
  select(-c("EXT_SOURCE_3", "APARTMENTS_AVG", "YEARS_BEGINEXPLUATATION_AVG",
            "YEARS_BUILD_AVG", "COMMONAREA_AVG","LIVINGAPARTMENTS_AVG",
            "LIVINGAREA_AVG", "APARTMENTS_MODE", "YEARS_BEGINEXPLUATATION_MODE",
            "YEARS_BUILD_MODE", "COMMONAREA_MODE", "LIVINGAPARTMENTS_MODE",
            "LIVINGAREA_MODE","APARTMENTS_MEDI", "YEARS_BEGINEXPLUATATION_MEDI",
            "YEARS_BUILD_MEDI", "COMMONAREA_MEDI", "LIVINGAPARTMENTS_MEDI",
            "LIVINGAREA_MEDI", "FONDKAPREMONT_MODE", "TOTALAREA_MODE"))

# DELETE THIS IF MODEL DOESN'T END UP WORKING
app_test <- app_test |>
  select(-c("EXT_SOURCE_3", "APARTMENTS_AVG", "YEARS_BEGINEXPLUATATION_AVG",
            "YEARS_BUILD_AVG", "COMMONAREA_AVG","LIVINGAPARTMENTS_AVG",
            "LIVINGAREA_AVG", "APARTMENTS_MODE", "YEARS_BEGINEXPLUATATION_MODE",
            "YEARS_BUILD_MODE", "COMMONAREA_MODE", "LIVINGAPARTMENTS_MODE",
            "LIVINGAREA_MODE","APARTMENTS_MEDI", "YEARS_BEGINEXPLUATATION_MEDI",
            "YEARS_BUILD_MEDI", "COMMONAREA_MEDI", "LIVINGAPARTMENTS_MEDI",
            "LIVINGAREA_MEDI", "FONDKAPREMONT_MODE", "TOTALAREA_MODE"))
```

How to Approach Imbalanced Data?

As we know from the earlier analysis of the target variable, the data is highly imbalanced. This needs to be taken care of in order to use logistic regression. My approach was to undersample using the ROSE library.

```
# Address class imbalanced data by under sampling
balanced_data <- ovun.sample(TARGET ~ ., data = filtered_imputed, method = "under", N =
2 * 24825)$data

# Balanced amount of observations which may be helpful for regression
ggplot(data = balanced_data, mapping = aes(x=TARGET)) +
  geom_bar(fill = "magenta4") + labs(title = "Balanced Target Observations") +
  xlab("Target") + ylab("Count")
```



Modeling - Which Predictors are Significant?

Simple Models

Does Car Ownership Predict Default?

The answer is yes! The p-value was significant and received a result of 0.50989 in Kaggle. Approach was a simple binary logistic regression.

```
# Set up logistic regression
car.glm <- glm(TARGET ~ FLAG_OWN_CAR ,data=balanced_data, family = "binomial")
summary(car.glm)
```

```
##
## Call:
## glm(formula = TARGET ~ FLAG_OWN_CAR, family = "binomial", data = balanced_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.20122  -1.20122   0.01274   1.15379   1.22731
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.05585    0.01092   5.114 3.15e-07 ***
## FLAG_OWN_CARY -0.17246    0.01920  -8.981  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 68830  on 49649  degrees of freedom
## Residual deviance: 68749  on 49648  degrees of freedom
## AIC: 68753
##
## Number of Fisher Scoring iterations: 3
```

```
## Make predictions
predictions <- data.frame(.pred = predict(car.glm, newdata=app_test, type = "response"))

# Prepare data for kaggle submission
kaggle_submission <- predictions %>%
  bind_cols(., app_test) %>%
  select(SK_ID_CURR, .pred) %>%
  rename(TARGET=.pred)

# Write out file
# vroom::vroom_write(x=kaggle_submission, file="./CarOwnershipLogPreds.csv", delim=",")
```

Does Car Age Predict Default?

This is a column I featured engineered, the p-value was significant and received a result of 0.54051 in Kaggle. Another simple binary logistic regression.

