ML Assignment 1: HDMA data

Imran Aziz, Conrard G. T. Feugmo, Andre Schardong, Milton Segura, Sarpreet Gill (York University)

October 06, 2019

Contents

1 Introduction: Supervised Machine Learning Problem.

For this supervised machine learning problem, we have decided to use a dataset provided by Home Disclosure Mortgage Act (HDMA) for mortgage/loan applicants in Washington State in 2016. The dataset is available on the kaggle website. The US regulation for HDMA indicates that applications for mortgages need to be public with the exception Personally Identifable Information (PII) data such as names, phone numbers, etc.

In the business problem below, we have created a hypothetical situation in which the state government of Washington State is attempting to address a social issue in the origination of mortgages/loans. We will speak about the business and government interchangably within this paper. The government is the "business".

2 Business Problem Definition

The state government of Washington State (USA) has identified the following problems within the housing market: people are not getting approved for home mortgage loans (new homes and re-financing) by financial institutions. With people not getting approved for their loans, an increased reliance on rental properties is driving rent costs to rise. Increased rent costs is reducing the standard of living within the state. The government would like to start a pilot program to assist those who would most likely not receive approval for a mortgage for their primary residence. The pilot program would provide financial advice and assistance to increase the likelihood of securing a mortgage. The government would like assistance to identify individuals who most likely would not be able to secure a loan for their primary residence. And they approached our Machine Learning group for assistance.

3 Re-framing the Business Problem as an analytic problem

The business problem indicates applicants need to be identified as a "candidate" or "not a candidate" for the pilot program (which provides assistance to those who unable to acquire a mortgage). The dataset for HDMA contains loan application details. Each application has a result or outcome within the "action_taken_name" column of the dataset. The column "action_taken_name" can have the following values: * Application approved but not accepted

- Application denied by a financial institution
- Application withdrawn by the applicant
- File closed for incompleteness
- Loan originated
- Loan purchased by the institution
- Pre-Approval request approved but not accepted

- Pre- Approval request denied by financial institution

Our first job will be to take the above outcomes and translate them to a target category of either "candidate" or "not candidate". The above results introduce nuisance to the problem. Although a loan can be approved, an applicant may choose not to accept the loan. There is a distinction between a loan being accepted versus the loan being approved. An applicant can submit an application for a loan. That application may or may not be approved by the financial institution; however, in the event, the application is approved, the applicant has the choice to accept or not accept the loan/mortgage. Additionally, an applicant may not complete an application or withdraw an application before an approval decision has been made on application.

In the below table we provide an explanation for each of the outcomes and whether they would be an ideal candidate for the pilot program to receive assistance on securing a mortgage:

action_taken_value_desc	is_applicant_candidate
Application approved but not accepted	The loan is not accepted; however, the applicant demonstrates the capacity to obtain approval. Therefore, this applicant would not be a candidate for the pilot program
Application denied by financial institution	The application is denied by institution; this applicant would be an ideal candidate for the pilot program.
Application withdrawn by applicant	The application is withdrawn by the applicant; it is unclear whether approval would be granted if the submission was completed; however, since the withdrawal was voluntary, this application will not be considered an ideal candidate for the pilot program.
File closed for incompleteness	The applicant did not complete the application. This is an interesting situation because it is ambiguous as to whether the applicant would have been approved if loan if the application was completed. Without a completed application, this applicant will not be considered an ideal candidate for the pilot program.
Loan originated	The loan was approved and accepted. This applicant would not be an ideal candidate for the pilot program
Loan purchased by the institution	The institution has purchased the loan from the secondary market; in this outcome, the financial institution assumes being the new lender. This situation is equivalent to the loan being originated. This applicant would not be an ideal candidate for the pilot program.
Preapproval request approved but not accepted	The applicant receives approval, but not accept the loan. This is not an ideal candidate for the pilot program.
Preapproval request denied by financial institution	The applicant is denied approval for a loan. This applicant is an ideal candidate for the pilot program.

Deciding whether to include a candidate within the target was a difficult choice for some of the outcomes. For example, "Application withdrawn by applicant", would the applicant be denied a loan if they completed the application? Or would they have been approved for the mortgage? Per our rational, we assert that excluding potential candidates from the pilot program is more palatable then incorrectly adding improper candidates to the pilot program. This is further discussed below within the section "Cost of Wrong Predictions" where we discuss the topic in more detail. Per the above analysis of all the outcomes, the following 2 outcomes represent the ideal target (Target =1) for the pilot program to receive assistance on securing a mortgage/loan.

action_taken_value_target_is_1	target_value_candidate	target_label_candidate
Application denied by financial institution	1	CANDIDATE
Preapproval request denied by financial institution	1	CANDIDATE

The following outcomes detail situations where the applicant would not be an ideal candidate for the pilot program (Target = 0).

action_taken_value_target_is_0	target_value_not_candidate	target_label_not_candidate
Application approved but not accepted	0	NOT_CANDIDATE
Application withdrawn by applicant	0	NOT_CANDIDATE
File closed for incompleteness	0	NOT_CANDIDATE
Loan originated	0	NOT_CANDIDATE
Loan purchased by the institution	0	NOT_CANDIDATE
Preapproval request approved but not accepted	0	NOT_CANDIDATE

Our machine learning problem is to predict whether an applicant would be a good candidate for the pilot program (Target =1) or not a candidate (Target=0) for the pilot program; in other words, we want to identify potential applicants who would normally be rejected for a mortgage given the HDMA dataset which contains loan applications from Washington State in 2016.

As the target attributes do not currently exist within dataset, we will have to add those attributes with the following code:

```
#Add the target label to each row of either "CANDIDATE" or "NOT_CANDIDATE"
HDMA_dataset = cbind(HDMA_dataset, TARGET_LABEL= ifelse(
    HDMA_dataset$action_taken_name == "Application denied by financial institution" |
    HDMA_dataset$action_taken_name == "Preapproval request denied by financial institution",
    "CANDIDATE", "NOT_CANDIDATE" ))

#Add the target value to each row, where T=0=Not a candidate and if T=1=Candidate
HDMA_dataset = cbind(HDMA_dataset, TARGET_VALUE= ifelse(
    HDMA_dataset$TARGET_LABEL == "CANDIDATE" , 1, 0 ))

summary(HDMA_dataset$TARGET_LABEL)
    CANDIDATE NOT_CANDIDATE
        64212        402354
```

3.1 Cost of wrong predictions

Our model for predicating those who are valid candidates for the pilot program, just like any other model, will not be able to predict with 100% accuracy whether an applicant is a candidate or not. What if the model makes the wrong prediction? What is impact on the business context?

False Negative - Is a result where a prospective applicant who is suitable for the pilot program is wrongly classified as not a candidate. Although, this is not ideal, it is not a terrible situation either considering the business context. Recall, the business problem is to identify candiates for a pilot program to assist with securing a home mortgage. Ideally, we would like to predict this as accurately as possible; however, missing a couple of potential candidates might be reonsable because this is only a pilot program and typically a pilot program does not need to be inclusive of all potential candidates.

False Positive - Is a result where the applicant is NOT a candidate for the pilot program, but they have been wrongly classified as a candidate. We should take precaution to prevent this type of result. This would entail using government resources within a pilot program to assist an applicant who normally would not require the assistance. This could imply a waste of resources which would otherwise go to an applicant that could benefit from the pilot program.

When creating a confusion matrix for our model, we may need to revisit these tradeoffs and incorporate them into how we create the model. In summary, the cost of false negative and false positive are not the same. False positive clearly has a more negative impact on the business.

3.2 Limitations

Although the HDMA data includes various attributes for applicants. It does not contain names or unique identifiers for each applicant (in other words, Personally Identifiable Information). We may not be able to track if a particular applicant is making multiple applications. This by design as HDMA exists to bring transparncy to the mortgage/loan process and not place a spotlight on a particular individual. Additonally, applicants may or may not be a natural person (for example, small business). It is important to consider this factor when exploring categorical attributes such as gender, race, and ethnicity.

Within the target category of 1(Candidate for the program to receive assistance on obtaining mortgage), we have included mortgage outcomes where loans are rejected which seems reasonable. If someone is rejected for a loan, then they may need assistance with obtaining a loan; however, an appliant being rejected for a loan does not neccessairly imply that the applicant needs a home. For example, a wealthy applicant may already own a living residence, but they are applying for an additional rental property. And this may exacerbate the problem the business was originally trying to resolve. Recall, the business indicates that assistance is to be provided only to those who were rejected for a primary residance and not assist those who are trying to obtain additional properties. It might be difficult to ascertain this use case within the data.

