Credit Card Application Approvals using R

April 24, 2019

0.1 Project: Credit Card Application Approvals using R

This notebook contains a credit card approval predictor for commercial banks using machine learning techniques.

0.1.1 Dataset

For this project, the dataset was extracted from the UCI Machine Learning Repository Credit Card Approval dataset

The dataset contains data for 690 customers that applied for credit with a retail bank. There are 16 attributes captured for each customer; including a decision flag which allows you to identify those customers which were approved and denied for credit.

0.1.2 Summary

The analysis of this project consist on the creation of a model to evaluate the decision to approve or deny credit card applications. The final model created is a logarithmic regression model. This model was able to predict the outcome of a credit applications with 84% accuracy which was significantly better performance than the baseline model.

As a conclusion, there are four drivers that possitively affect the approval decision, as these factors increase, so does the probability that a credit card will be issued.

Applications can get rejected for many reasons, including, like high loan balances, low income levels, or too many inquiries on an individual's credit report, among others. The four influencing factors are:

Prior default, Years employed, Credit score, and Income level. Other variables such as age, sex, or ethnicity did not have an influence on whether the application was denied. A Chi Squared test for independence validated our conclusion Ethnicity and Approval status are independent.

0.1.3 Notebook' structure

The structure of this notebook is as follows:

First, loading and viewing the dataset.

Second, preprocessing the dataset to ensure the machine learning model we choose can make good predictions.

Third, doing some exploratory data analysis to build our intuitions.

Finally, we will build a machine learning model using Logistic Regression that can predict if an individual's application for a credit card will be accepted.

0.2 1. Loading and viewing the dataset

All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. This dataset is interesting because there is a good mix of attributes – continuous, nominal with small numbers of values, and nominal with larger numbers of values.
There are also a few missing values.

- Number of Instances: 690
- Number of Attributes: 15 + class attribute
- Class Distribution +: 307 (44.5%) -: 383 (55.5%)

```
In [1]: # Setup the environment
        # package.list<- c("knitr","ggplot2","dplyr","reshape2","ROCR","caTools","rpart",
        #                   "rpart.plot","arules","scales")
        # lapply(package.list, require, character.only = TRUE)
        # options(scipen=6, width=100)

        library(ggplot2)
        #install.packages('caTools')
        library(caTools)  #Used for data splitting

        # Load dataset

        credit_df = read.csv("/Users/jay/Downloads/Predicting Credit Card Approvals/datasets/c

        # Inspect data
        tail(credit_df,17)
```

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed
674	?	29.5	2.000	y	p	e	h	2.000
675	a	37.33	2.500	u	g	i	h	0.210
676	a	41.58	1.040	u	g	aa	v	0.665
677	a	30.58	10.665	u	g	q	h	0.085
678	b	19.42	7.250	u	g	m	v	0.040
679	a	17.92	10.210	u	g	ff	ff	0.000
680	a	20.08	1.250	u	g	c	v	0.000
681	b	19.5	0.290	u	g	k	v	0.290
682	b	27.83	1.000	y	p	d	h	3.000
683	b	17.08	3.290	u	g	i	v	0.335
684	b	36.42	0.750	y	p	d	v	0.585
685	b	40.58	3.290	u	g	m	v	3.500
686	b	21.08	10.085	y	p	e	h	1.250
687	a	22.67	0.750	u	g	c	v	2.000
688	a	25.25	13.500	y	p	ff	ff	2.000
689	b	17.92	0.205	u	g	aa	v	0.040
690	b	35	3.375	u	g	c	h	8.290

1 2. Exploratory Data Analysis

Inspecting the structure, numerical summary, and specific rows of the dataset. - the dataset has a mixture of numerical and non-numerical features. This can be fixed with some preprocessing.