```
# Set up logistic regression
car_age_bin.glm <- glm(TARGET ~ OWN_CAR_AGE_BIN , data = balanced_data, family = "binomi
al")
summary(car_age_bin.glm)
```

```
##
## Call:
## glm(formula = TARGET ~ OWN_CAR_AGE_BIN, family = "binomial",
##     data = balanced_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.24102  -1.20122   0.04949   1.15379   1.34774
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)             0.14833    0.03055   4.855 1.20e-06 ***
## OWN_CAR_AGE_BINLow     -0.54028    0.04423 -12.215  < 2e-16 ***
## OWN_CAR_AGE_BINMedium  -0.27028    0.03813  -7.089 1.35e-12 ***
## OWN_CAR_AGE_BINMissing -0.09248    0.03245  -2.850  0.00437 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 68830  on 49649  degrees of freedom
## Residual deviance: 68598  on 49646  degrees of freedom
## AIC: 68606
##
## Number of Fisher Scoring iterations: 3
```

```
## Make predictions
predictions <- data.frame(.pred = predict(car_age_bin.glm, newdata=app_test, type = "res
ponse"))

# Prepare data for kaggle submission
kaggle_submission <- predictions %>%
  bind_cols(., app_test) %>%
  select(SK_ID_CURR, .pred) %>%
  rename(TARGET=.pred)

# Write out file
# vroom::vroom_write(x=kaggle_submission, file="./car_age_bin_LogPreds.csv", delim=",")
```

Does Count Children Bin Predict Default?

Yes, looks like bin of 1-3 children was significant.

```
# Set up logistic regression
cnt_children.glm <- glm(TARGET ~  CNT_CHILDREN_BIN, data = balanced_data, family = "bino
mial")