4 Data Exploration

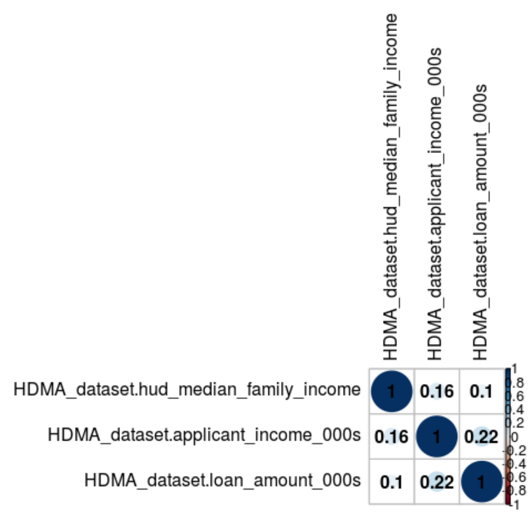
The HDMA dataset includes 47 attributes/columns (not including target attributes). Each row within the dataset represents an attempt from individual to secure a mortgage/loan from a financial institution.

4.1 Categorical and Numerical Values

For the purpose of exploration, we will need to investigate both numerical and categorical values.

4.2 Income, loan amount requested, a median income

It seems reasonable to investigate whether the amount of the loan requested would have a relation with the income of the applicant. Although, in practise, many factors impact whether a mortgage or loan is approved such as debt to income ratios and outstanding credit.

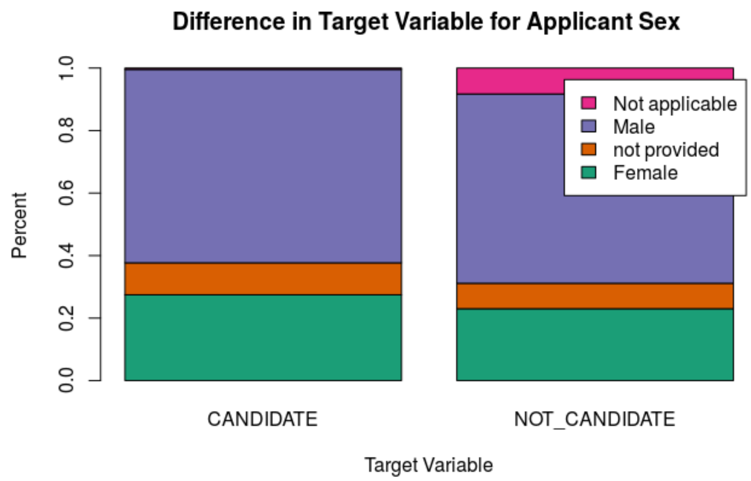


4.3 Potential for discrimination

In exploring data, we will consider whether the data suggests that gender, race, ethnicity impacts the candidacy of an applicant to receive a loan. Recall, that the target is to define whether someone would be a good candidate. We understand we will not be able to conclude or determine causation.

4.3.1 Applicant Gender

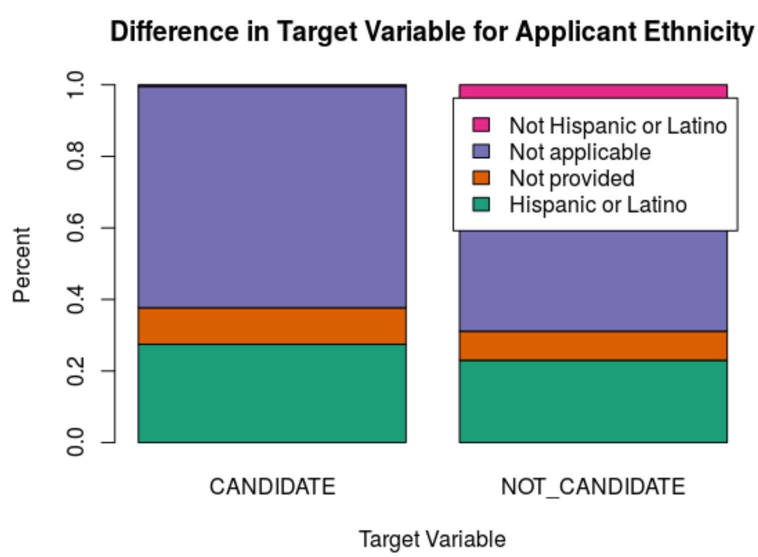
The gender values include values for “not applicable” and “not provided”. “Not applicable” could imply that individual’s gender identity does not conform with either male or female; or it could simply imply the individual is not a person and is rather a business entity. related be provided in respone to an individual whose gender identity is not defined by male or female.



Visually, there does not appear a significance difference in the categories. But of course, the ambiguity related to semantics of “not provided” and “not applicable” could have an impact. Additionally, “Not provided” and “male” are not strictly mutually exclusive categories (for example, a male applicant may choose not to provide their gender/sex during their application).

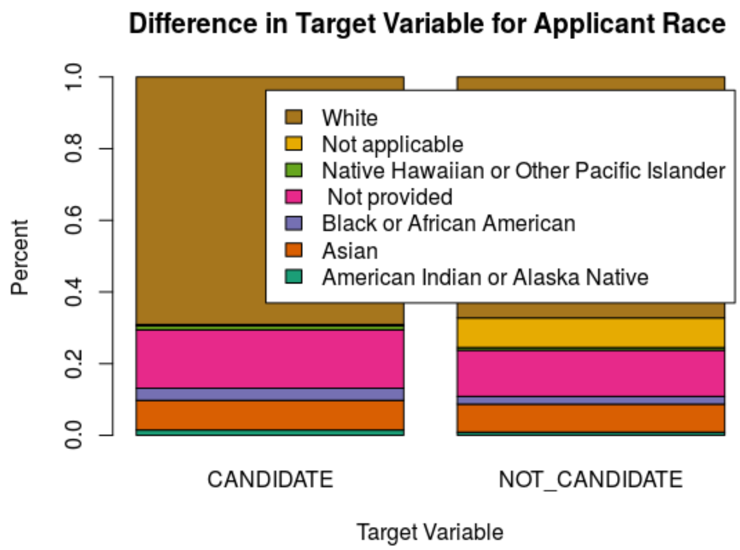
4.3.2 Ethnicity

The plot bellow shows the ethnicty of the applicant for the targe values. The range of values for ethnicity are not mutually exclusive.



4.3.3 Applicant Race

The plot below shows the race of the applicant against the target values.



4.4 Which attributes to ignore

As this dataset is rather large, we are also interested in omitting values which are either homogeneous or act as dependent values for the target.

4.4.1 Homogeneous attributes

For example, there are several homogeneous attribute within the target data set such as "as_of_year" which is 2016 for all values. As we find this attributes, we will choose not to include them in our feature selection and models.

4.4.2 Attributes that are depenent on the target

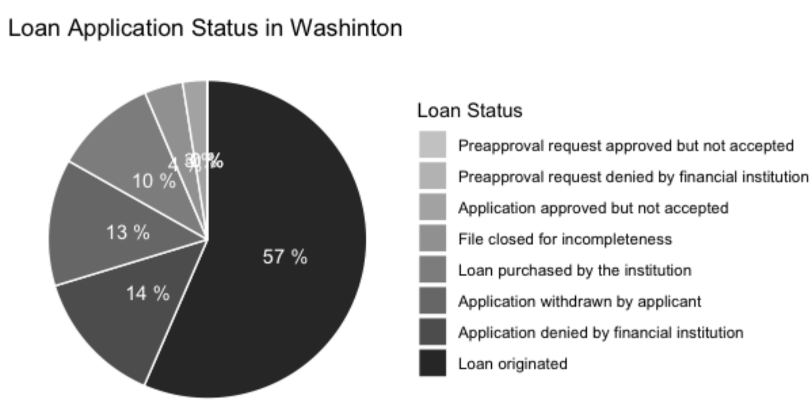
There are 3 attributes related to explicating a denial reason for the loan. For example, "denial_reason_name_1" provides a denial reason for the loan/mortgage which becomes rejected. This field is dependent on whether the loan is not approved and has no context for approved loans. These type of attributes we also not include in our feature selection and models.