- Specifically, the features 2, 7, 10 and 14 contain numeric values (of types float64, float64, int64 and int64 respectively) and all the other features contain non-numeric values.
- The dataset also contains values from several ranges. Some features have a value range of 0 - 28, some have a range of 2 - 67, and some have a range of 1017 - 100000.
- We can get useful statistical information (like mean, max, and min) about the features that have numerical values.

```
In [2]: # Print summary statistics
        credit_df_description = summary(credit_df)
        print(credit_df_description)

        # Print DataFrame information (View the structure of the data)
        str(credit_df)
```

```
 Gender       Age            Debt          Married  BankCustomer EducationLevel
 ?: 12   ?       : 12   Min.   : 0.000   ?:  6    ? :  6       c      :137
 a:210   22.67   :  9   1st Qu.: 1.000   l:  2    g :519       q      : 78
 b:468   20.42   :  7   Median : 2.750   u:519    gg:  2       w      : 64
         18.83   :  6   Mean   : 4.759   y:163    p :163       i      : 59
         19.17   :  6   3rd Qu.: 7.207                         aa     : 54
         20.67   :  6   Max.   :28.000                         ff     : 53
         (Other):644                                           (Other):245
   Ethnicity    YearsEmployed    PriorDefault Employed  CreditScore
 v      :399   Min.   : 0.000   f:329        f:395    Min.   : 0.0
 h      :138   1st Qu.: 0.165   t:361        t:295    1st Qu.: 0.0
 bb     : 59   Median : 1.000                         Median : 0.0
 ff     : 57   Mean   : 2.223                         Mean   : 2.4
 ?      :  9   3rd Qu.: 2.625                         3rd Qu.: 3.0
 j      :  8   Max.   :28.500                         Max.   :67.0
 (Other): 20
 DriversLicense Citizen    ZipCode        Income        ApprovalStatus
 f:374          g:625   0      :132   Min.   :     0.0   -:383
 t:316          p:  8   120    : 35   1st Qu.:     0.0   +:307
                s: 57   200    : 35   Median :     5.0
                        160    : 34   Mean   :  1017.4
                        100    : 30   3rd Qu.:   395.5
                        80     : 30   Max.   :100000.0
                        (Other):394
'data.frame':        690 obs. of  16 variables:
 $ Gender        : Factor w/ 3 levels "?","a","b": 3 2 2 3 3 3 3 2 3 3 ...
 $ Age           : Factor w/ 350 levels "?","13.75","15.17",..: 158 330 91 127 45 170 181 76 3
 $ Debt          : num  0 4.46 0.5 1.54 5.62 ...
 $ Married       : Factor w/ 4 levels "?","l","u","y": 3 3 3 3 3 3 3 3 4 4 ...
 $ BankCustomer  : Factor w/ 4 levels "?","g","gg","p": 2 2 2 2 2 2 2 2 4 4 ...
 $ EducationLevel: Factor w/ 15 levels "?","aa","c","cc",..: 14 12 12 14 14 11 13 4 10 14 ...
 $ Ethnicity     : Factor w/ 10 levels "?","bb","dd",..: 9 5 5 9 9 9 5 9 5 9 ...
 $ YearsEmployed : num  1.25 3.04 1.5 3.75 1.71 ...
 $ PriorDefault  : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...
 $ Employed      : Factor w/ 2 levels "f","t": 2 2 1 2 1 1 1 1 1 1 ...
 $ CreditScore   : int  1 6 0 5 0 0 0 0 0 0 ...
 $ DriversLicense: Factor w/ 2 levels "f","t": 1 1 1 2 1 2 2 1 1 2 ...
 $ Citizen       : Factor w/ 3 levels "g","p","s": 1 1 1 1 3 1 1 1 1 1 ...
 $ ZipCode       : Factor w/ 171 levels "?","0","100",..: 44 120 76 3 10 98 27 161 36 142 ...
 $ Income        : int  0 560 824 3 0 0 31285 1349 314 1442 ...
 $ ApprovalStatus: Factor w/ 2 levels "-","+": 2 2 2 2 2 2 2 2 2 2 ...
```

1.1 3. Handling missing values (Marking missing values as NaN)

Marking Missing Values or corrupted data as NaN. Then, we can count the number of true values in each column.