# Looks like the factor level of 1-3 is significant
summary(cnt_children.glm)
```

```
##
## Call:
## glm(formula = TARGET ~ CNT_CHILDREN_BIN, family = "binomial",
##     data = balanced_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.27890  -1.17585  -0.04832   1.17897   1.17897
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 0.23546    0.07267   3.240  0.00119 **
## CNT_CHILDREN_BINOne-Three -0.23914    0.07323  -3.266  0.00109 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 68830  on 49649  degrees of freedom
## Residual deviance: 68819  on 49648  degrees of freedom
## AIC: 68823
##
## Number of Fisher Scoring iterations: 3
```

Complex Model

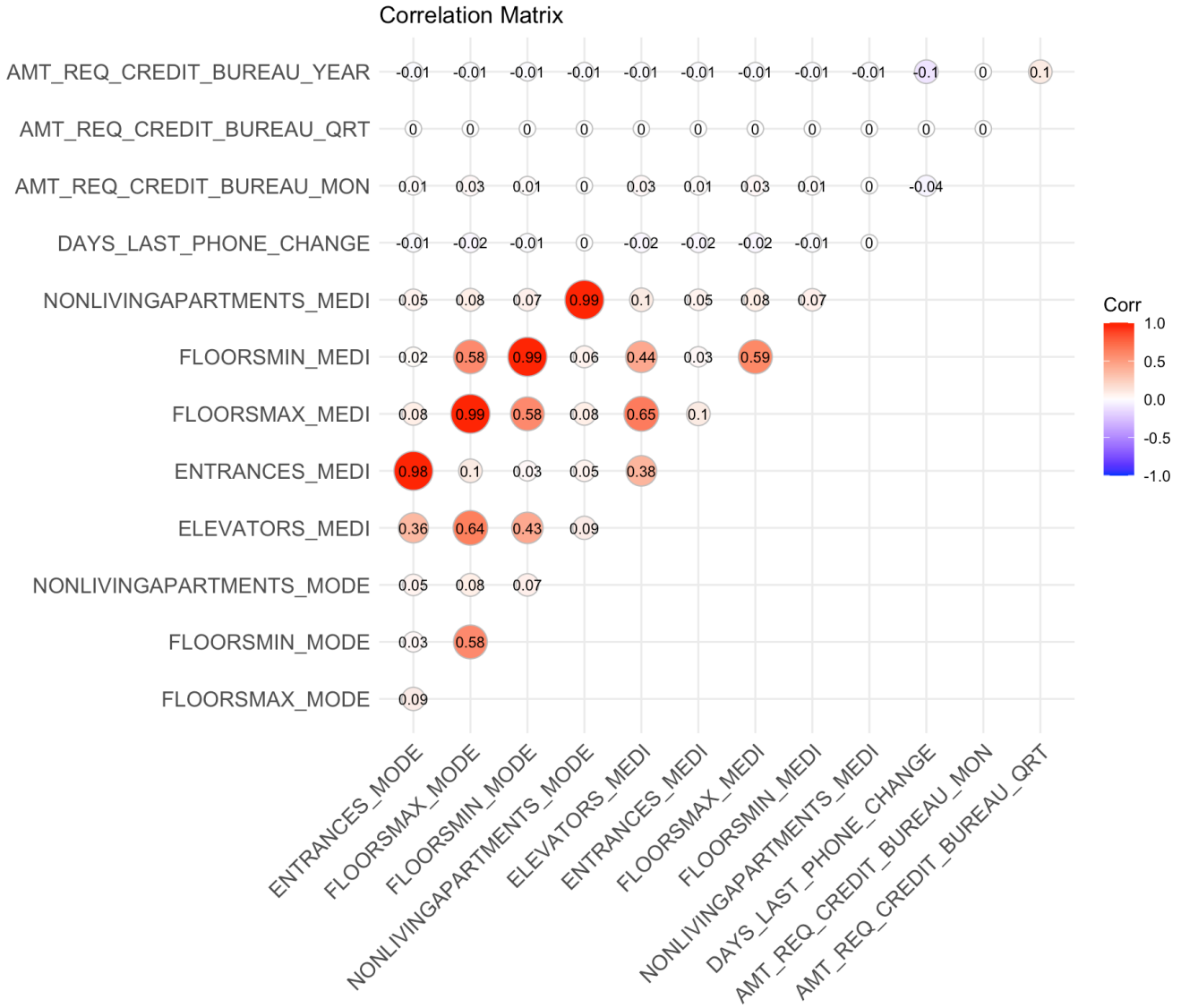
Since I did some simple models, I wanted to create a more complex one that identified which predictors to keep. This is a hefty task, so it required some work arounds.

To start off, I subsetted the data 3 times. Each time, I put the data into a logistic regression model. I looked at whether or not the predictors were significant. From there, I would fine tune which variables to keep or drop. With subset 3, I was running into some issues revolved around highly correlated variables. For that reason, I separated all the numeric variables to check which ones in the group were highly correlated. Several of the home variables related to mode and median were correlated, so I ended up removing the modes.

After that work, I finally started experimenting with putting all the significant predictors in a model.

```
# First subset
subset_data1 <- subset_data <- balanced_data[, c("TARGET", names(balanced_data)[1:21])]

all.glm <- glm(TARGET ~ ., data=subset_data1, family = "binomial")
#summary(all.glm)

sig_subset1.glm <- glm(TARGET ~ NAME_CONTRACT_TYPE + CODE_GENDER + AMT_CREDIT + AMT_ANNU
ITY + AMT_GOODS_PRICE + NAME_TYPE_SUITE + NAME_EDUCATION_TYPE + NAME_FAMILY_STATUS + REG
ION_POPULATION_RELATIVE + DAYS_BIRTH + DAYS_REGISTRATION + DAYS_ID_PUBLISH + FLAG_EMP_PH
ONE + FLAG_WORK_PHONE, data=balanced_data, family = "binomial")
#summary(sig_subset1.glm)

# Second subset
subset_data2 <- subset_data <- balanced_data[, c("TARGET", names(balanced_data)[21:41])]

all.glm <- glm(TARGET ~ ., data=subset_data2, family = "binomial")
#summary(all.glm)

sig_subset2.glm <- glm(TARGET ~ FLAG_PHONE + OCCUPATION_TYPE + REGION_RATING_CLIENT + WE
EKDAY_APPR_PROCESS_START + HOUR_APPR_PROCESS_START + REG_CITY_NOT_LIVE_CITY + EXT_SOURCE
_1 + EXT_SOURCE_2 + ENTRANCES_AVG + FLOORSMAX_AVG, data=balanced_data, family = "binomia
l")
#summary(sig_subset2.glm)