4.5 Exploring other attributes

Additionally, we explored the other attributes to understand their relation to the target. Below, we have included code samples from various attributes we have visualized:

4.5.1 Loan Application Status

```
#############  Loan Application Status in Washington   ####################
Loan_status <- HDMA_dataset %>%
  group_by(action_taken_name) %>%
  summarize(count=n()) %>%
  arrange(desc(count))
head(Loan_status)
#Pie plot
ggplot(Loan_status, aes(x = "", y =  round(100*count/sum(count), 1),
                        fill =  reorder(action_taken_name,count))) +
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = cumsum(100*count/sum(count)) - 0.5*(100*count/sum(count)),
              label = paste(round(count/sum(count)*100),"%")), color = "white")+
  ggtitle("Loan Application Status in Washington")+
  #scale_fill_brewer("Loan Status") + theme_void()
  scale_fill_grey(start = 0.8, end = 0.2,"Loan Status") + theme_void()
```

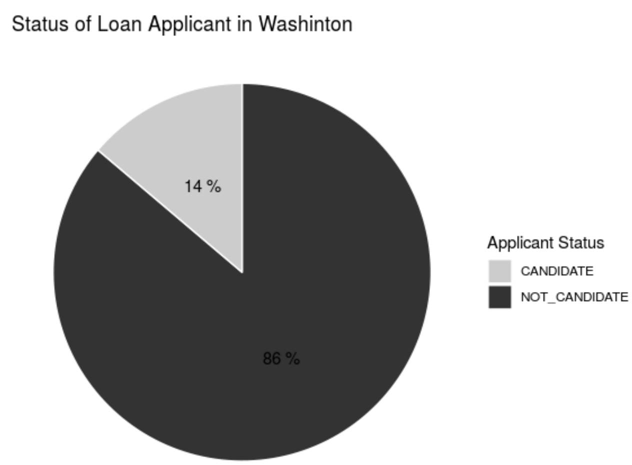


4.5.2 Applicant status in Washington

```
############# Applicant  Status in Washington   ####################
Canditate_status <- HDMA_dataset %>%
  group_by(TARGET_LABEL) %>%
  summarize(count=n()) %>%
  arrange(desc(count))
```

```r
head(Canditate_status)

#Pie plot
ggplot(Canditate_status, aes(x = "", y =  round(100*count/sum(count), 1),
                             fill =  reorder(TARGET_LABEL,count))) +
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = cumsum(100*count/sum(count)) - 0.5*(100*count/sum(count)),
              label = paste(round(count/sum(count)*100),"%")), color = "black")+
  ggtitle(" Status of Loan Applicant in Washington")+
  scale_fill_grey(start = 0.8, end = 0.2,"Applicant Status") + theme_void()
```

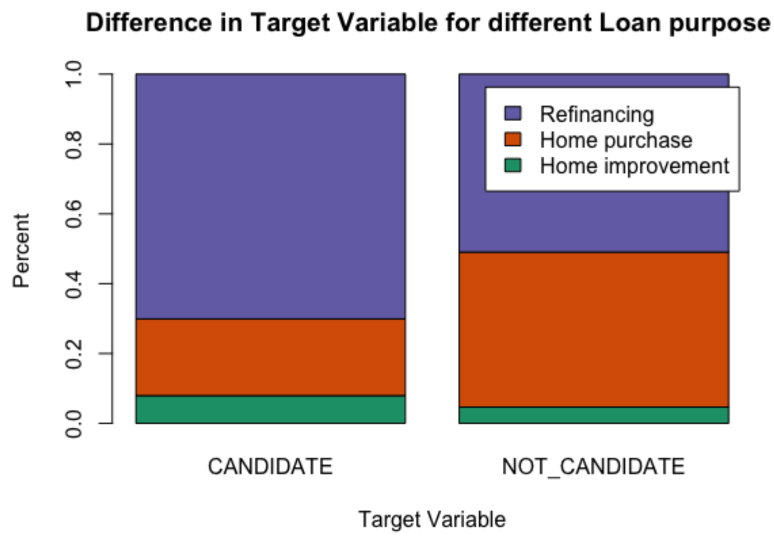
Of the outcomes, the distribution of candidates and non-candidates is shown below:



4.5.3 Loan Purpose

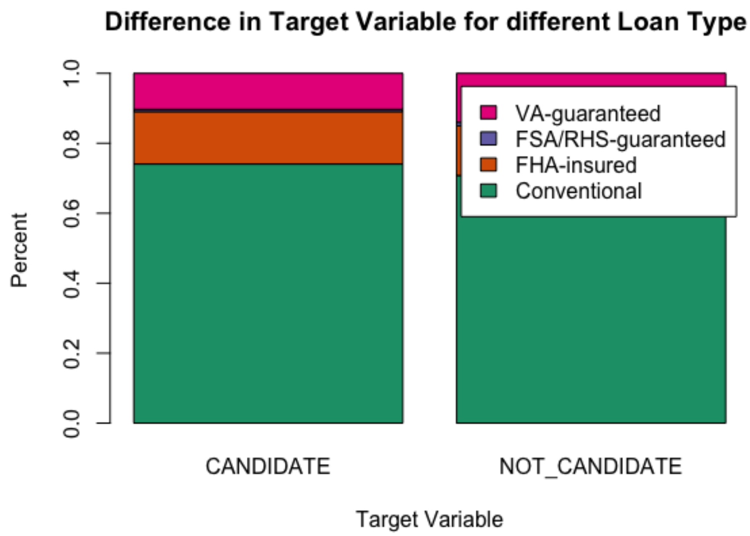
```r
################## Loan Purpose  Vs TARGET_LABEL ########################
# Create contigency table
#Loan_purpose = table(HDMA_dataset$loan_purpose_name,HDMA_dataset$TARGET_LABEL)
#head(Loan_purpose)
#freq_tbl = table(HDMA_dataset$loan_purpose_name)
#round(prop.table(freq_tbl),2)

# Creating crosstabs for categorical variable
Loan_purpose_xtab = xtabs(~HDMA_dataset$loan_purpose_name+HDMA_dataset$TARGET_LABEL)
head(Loan_purpose_xtab)
round(prop.table(Loan_purpose_xtab),2)
#display.brewer.all()
```

```r
barplot(prop.table(Loan_purpose_xtab,2),
        legend = rownames(Loan_purpose_xtab), beside = T,
        ylab = "Percent", xlab = "Target Variable",
        col = brewer.pal(3, name = "Dark2"),
        main = "Difference in Target Variable for different Loan purpose ")

barplot(prop.table(Loan_purpose_xtab,2),
        legend = rownames(Loan_purpose_xtab),
        ylab = "Percent", xlab = "Target Variable",
        col = brewer.pal(3, name = "Dark2"),
        main = "Difference in Target Variable for different Loan purpose ")
```



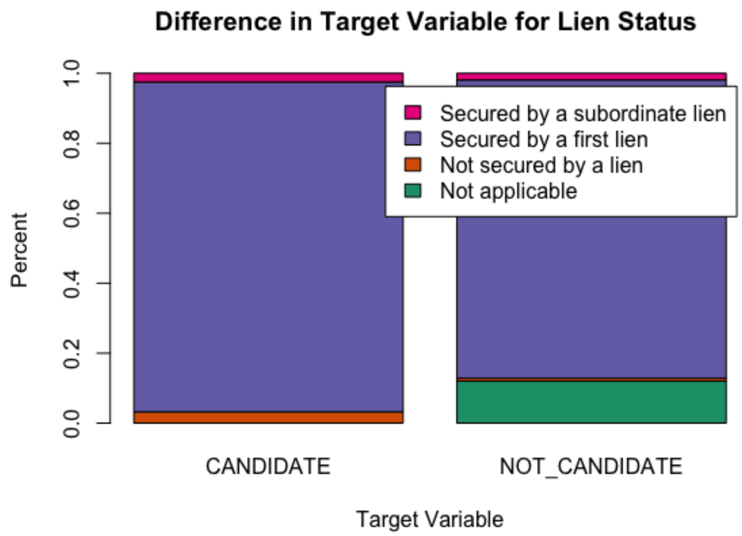
4.5.4 Loan Type

```r
################## Loan Type  Vs TARGET_LABEL (Applicant Status) ########################
# Creating crosstabs for categorical variable
Loan_type_xtab = xtabs(~HDMA_dataset$loan_type_name+HDMA_dataset$TARGET_LABEL)
head(Loan_type_xtab)
round(prop.table(Loan_type_xtab),2)
#display.brewer.all()
barplot(prop.table(Loan_type_xtab,2),
        legend = rownames(Loan_type_xtab),
        ylab = "Percent", xlab = "Target Variable",
        col = brewer.pal( 4, name = "Dark2"),
        main = "Difference in Target Variable for different Loan Type ")
```