- The dataset has missing values. The missing values in the dataset are labeled with '?'.
- Let's temporarily replace these missing value question marks with NaN.
- A total of 67 missing values were identified

In [3]:
```
# Inspect missing values in the dataset
print(tail(credit_df))

# Count the number of NaNs in each column
colSums(is.na(credit_df))

#count the number of NaNS in dataframe
sum(is.na(credit_df))

# checking an example of '?' value
credit_df[674,]

# Replace the '?'s with NaN
credit_df[ credit_df == "?" ] <- NA
#Updating the levels of the factor variables
credit_df[,-c(2:3,8,11,15)] <- lapply( credit_df[,-c(2:3,8,11,15)], factor )

# Count the number of NaNs in each column
colSums(is.na(credit_df))

#count the number of NaNS in dataframe
sum(is.na(credit_df))

# verify transformation
credit_df[674,]
```

```
    Gender    Age    Debt Married BankCustomer EducationLevel Ethnicity
685      b 40.58   3.290       u            g              m         v
686      b 21.08  10.085       y            p              e         h
687      a 22.67   0.750       u            g              c         v
688      a 25.25  13.500       y            p             ff        ff
689      b 17.92   0.205       u            g             aa         v
690      b     35   3.375       u            g              c         h
    YearsEmployed PriorDefault Employed CreditScore DriversLicense Citizen
685          3.50            f        f           0              t       s
686          1.25            f        f           0              f       g
687          2.00            f        t           2              t       g
688          2.00            f        t           1              t       g
689          0.04            f        f           0              f       g
690          8.29            f        f           0              t       g
    ZipCode Income ApprovalStatus
685     400      0              -
686     260      0              -
687     200    394              -
688     200      1              -
689     280    750              -
690       0      0              -
```

Gender 0 **Age** 0 **Debt** 0 **Married** 0 **BankCustomer** 0 **EducationLevel** 0 **Ethnicity** 0 **YearsEmployed** 0 **PriorDefault** 0 **Employed** 0 **CreditScore** 0 **DriversLicense** 0 **Citizen** 0 **ZipCode** 0 **Income** 0 **ApprovalStatus** 0

0

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	P
674	?	29.5	2	y	p	e	h	2	f

Gender 12 **Age** 12 **Debt** 0 **Married** 6 **BankCustomer** 6 **EducationLevel** 9 **Ethnicity** 9 **YearsEmployed** 0 **PriorDefault** 0 **Employed** 0 **CreditScore** 0 **DriversLicense** 0 **Citizen** 0 **ZipCode** 13 **Income** 0 **ApprovalStatus** 0

67

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	P
674	NA	29.5	2	y	p	e	h	2	f

1.2 5. Handling the missing values (Data Imputation)

Median Imputation for numerical data and Frequent value for categorical data.

- There are missing values for numerical variables. We could simply use the mean of all the existing values to do so. Another method would be to check the relationship among the numeric values and use a linear regression to fill them in.
- There are still some missing values to be imputed for columns Gender, Age, Married, BankCustomer, EducationLevel, Ethnicity and ZipCode. All of these columns contain non-numeric data and we are going to impute these missing values with the most frequent values as present in the respective columns.

```
In [4]: ## Imputation numerical variables

        # Transforming Age into numerical value
        credit_df$Age<-as.numeric(credit_df$Age)

        ## Imputation Age
        mean_age<- mean(credit_df$Age,na.rm=T)

        # Use correlation among numerical variables to predict missing age values
        Numeric       <- credit_df[,c(2:3,8,11,15)]
        colnames(Numeric)

        round(cor(Numeric,use="complete.obs"),3)
        #  The largest value in the first row is 0.395 meaning age is most closely correlated 

        age_imputate<-lm(Age~YearsEmployed, data=credit_df, na.action=na.exclude)
        age_imputate$coefficients
        age_missing_index<-which(is.na(credit_df$Age))
        credit_df$Age[age_missing_index]<- predict(age_imputate,newdata=credit_df[age_missing_
```

1. 'Age' 2. 'Debt' 3. 'YearsEmployed' 4. 'CreditScore' 5. 'Income'

	Age	Debt	YearsEmployed	CreditScore	Income
Age	1.000	0.149	0.395	0.183	0.019
Debt	0.149	1.000	0.301	0.272	0.122
YearsEmployed	0.395	0.301	1.000	0.327	0.053
CreditScore	0.183	0.272	0.327	1.000	0.063
Income	0.019	0.122	0.053	0.063	1.000