# Third subset
subset_data3 <- subset_data <- balanced_data[, c("TARGET", names(balanced_data)[42:66])]
subset_data3 <- subset_data3 |>
  select(-c(OBS_60_CNT_SOCIAL_CIRCLE, OBS_30_CNT_SOCIAL_CIRCLE, -TARGET.1))


# Check which variables from this group are highly correlated
numeric_subset_data3 <- subset_data3 |>
  select(ENTRANCES_MODE, FLOORSMAX_MODE, FLOORSMIN_MODE,
         NONLIVINGAPARTMENTS_MODE, ELEVATORS_MEDI, ENTRANCES_MEDI,
         FLOORSMAX_MEDI, FLOORSMIN_MEDI, NONLIVINGAPARTMENTS_MEDI,
         DAYS_LAST_PHONE_CHANGE, AMT_REQ_CREDIT_BUREAU_MON,
         AMT_REQ_CREDIT_BUREAU_QRT, AMT_REQ_CREDIT_BUREAU_YEAR)
```

```
# Calculate the correlation matrix (exclude the TARGET column)
cor_matrix <- cor(numeric_subset_data3, use = "complete.obs")  # Exclude TARGET

# Plot correlation matrix
ggcorrplot(cor_matrix, method = "circle",
           type = "upper",
           lab = TRUE,                    # Show correlation coefficients
           lab_size = 3,                  # Size of the text labels
           colors = c("blue", "white", "red"),
           title = "Correlation Matrix",
           ggtheme = theme_minimal())      # Use a minimal theme
```




```
# Correlation matrix is showing very high between ENTRANCES_MEDI + ENTRANCES_MODE, FLOOR
SMAX_MODE + FLOORSMAX_MEDI, FLOORSMIN_MODE + FLOORSMIN_MEDI, NONLIVINGAPARTMENTS_MEDI +
NONLIVING APARTMENTS MODE. For the sake of this, lets drop all the modes from subset3
```

```
subset_data3 <- subset_data3 |>
  select(-c(ENTRANCES_MODE, FLOORSMAX_MODE, FLOORSMIN_MODE, NONLIVINGAPARTMENTS_MODE))

# Do all the predictors now from the subsets
all.glm <- glm(TARGET ~ NAME_CONTRACT_TYPE + CODE_GENDER + AMT_CREDIT + AMT_ANNUITY +
                 AMT_GOODS_PRICE + NAME_TYPE_SUITE + NAME_EDUCATION_TYPE +
                 NAME_FAMILY_STATUS + REGION_POPULATION_RELATIVE + DAYS_BIRTH +
                 DAYS_REGISTRATION + DAYS_ID_PUBLISH + FLAG_EMP_PHONE +
                 FLAG_WORK_PHONE + FLAG_PHONE + OCCUPATION_TYPE +
                 REGION_RATING_CLIENT + WEEKDAY_APPR_PROCESS_START +
                 HOUR_APPR_PROCESS_START + REG_CITY_NOT_LIVE_CITY +
                 EXT_SOURCE_1 + EXT_SOURCE_2 + ENTRANCES_AVG +
                 FLOORSMAX_AVG + WALLSMATERIAL_MODE + DAYS_LAST_PHONE_CHANGE +
                 OWN_CAR_AGE_BIN + HOUSING_INFO +
                 AMT_REQ_CREDIT_BUREAU_MON + AMT_REQ_CREDIT_BUREAU_QRT +
                 AMT_REQ_CREDIT_BUREAU_YEAR,
               data = balanced_data,
               family = "binomial")
#summary(all.glm)

# In the final one NAME_TYPE_SUITE not signifcant, HOUR_APPR_PROCESS_START,
# WALLSMATERIAL, AMT_REQ_CREDIT_BUREAU_MON, AMT_REQ_CREDIT_BUREAU_QRT
```

Finalized Complex Logistic Regression Model

This model performed fairly well in Kaggle, I got a score of 0.68975. This model provided my best result so far.