4.5.5 Lien Status

```
################## Lien Status  Vs TARGET_LABEL ########################
# Creating crosstabs for categorical variable

Lien_status_xtab = xtabs(~HDMA_dataset$lien_status_name+HDMA_dataset$TARGET_LABEL)
head(Lien_status_xtab)
round(prop.table(Lien_status_xtab),2)
#display.brewer.all()
barplot(prop.table(Lien_status_xtab,2),
        legend = rownames(Lien_status_xtab) ,
        ylab = "Percent", xlab = "Target Variable",
        col = brewer.pal( 4, name = "Dark2"),
        main = "Difference in Target Variable for Lien Status ")
```



4.5.6 HOEPA status

```
################## HOEPA Status  Vs TARGET_LABEL ########################
# Creating crosstabs for categorical variable

HOEPA_status_xtab = xtabs(~HDMA_dataset$hoepa_status_name+HDMA_dataset$TARGET_LABEL)
head(HOEPA_status_xtab)
round(prop.table(HOEPA_status_xtab),2)
#display.brewer.all()
barplot(prop.table(HOEPA_status_xtab,2),
        legend = rownames(HOEPA_status_xtab) ,
        ylab = "Percent", xlab = "Target Variable",
        col = brewer.pal( 2, name = "Dark2"),
        main = "Difference in Target Variable for Lien Status ")


################## HOEPA Status  Vs TARGET_LABEL ########################
# Creating crosstabs for categorical variable

HOEPA_status_xtab = xtabs(~HDMA_dataset$hoepa_status_name+HDMA_dataset$TARGET_LABEL)
head(HOEPA_status_xtab)
round(prop.table(HOEPA_status_xtab),2)
#display.brewer.all()
barplot(prop.table(HOEPA_status_xtab,2),
        legend = rownames(HOEPA_status_xtab) ,
```

```
        ylab = "Percent", xlab = "Target Variable",
        col = brewer.pal( 2, name = "Dark2"),
        main = "Difference in Target Variable for Lien Status ")
```

4.5.7 Comparing applicants with county

```
################# Applicant CANDIDATE from different county in Washington   ########################
#County_name <- data.frame(HDMA_dataset$county_name,HDMA_dataset$TARGET_LABEL)
#head(County_name)

 library(dplyr)
County_name_of_Candidate <- HDMA_dataset %>%
  filter(TARGET_LABEL == "CANDIDATE")%>%
  group_by(county_name) %>%
  summarize(count=n()) %>%
  arrange(desc(count))

ggplot(County_name_of_Candidate, aes(x=reorder(county_name,count), y=count )) +
  geom_bar(stat="identity")+
  #ggtitle("") +
  #xlab("") + ylab("")
  labs(title = "County of Applicants CANDIDATE", x ="County name" , y =  "Nber of CANDIDATE")+
  coord_flip()+  scale_fill_brewer(palette="Greys")+
  theme_minimal()

library(dplyr)
County_name_of_NotCandidate <- HDMA_dataset %>%
  filter(TARGET_LABEL == "NOT_CANDIDATE")%>%
  group_by(county_name) %>%
  summarize(count=n()) %>%
  arrange(desc(count))

head(County_name_of_NotCandidate)

County <- County_name_of_Candidate %>%
  mutate(NOT_CANDIDATE=County_name_of_NotCandidate$count)%>%
  rename(CANDIDATE=count)
head(County)
```

5 Data Cleaning/Preparing

During the data cleaning step, we will proceed to identify missing values and impute them.

5.1 Missing Values

There are many missing values within the dataset, our first task is to identify missing values:

```
> #Check the data for missing values.
> sapply(HDMA_Processed, function(x) sum(is.na(x)))
        tract_to_msamd_income                   rate_spread                    population
                          615                        457928                           610
           minority_population number_of_owner_occupied_units number_of_1_to_4_family_units
                          610                           622                           611
              loan_amount_000s       hud_median_family_income         applicant_income_000s
                            0                           606                         62033
                    state_name                    state_abbr               sequence_number
                            0                             0                             0
                 respondent_id            purchaser_type_name            property_type_name
                            0                             0                             0
              preapproval_name          owner_occupancy_name                    msamd_name
                            0                             0                         38274
                loan_type_name             loan_purpose_name              lien_status_name
                            0                             0                             0
             hoepa_status_name              edit_status_name           denial_reason_name_3
                            0                        392061                        465320
          denial_reason_name_2           denial_reason_name_1                   county_name
                       459820                        432067                           367
         co_applicant_sex_name      co_applicant_race_name_5      co_applicant_race_name_4
                            0                        466552                        466545
      co_applicant_race_name_3      co_applicant_race_name_2      co_applicant_race_name_1
                       466461                        464704                             0
   co_applicant_ethnicity_name           census_tract_number                    as_of_year
                            0                           606                             0
    application_date_indicator            applicant_sex_name         applicant_race_name_5
                            0                             0                        466520
         applicant_race_name_4         applicant_race_name_3         applicant_race_name_2
                       466498                        466269                        462088
         applicant_race_name_1      applicant_ethnicity_name                   agency_name
                            0                             0                             0
                   agency_abbr             action_taken_name                  TARGET_LABEL
```

```
                        0                               0                               0
              TARGET_VALUE
                        0
```

Based on our data exploration and understanding of the business problem, we've decided to take a subset of "interesting" attributes (categorical and numerical) to perform cleaning on:

```
> sapply(HDMA_select_data, function(x) sum(is.na(x)))
         tract_to_msamd_income                      population            minority_population
                           615                             610                            610
number_of_owner_occupied_units  number_of_1_to_4_family_units                loan_amount_000s
                           622                             611                              0
      hud_median_family_income             applicant_income_000s             property_type_name
                           606                           62033                              0
                loan_type_name                loan_purpose_name              hoepa_status_name
                             0                               0                              0
            applicant_sex_name             applicant_race_name_1        applicant_ethnicity_name
                             0                               0                              0
                  TARGET_VALUE
                             0
>
```

5.2 Imputation

We have used MICE package to assist with imputing the data for the purpose of cleaning. The MICE package inclues PMM (Predicative Mean Matching) for numeric variables. We've applied imputation to our subset of interesting numeric values.

```
HDMA_Processed <- read.csv('../Data/Working/HDMA_select_imputed.csv',
                           header = TRUE, na.strings = c("NA","","#NA"))
#str(HDMA_Processed)

#Check the data for missing values.
sout <- sapply(HDMA_Processed, function(x) sum(is.na(x)))
kable(sout)

# Select Interesting variables
NumColumn  <- c("tract_to_msamd_income", "population", "minority_population",
                "number_of_owner_occupied_units", "number_of_1_to_4_family_units",
                "loan_amount_000s","hud_median_family_income","applicant_income_000s")
#
CategoColum = c("property_type_name","loan_type_name","loan_purpose_name","hoepa_status_name",
```

```
                "applicant_sex_name","applicant_race_name_1","applicant_ethnicity_name")

WorkingColumn= c(NumColumn,CategoColum,"TARGET_VALUE")
#data <- subset(data, select = -c(TARGET_VALUE))
HDMA_select_data = subset(HDMA_Processed,select=WorkingColumn )

#head(HDMA_select_data)
#Check the data for missing values.
sapply(HDMA_select_data, function(x) sum(is.na(x)))

#==============================
# set Categorical varibla as factor

HDMA_select_data <- HDMA_select_data %>%
  mutate(
    TARGET_VALUE  = as.numeric(TARGET_VALUE),
    property_type_name = as.factor(property_type_name),
    loan_type_name = as.factor(loan_type_name),
    loan_purpose_name =  as.factor(loan_purpose_name),
    hoepa_status_name =  as.factor(hoepa_status_name),
    applicant_sex_name =  as.factor(applicant_sex_name),
    applicant_race_name_1 =  as.factor(applicant_race_name_1),
    applicant_ethnicity_name = as.factor( applicant_ethnicity_name)
  )

#Look the dataset structure.
#str(HDMA_select_data)


#============ Visualize missing Data ========================================
#Calculates every unique combination of missing data & shows of times that happens
HDMA_select_num_subset  <- subset(HDMA_select_data, select = c(NumColumn) )
md.pattern(HDMA_select_num_subset)
md.pattern(HDMA_select_num_subset,rotate.names = TRUE)


#===========================  IMPUTATION( MICE package)   =======================
#recisely, the methods used by this package are:
#1)-PMM (Predictive Mean Matching) - For numeric variables
#2)-logreg(Logistic Regression) - For Binary Variables( with 2 levels)
```

```
#3)-polyreg(Bayesian polytomous regression) - For Factor Variables (>= 2 levels)
#4)-Proportional odds model (ordered, >= 2 levels)
#==============================================================================

init = mice(HDMA_select_data, maxit=0)
meth = init$method
predM = init$predictorMatrix

##remove the variable as a predictor but still will be imputed. Just for illustration purposes,
##I select the "TARGET_VALUE" variable to not be included as predictor during imputation.
predM[, c(CategoColum)]=0

##If you want to skip a variable from imputation use the code below.
##This variable will still be used for prediction.
#++++++++++++++++++++++++++++++++++
#meth[c("Variable")]=""
meth[c("TARGET_VALUE")]=""
meth[c("property_type_name")]=""
meth[c("loan_type_name")]=""
meth[c("loan_purpose_name")]=""
meth[c("hoepa_status_name")]=""
meth[c("applicant_sex_name")]=""
meth[c("applicant_race_name_1")]=""
meth[c("applicant_ethnicity_name")]=""
#++++++++++++++++++++++++++++++++++

##Now let specify the methods for imputing the missing values.
## we impute only the Numerical Variable
meth[c("tract_to_msamd_income")]="pmm"
meth[c("population")]="pmm"
meth[c("minority_population")]="pmm"
meth[c("number_of_owner_occupied_units")]="pmm"
meth[c("number_of_1_to_4_family_units")]="pmm"
meth[c("loan_amount_000s")]="pmm"
meth[c("hud_median_family_income")]="pmm"
meth[c("applicant_income_000s")]="pmm"
#meth[c("property_type_name")]="norm"
#meth[c("loan_type_name")]="logreg"
#meth[c("loan_purpose_name")]="polyreg"
```

```r
set.seed(103)
 imputed = mice(HDMA_select_data, method=meth, predictorMatrix=predM, m=5)

#Create a dataset after imputation.
HDMA_select_imputed<- complete(imputed)

#Check for missings in the imputed dataset.
sapply(HDMA_select_imputed, function(x) sum(is.na(x)))

HDMA_Processed_imputed <-  HDMA_Processed%>%
  mutate(
      tract_to_msamd_incomem = HDMA_select_imputed$tract_to_msamd_incomem,
      population = HDMA_select_imputed$population,
      minority_population = HDMA_select_imputed$minority_population,
      number_of_owner_occupied_units = HDMA_select_imputed$number_of_owner_occupied_units,
      number_of_1_to_4_family_units = HDMA_select_imputed$number_of_1_to_4_family_units,
      loan_amount_000s = HDMA_select_imputed$loan_amount_000s,
      hud_median_family_income = HDMA_select_imputed$hud_median_family_income,
      applicant_income_000s = HDMA_select_imputed$applicant_income_000s,
  )


#write.csv(HDMA_Processed_imputed , "HDMA_Processed_imputed.csv")

#===============================Accuracy========================================
#### Var1
#actual <- original$Var1[is.na(dat$Var1)] #
#predicted <- imputed$Var1l[is.na(dat$Var1)]
##### Var2
#actual <- original$Var2[is.na(dat$Var2)]
#predicted <- imputed$Var2[is.na(dat$Var2)]

#table(actuals)
#table(predicted)
#mean(actual)
#mean(predicted)
#==================================================================================


####################################################################################
```

```
##########################                       ###############################
##########################  Correlation Matrix    ###############################
#################################################################################
#HDMA_select_imputed  <- read.csv(file.choose(), header = TRUE, na.strings = c("NA","","#NA"))
#str(HDMA_select_imputed)

#================= Select numerical variables ==================================

NumColumn  <- c("tract_to_msamd_income", "population", "minority_population",
                "number_of_owner_occupied_units", "number_of_1_to_4_family_units",
                "loan_amount_000s","hud_median_family_income","applicant_income_000s")




HDMA_select_num_subset  <- subset(HDMA_select_imputed, select = c(NumColumn,"TARGET_VALUE") )

#Correlation Matrix - default one in R
cor(subset(HDMA_select_num_subset,select = -c(TARGET_VALUE)))
#correlation matrix with statistical significance

cor_result=rcorr(as.matrix(subset(HDMA_select_num_subset,select = -c(TARGET_VALUE))))

#The function corrplot() takes the correlation matrix as the first argument.
#The second argument (type="upper")
#is used to display only the upper triangular of the correlation matrix.
corrplot(cor_result$r, type = "upper", order = "hclust", tl.col = "black", tl.srt = 90)
```

After imputing the values, we can confirm there are no more missing values within the dataset.

```
> #Check for missings in the imputed dataset.
> sapply(HDMA_select_imputed, function(x) sum(is.na(x)))
                          X          tract_to_msamd_income                     population
                          0                              0                              0
        minority_population number_of_owner_occupied_units  number_of_1_to_4_family_units
                          0                              0                              0
           loan_amount_000s       hud_median_family_income          applicant_income_000s
                          0                              0                              0
         property_type_name                 loan_type_name              loan_purpose_name
                          0                              0                              0
           hoepa_status_name             applicant_sex_name          applicant_race_name_1
                          0                              0                              0
```

```
    applicant_ethnicity_name                       TARGET_VALUE
                           0                                  0
```

6 Model: Logistic Regression

Now that we have concluded data exploration and data cleaning, we can proceed with creating a model. We have decided to use the logistic regression algorithm to help predict the outcome of this problem. In the event the results are not satisfactory, we may pursue other algorithms. Below, we have detailed 3 models using the logistic regression algorithm.

6.1 Training and Test Data

We performed sampling to segregate the dataset into training and testing datasets: We used 75% for the training data and 25% for the testing data. The dataset was splitted randomly. We will use these training sets in creating our models.

6.2 Model creation

For the first attempt a model is created with larger number of variables model 1 (logit 1). The code of the model is presented below as well as the summary. One can notice that the p-values for logit 1 indicate that "number_of_owner_occupied_units" variable could be potentially removed from the model.