```
finalized_glm_cols <- balanced_data |>
  select(TARGET, NAME_CONTRACT_TYPE, CODE_GENDER, AMT_CREDIT, AMT_ANNUITY,
         AMT_GOODS_PRICE, NAME_EDUCATION_TYPE, NAME_FAMILY_STATUS,
         REGION_POPULATION_RELATIVE, DAYS_BIRTH, DAYS_REGISTRATION,
         DAYS_ID_PUBLISH, FLAG_EMP_PHONE, FLAG_WORK_PHONE, FLAG_PHONE,
         OCCUPATION_TYPE, REGION_RATING_CLIENT, WEEKDAY_APPR_PROCESS_START,
         REG_CITY_NOT_LIVE_CITY,EXT_SOURCE_1, EXT_SOURCE_2, ENTRANCES_AVG,
         FLOORSMAX_AVG, DAYS_LAST_PHONE_CHANGE, OWN_CAR_AGE_BIN, HOUSING_INFO,
         AMT_REQ_CREDIT_BUREAU_YEAR)

finalized.glm <- glm(TARGET ~ .,
              data = finalized_glm_cols,
              family = "binomial")
## Make predictions
predictions <- data.frame(.pred = predict(finalized.glm, newdata=test_imputed_data, type
= "response"))

# Prepare data for kaggle submission
kaggle_submission <- predictions %>%
  bind_cols(., app_test) %>%
  select(SK_ID_CURR, .pred) %>%
  rename(TARGET=.pred)

# Write out file
#vroom::vroom_write(x=kaggle_submission, file="./glmPreds.csv", delim=",")
```

Results

It is clear that there are many data problems! To start off, there were categorical variables that needed to be factored. The target variable was highly imbalanced. Moreover, there was a lot of missing data in both the train and test csv. I decided that I needed to clean up the data a bit, feature engineer some columns, and then impute the missing data. While cleaning up the data, I noticed there were many near zero variance predictors which were removed. Also, I recategorized some variables - CODE_GENDER and ORGANIZATION_TYPE. I featured engineered some variables to bins. There were also some levels in train and test that didn't line up, so I fixed that too. As I was modeling the data using logistic regression, I ran into issues with highly correlated variables. I was able to create a plot to figure out which ones to remove.

Each step of my EDA, the many issues I ran into, completely changed how I thought about my analytics approach. For instance, I didn't realize that an imbalanced target variable would give me a such a hard time. This led me to an under sampling approach to help my data be more "balanced" for my models. As I created the logistic regression models, I realized it may be nice to use a Black box modeling method. One that comes to mind is Random Forests since it picks features all on its own. If that went well, maybe a more advanced model like XGBoost would do well too.

The predictors I included in my complex model all had some strong relationships going on - accuracy of 68%. Considering all the models I created are preliminary, they did a decent job, there is still a long way to go though! The featured engineered columns related to car age and car ownership did a good job at predicting default - around 50%. Moreover, the ext_source variables seemed to do a good job at predicting default, when reading the

data description it turns out they are credit scores. However, in my business statement, this was something I wanted to avoid using on its own. If I continue to include these variables, there will certainly be several other demographic variables included in the model.

Future Plans

I have done a lot of work on this dataset so far, but I know there is a lot more to go. First of all, I would like to figure out how to handle outliers, I haven't touched that too much yet. When actually modeling the data, I would like to do hypothesis tests and look at confidence intervals. It would be nice to have some statistical answers as to what is going on in the data. Next, I would like to join more of the csv files provided (transactional, bureau, previous application), but this would be a big step to see if there are more predictive variables that would increase model accuracy.

In my approach with the EDA, I didn't create validation sets from the train data because there was already a separate test csv. For the actual modeling, I think there would be benefit with creating validation sets since so many models will be made.

Lastly, I'd like to see if missing data from ext_source variables are correlated with repayment? A variable could be created to identify whether the credit score was missing. I look forward to joining with my group and seeing what they were able to find! I know working with a team will be a great experience. Thanks.