This led to model 2 (logit 2) where the "number_of_owner_occupied_units" was removed as a predictor. The code is listed bellow.

```
##=========================     TRANING DATA     ================================
#mylogit <- glm(TARGET_VALUE ~ loan_amount_000s + hud_median_family_income + applicant_income_000s, dat

logit1 <- glm(TARGET_VALUE ~tract_to_msamd_income +
           population                      +
           minority_population             +
           number_of_owner_occupied_units  +
           number_of_1_to_4_family_units   +
           loan_amount_000s                +
           hud_median_family_income        +
           applicant_income_000s,family=binomial(link='logit'),data=train1, maxit = 100)

 > summary(logit1)

Call:
```

```
glm(formula = TARGET_VALUE ~ tract_to_msamd_income + population +
    minority_population + number_of_owner_occupied_units + number_of_1_to_4_family_units +
    loan_amount_000s + hud_median_family_income + applicant_income_000s,
    family = binomial(link = "logit"), data = train1, maxit = 100)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.8215  -0.5707  -0.5303  -0.4773   5.8257

Coefficients:
                                 Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -8.837e-01  4.676e-02 -18.897  < 2e-16 ***
tract_to_msamd_income          -1.784e-03  2.416e-04  -7.383 1.54e-13 ***
population                     -3.890e-05  5.666e-06  -6.866 6.59e-12 ***
minority_population             6.961e-03  3.989e-04  17.451  < 2e-16 ***
number_of_owner_occupied_units -2.052e-05  2.729e-05  -0.752   0.4522
number_of_1_to_4_family_units   9.101e-05  1.643e-05   5.540 3.03e-08 ***
loan_amount_000s               -5.529e-05  2.253e-05  -2.455   0.0141 *
hud_median_family_income       -8.953e-06  4.547e-07 -19.691  < 2e-16 ***
applicant_income_000s          -1.659e-03  7.722e-05 -21.491  < 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: 280382  on 349924  degrees of freedom
Residual deviance: 277821  on 349916  degrees of freedom
AIC: 277839

Number of Fisher Scoring iterations: 6


# logit2 : we remove "number_of_owner_occupied_units" as predictor
logit2<-glm(TARGET_VALUE~tract_to_msamd_income +
           number_of_owner_occupied_units    +
           number_of_1_to_4_family_units     +
           loan_amount_000s                  +
           hud_median_family_income          +
           applicant_income_000s, data=train1,
          family=binomial(link='logit'))
```

```
> summary(logit2)

Call:
glm(formula = TARGET_VALUE ~ tract_to_msamd_income + population +
    number_of_owner_occupied_units + number_of_1_to_4_family_units +
    loan_amount_000s + hud_median_family_income + applicant_income_000s,
    family = binomial(link = "logit"), data = train1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7297  -0.5724  -0.5336  -0.4797   5.8305

Coefficients:
                                 Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -7.586e-01  4.628e-02 -16.389  < 2e-16 ***
tract_to_msamd_income          -2.726e-03  2.364e-04 -11.532  < 2e-16 ***
population                      5.979e-06  5.005e-06   1.195   0.2322
number_of_owner_occupied_units -1.335e-04  2.639e-05  -5.058 4.23e-07 ***
number_of_1_to_4_family_units   6.693e-05  1.641e-05   4.078 4.54e-05 ***
loan_amount_000s               -5.562e-05  2.251e-05  -2.471   0.0135 *
hud_median_family_income       -7.467e-06  4.486e-07 -16.645  < 2e-16 ***
applicant_income_000s          -1.664e-03  7.724e-05 -21.547  < 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: 280382  on 349924  degrees of freedom
Residual deviance: 278121  on 349917  degrees of freedom
AIC: 278137

Number of Fisher Scoring iterations: 6
```

##Interpretion of the coefficients in the multiple regression

The output includes a summary of deviance residuals and Akaike information criterion (AIC) which are both measures of fit. It also includes coefficients associated with each independent variable and the intercept. Log of odds is rather hard to interpret, therefore, we often take the exponential of the coefficients.

```
> round(exp(coef(logit1)),3)
                   (Intercept)          tract_to_msamd_income
```

```
                          0.382                          0.999
                     population             minority_population
                          1.000                          1.007
number_of_owner_occupied_units  number_of_1_to_4_family_units
                          1.000                          1.000
               loan_amount_000s        hud_median_family_income
                          1.000                          1.000
          applicant_income_000s
                          0.999

> round(exp(coef(logit2)),3)
                    (Intercept)           tract_to_msamd_income
                          0.446                          0.997
                     population number_of_owner_occupied_units
                          1.000                          1.000
  number_of_1_to_4_family_units               loan_amount_000s
                          1.000                          1.000
       hud_median_family_income           applicant_income_000s
                          1.000                          0.998
```

It is important to keep in mind that with more than one independent variable, the interpretation of the coefficient on an independent variables is the effect of that independent variable holding all other independent variables constant. For example, holding all the variables constant except loan_amount, the effect of raising this later score by 1 point raises the odds of being a candidate by 1.00 percent.

6.3 Variance inflation factor

Another assumption of a classical regression model is that there is collinearity between the explanatory variables. This can be easily identified by the VIF - Variance Inflation Factor that is able to detect the multicollinearity between the variable. An empirical rule-of-thumb is that when VIF more significant than 10 for any, the multicollinearity is an important consideration. The VIF for the transformed model was calculated using the **vif()** function in R, and the result is presented in Figure 8. The variables selected have VIF lower than 4, and therefore no multicollinearity is presented in the fitted model.

As models 1 and 2 use numerical values, we can use Variance Inflation Factor (VIF) to check how the variance of estimated regression coefficients are increased due to collinearity.

```
tab_model(logit1, logit2)

ans1<-vif(logit1)
barplot(ans1, col = "blue", ylab = "VIF", ylim =c(0,20),las=2, space=1)
abline(h=10, col="red", lwd=2)
```

```r
ans2<-vif(logit2)
barplot(ans2, col = "blue", ylab = "VIF", ylim =c(0,20),las=2, space=1)
abline(h=10, col="red", lwd=2)
```

6.4 Confidence Intervals

We also calculated the confidence intervals for the coefficient estimates.
The confidence intervals presented are from 2.5% to 97.5%

```r
#We can use the confint function to obtain confidence intervals for the coefficient estimates.

## CIs using profiled log-likelihood
#confint(selectedModel)
## CIs using standard errors
confint.default(logit1)
confint.default(logit2)

> confint.default(selectedModel1)
                                      2.5 %        97.5 %
(Intercept)                   -9.753304e-01 -7.920264e-01
tract_to_msamd_income         -2.257377e-03 -1.310310e-03
population                    -5.000639e-05 -2.779768e-05
minority_population            6.179163e-03  7.742778e-03
number_of_owner_occupied_units -7.400240e-05  3.297006e-05
number_of_1_to_4_family_units  5.881058e-05  1.232062e-04
loan_amount_000s              -9.944379e-05 -1.114391e-05
hud_median_family_income      -9.843907e-06 -8.061627e-06
applicant_income_000s         -1.810758e-03 -1.508079e-03


> confint.default(selectedModel2)
                                     2.5 %        97.5 %
(Intercept)                   -8.492857e-01 -6.678554e-01
tract_to_msamd_income         -3.188910e-03 -2.262401e-03
population                    -3.830311e-06  1.578879e-05
number_of_owner_occupied_units -1.852289e-04 -8.177189e-05
number_of_1_to_4_family_units  3.476038e-05  9.909166e-05
loan_amount_000s              -9.973609e-05 -1.149441e-05
hud_median_family_income      -8.346057e-06 -6.587603e-06
applicant_income_000s         -1.815761e-03 -1.512976e-03
```

6.5 ROC curve and AUC

The Receiver Operating Characteristic (ROC) curve compares the rank of prediction and answer. The Area under the curve (AUC) is a typical performance measurement for a binary classifier. AUC closer to 1 is ideal

We have calculated the accuracy and "Area Under the Curve" for our models. The accuracy of the the two models selected are high, around 0.86 as indicated below. Additionally, the AUC for model 1 and 2 was also similar around 0.586 and 0.584, respectively.

This is promising; however, we will perform cross validation to further validate our findings.

```
> fitted1 <- predict(logit1,test1,type='response')
> #Our decision boundary will be 0.5. If P(y=1|X) > 0.5 then y = 1 otherwise y=0.
> fitted1 <- ifelse(fitted1 > 0.5,1,0)
> misClasificError1 <- mean(fitted1 != test1$TARGET_VALUE)
# Accuracy_logit1 = 1-misClasificError1
predicted_1_factor <- factor(fitted1, levels=levels(test1$TARGET_VALUE))
str(fitted1)
> confusionMatrix(predicted_1_factor,test1$TARGET_VALUE)
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 100588  16053
         1      0      0

               Accuracy : 0.8624
                 95% CI : (0.8604, 0.8643)
    No Information Rate : 0.8624
    P-Value [Acc > NIR] : 0.5021

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8624
         Neg Pred Value :    NaN
             Prevalence : 0.8624
         Detection Rate : 0.8624
   Detection Prevalence : 1.0000
```

```
      Balanced Accuracy : 0.5000


       'Positive' Class : 0



> auc1 <- performance(pr1, measure = "auc")
> auc1 <- auc1@y.values[[1]]
> print(paste('AUC logit1', round(auc1,2)))
[1] "AUC logit1 0.58"
```

```
 fitted2 <- predict(logit2,test1,type='response')
> fitted2 <- ifelse(fitted2 > 0.5,1,0)
> misClasificError2 <- mean(fitted2 != test1$TARGET_VALUE)
> Accuracy_logit2 = 1-misClasificError2
> print(paste('Accuracy', round(Accuracy_logit2,2)))
[1] "Accuracy 0.86"

> predicted_2_factor <- factor(fitted2, levels=levels(test1$TARGET_VALUE))
>
> confusionMatrix(predicted_2_factor,test1$TARGET_VALUE)
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 100588  16053
         1      0      0

               Accuracy : 0.8624
                 95% CI : (0.8604, 0.8643)
    No Information Rate : 0.8624
    P-Value [Acc > NIR] : 0.5021

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8624
         Neg Pred Value :    NaN
             Prevalence : 0.8624
```

```
        Detection Rate : 0.8624
  Detection Prevalence : 1.0000
     Balanced Accuracy : 0.5000

      'Positive' Class : 0

> auc2 <- performance(pr2, measure = "auc")
> auc2 <- auc2@y.values[[1]]
> print(paste('AUC logit2', round(auc2,2)))
[1] "AUC logit2 0.58"
```

We set the cut off for classifying a person as candidat at 0.50 probability. This gave us sensitivity (probability of detecting NOT_CANDIDATE among NOT_CANDIDATE = True positive rate) of 1.00, and specificity (probability of detecting CANDIDATE among CANDIDATE = False positive rate) of 0.00.

We have accuracy of 86%. We detect NOT_CANDIDATE in 100% of cases, and we label CANDIDATE as CANDIDATE in 00% of cases. The first and second models both perform AUC at about 0.58. This is promising;

However, we will perform cross validation to further validate our findings.

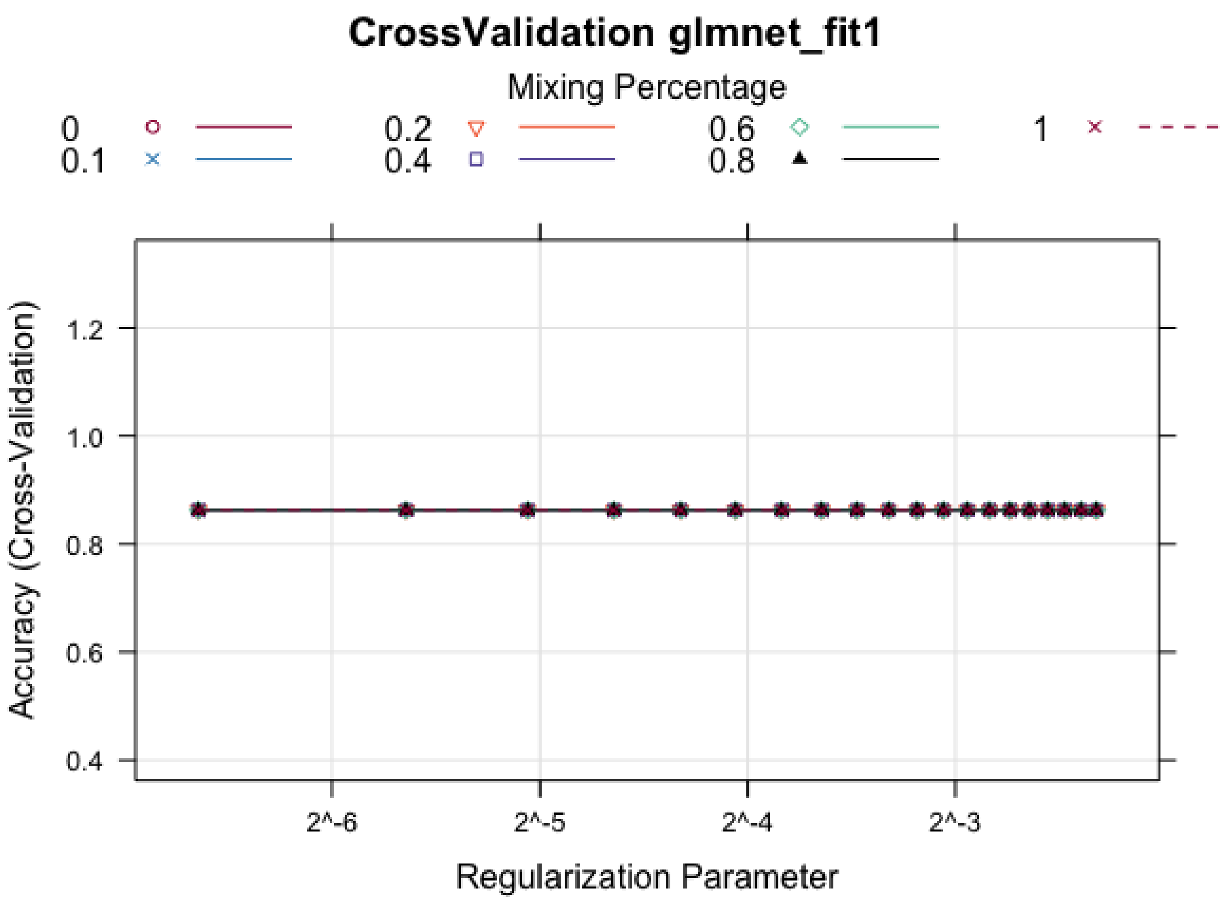
6.6 Cross-Validation

The cross validation is perfromed according to the model 2

```
HDMA_select_num_subset$TARGET_VALUE =  as.factor(HDMA_select_num_subset$TARGET_VALUE)

seed <- 1500
classes1 <- HDMA_select_num_subset[, "TARGET_VALUE"]
str(classes1)
#Exclude "minority_population" as predictor == logit2
predictors1 <- HDMA_select_num_subset[, -match(c("TARGET_LABEL","TARGET_VALUE","minority_population",),
predictors1
require(caret)
train_set1 <- createDataPartition(classes1, p = 0.75, list = FALSE)
str(train_set1)

train_predictors1 <- predictors1[train_set1, ]
train_classes1 <- classes1[train_set1]
test_predictors1 <- predictors1[-train_set1, ]
test_classes1 <- classes1[-train_set1]
str(train_predictors1)
```

```
set.seed(seed)
cv_splits1 <- createFolds(classes1, k = 10, returnTrain = TRUE)
str(cv_splits1)

require(glmnet)

set.seed(seed)

HDMA_num_subset_train <- HDMA_select_num_subset[train_set1, ]
HDMA_num_subset_test <- HDMA_select_num_subset[-train_set1, ]

glmnet_grid1 <- expand.grid(alpha = c(0,  .1,  .2, .4, .6, .8, 1),
                            lambda = seq(.01, .2, length = 20))
glmnet_ctrl1 <- trainControl(method = "cv", number = 10)
#================= Training
glmnet_fit1 <- train(TARGET_VALUE ~ tract_to_msamd_income +
           population                       +
           number_of_owner_occupied_units   +
           number_of_1_to_4_family_units    +
           loan_amount_000s                 +
           hud_median_family_income         +
           applicant_income_000s,  data = HDMA_num_subset_train,
                     method = "glmnet",
                     preProcess = c("center", "scale"),
                     tuneGrid = glmnet_grid1,
                     trControl = glmnet_ctrl1)

> glmnet_fit1
glmnet

349925 samples
     7 predictor
     2 classes: '0', '1'

Pre-processing: centered (7), scaled (7)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 314933, 314932, 314933, 314932, 314932, 314934, ...
Resampling results across tuning parameters:

  alpha  lambda  Accuracy   Kappa
```

```
  0.0    0.01    0.8623734  0
  0.0    0.02    0.8623734  0
  0.0    0.03    0.8623734  0
  0.0    0.04    0.8623734  0
  :
  :
  :
  1.0    0.20    0.8623734  0

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were alpha = 0 and lambda = 0.2



trellis.par.set(caretTheme())
plot(glmnet_fit1, scales = list(x = list(log = 2)))
#post(glmnet_fit1, file = "CrossValidation1.ps",
     #title = "Cross Validation for  HDMA Numeric Varible")
```



The previous plot shows the “accuracy”, that is the percentage of correctly classified observations,for the logistic regression model with each combination of the two tuning parameters α and λ. The optimal tuning parameter values are $\alpha = 0$ and $\lambda = 0.2$.

Then, it is possible to predict new samples with the identified optimal model using the predict method:

```
> #================ TESTING
> pred_classes1 <- predict(glmnet_fit1, newdata = HDMA_num_subset_test)
> table(pred_classes1)
pred_classes1
     0      1
116641      0



> pred_probs1 <- predict(glmnet_fit1, newdata = HDMA_num_subset_test, type = "prob")
> head(pred_probs1)
          0         1
1 0.8455307 0.1544693
2 0.8573492 0.1426508
3 0.8771290 0.1228710
4 0.8739576 0.1260424
5 0.8604260 0.1395740
6 0.8591333 0.1408667
```

6.7 Better solution?

Given our first 2 models use only numerical attributes. We attempted to produce a third model with logistic regression using both numerical and categorical attributes

6.8 Third model attempt (categorical/factor and numerical attributes)

For this we will include the followng categorical attributes: “loan_type_name” and “loan_purpose_name”

6.9 AUC and Accurracy

We performed testing AUC and accuray for this model:

```
#==============================   TESTING DATA =================================
fitted3 <- predict(logit3,test3,type='response')
fitted3 <- ifelse(fitted3 > 0.5,1,0)

misClasificError3 <- mean(fitted3 != test3$TARGET_VALUE)
```

```
> Accuracy_logit3 = 1-misClasificError3
> print(paste('Accuracy', round(Accuracy_logit3,2)))
[1] "Accuracy 0.86"

require(ROCR)
p3 <- predict(logit3, test3, type="response")
pr3 <- prediction(p3, test3$TARGET_VALUE)
prf3 <- performance(pr3, measure = "tpr", x.measure = "fpr")
plot(prf3,main = "ROC Curve logit3")
#plot(prf3,ylab = "Sensitivity", xlab = "Specificity",
#     main = "ROC Curve logit3")

> predicted_3_factor <- factor(fitted3, levels=levels(test1$TARGET_VALUE))
>
> confusionMatrix(predicted_3_factor,test1$TARGET_VALUE)
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 100587  16052
         1      1      1

               Accuracy : 0.8624
                 95% CI : (0.8604, 0.8643)
    No Information Rate : 0.8624
    P-Value [Acc > NIR] : 0.5021

                  Kappa : 1e-04

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.000e+00
            Specificity : 6.229e-05
         Pos Pred Value : 8.624e-01
         Neg Pred Value : 5.000e-01
             Prevalence : 8.624e-01
         Detection Rate : 8.624e-01
   Detection Prevalence : 1.000e+00
      Balanced Accuracy : 5.000e-01
```

```
       'Positive' Class : 0

> auc3 <- performance(pr3, measure = "auc")
> auc3 <- auc3@y.values[[1]]
> print(paste('AUC logit3', round(auc3,2)))
[1] "AUC logit3 0.66"
```

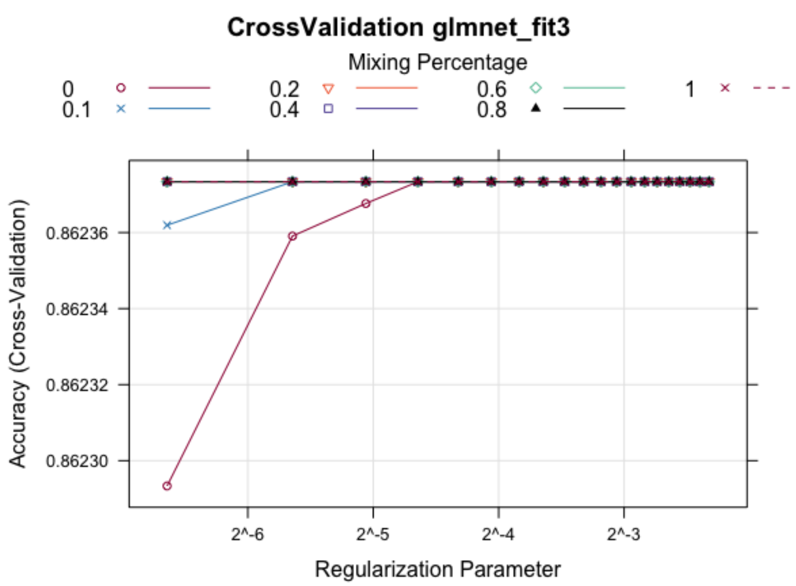
This model has an AUC of 0.66 which is an improvement compared to the model 1 (logit1) and 2 (logit2)

6.10 Cross-Validation for third model

```
#==============================   CROSS VALIDATION II   ===========================
HDMA_select_imputed$TARGET_VALUE =  as.factor(HDMA_select_imputed$TARGET_VALUE)

#Check the data for missing values.
sapply(HDMA_select_imputed, function(x) sum(is.na(x)))

#drop a column with missing data

str(HDMA_select_imputed)



not_features <- c("number_of_owner_occupied_units",
                    "property_type_name","hoepa_status_name",
                    "applicant_sex_name","applicant_race_name_1",
                    "applicant_ethnicity_name")




seed <- 250
classes2 <- HDMA_select_imputed[, "TARGET_VALUE"]
str(classes2)
predictors2 <- HDMA_select_imputed[, -match(c("TARGET_LABEL", "TARGET_VALUE", not_features),
                 colnames(HDMA_select_imputed))]
predictors2
require(caret)
train_set2 <- createDataPartition(classes2, p = 0.75, list = FALSE)
str(train_set2)
```

```r
train_predictors2 <- predictors2[train_set2, ]
train_classes2 <- classes2[train_set2]
test_predictors2 <- predictors2[-train_set2, ]
test_classes2 <- classes2[-train_set2]

set.seed(seed)
cv_splits2 <- createFolds(classes2, k = 10, returnTrain = TRUE)
str(cv_splits2)

require(glmnet)

set.seed(seed)

HDMA_imputed_train <- HDMA_select_imputed[train_set2, ]
HDMA_imputed_test <- HDMA_select_imputed[-train_set2, ]

glmnet_grid2 <- expand.grid(alpha = c(0,  .1,  .2, .4, .6, .8, 1),
                           lambda = seq(.01, .2, length = 20))
glmnet_ctrl2 <- trainControl(method = "cv", number = 10)
glmnet_fit2 <- train(TARGET_VALUE ~ ., data = HDMA_imputed_train,
                    method = "glmnet",
                    preProcess = c("center", "scale"),
                    tuneGrid = glmnet_grid2,
                    trControl = glmnet_ctrl2)

> glmnet_fit2
glmnet

349925 samples
    15 predictor
     2 classes: '0', '1'

Pre-processing: centered (28), scaled (28)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 314932, 314933, 314932, 314932, 314934, 314933, ...
Resampling results across tuning parameters:

  alpha  lambda  Accuracy   Kappa
  0.0    0.01    0.8622933  8.069641e-05
  0.0    0.02    0.8623591  3.160487e-05
```

```
  0.0    0.03    0.8623677  -1.142946e-05
  0.0    0.04    0.8623734   0.000000e+00
  0.0    0.05    0.8623734   0.000000e+00
  0.0    0.06    0.8623734   0.000000e+00
  0.0    0.07    0.8623734   0.000000e+00

  1.0    0.17    0.8623734   0.000000e+00
  1.0    0.18    0.8623734   0.000000e+00
  1.0    0.19    0.8623734   0.000000e+00
  1.0    0.20    0.8623734   0.000000e+00

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were alpha = 0 and lambda = 0.2.



trellis.par.set(caretTheme())
plot(glmnet_fit2, scales = list(x = list(log = 2)) ,
     main = "CrossValidation glmnet_fit3")
```



The optimal tuning parameter values are $\alpha = 0$ and $\lambda = 0.2$.

Again, it is possible to predict new samples with the identified optimal model using the predict method:

```
> #================ TESTING
> pred_classes2 <- predict(glmnet_fit2, newdata = HDMA_imputed_test)
> table(pred_classes2)
pred_classes2
     0      1
```

```
116641      0



> pred_probs2 <- predict(glmnet_fit2, newdata = HDMA_imputed_test, type = "prob")
> head(pred_probs2)
          0         1
1 0.8328016 0.1671984
2 0.8522192 0.1477808
3 0.8808983 0.1191017
4 0.8093916 0.1906084
5 0.8920562 0.1079438
6 0.8938589 0.1061411
```

7 Conclusion

We have applied a supervised machine learning method, the logistic regression to the HDMA dataset. After organizing, cleaning and processing the data, we tried to fit the first logistic regression model. The model provided reasonable accuracy. However, p-values indicate some variables could be excluded. After that, the second model attempt was built with insignificant accuracy gain. The last model (third model, logit 3) was built using categorical/factor and numerical attributes. An AUC value of 0.66 was obtained for logit3, and this has been improved compared to the model 1 (logit1) and 2 (logit2). Therefore, logit3 is our choice to present to the business.

8 References

Confusion Matrix, https://en.wikipedia.org/wiki/Confusion_matrix
Create Awesome HTML Table with knitr::kable and kableExtr, https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html
ROC and AUC, Clearly Explained!, https://www.youtube.com/watch?v=4jRBRDbJemM