(Intercept)	124.693475933034	**YearsEmployed**	11.0096259767489

```
In [5]: colSums(is.na(credit_df))
```

Gender	12	**Age**	0	**Debt**	0	**Married**	6
BankCustomer	6	**EducationLevel**	9	**Ethnicity**	9	**YearsEmployed**	0
PriorDefault	0	**Employed**	0	**CreditScore**	0	**DriversLicense**	0
Citizen	0	**ZipCode**	13	**Income**	0	**ApprovalStatus**	0

```
In [6]: ## Imputation Categorical variables

        # tracking value before imputation
        credit_df[674,]

        # Generating a mode function were the input is called data and it is in the form df$co
        mode <- function(data){
            val <- unique(data[!is.na(data)])
            output <- val[which.max(tabulate(match(data, val)))]
            return(output)
        }


        # Using the apply function to run the mode function accross the columns in the datafra
        categorical_col <- c(1,4:7,9:10,12:14)
        credit_df[categorical_col]<- lapply(credit_df[categorical_col],function(x) { x[is.na(x)


        #credit_df$Gender[is.na(credit_df$Gender)] <- mode(credit_df$Gender)
        #df1[,subset1] <- as.data.frame(lapply(df1[,subset1],function(x) { x[is.na(x)] <- 0; x

        # tracking value after imputation
        credit_df[674,]

        colSums(is.na(credit_df))
```

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	P
674	NA	142	2	y	p	e	h	2	f

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	P
674	b	142	2	y	p	e	h	2	f

Gender 0 **Age** 0 **Debt** 0 **Married** 0 **BankCustomer** 0 **EducationLevel** 0 **Ethnicity** 0 **YearsEmployed** 0 **PriorDefault** 0 **Employed** 0 **CreditScore** 0 **DriversLicense** 0 **Citizen** 0 **ZipCode** 0 **Income** 0 **ApprovalStatus** 0

1.3 6. Preprocessing the data (Label Encoding)

We will be converting all the non-numeric values into numeric ones using label encoding.

```
In [7]: # Convert binary values to 1 or 0
        credit_df$Gender <- factor(ifelse(credit_df$Gender=="a",1,0))
        credit_df$Employed <- factor(ifelse(credit_df$Employed=="t",1,0))
        credit_df$PriorDefault<- factor(ifelse(credit_df$PriorDefault=="t",1,0))
        credit_df$ApprovalStatus <- factor(ifelse(credit_df$ApprovalStatus=="+",1,0))

        str(credit_df)
```

```
'data.frame':        690 obs. of  16 variables:
 $ Gender        : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 1 ...
 $ Age           : num  158 330 91 127 45 170 181 76 312 257 ...
 $ Debt          : num  0 4.46 0.5 1.54 5.62 ...
 $ Married       : Factor w/ 3 levels "l","u","y": 2 2 2 2 2 2 2 2 3 3 ...
 $ BankCustomer  : Factor w/ 3 levels "g","gg","p": 1 1 1 1 1 1 1 1 3 3 ...
 $ EducationLevel: Factor w/ 14 levels "aa","c","cc",..: 13 11 11 13 13 10 12 3 9 13 ...
 $ Ethnicity     : Factor w/ 9 levels "bb","dd","ff",..: 8 4 4 8 8 8 4 8 4 8 ...
 $ YearsEmployed : num  1.25 3.04 1.5 3.75 1.71 ...
 $ PriorDefault  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ Employed      : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 1 1 1 1 ...
 $ CreditScore   : int  1 6 0 5 0 0 0 0 0 0 ...
 $ DriversLicense: Factor w/ 2 levels "f","t": 1 1 1 2 1 2 2 1 1 2 ...
 $ Citizen       : Factor w/ 3 levels "g","p","s": 1 1 1 1 3 1 1 1 1 1 ...
 $ ZipCode       : Factor w/ 170 levels "0","100","102",..: 43 119 75 2 9 97 26 160 35 141 ...
 $ Income        : int  0 560 824 3 0 0 31285 1349 314 1442 ...
 $ ApprovalStatus: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

1.4 7. Splitting the dataset into train and test sets and Feature selection

Now, we will split our data into train and test sets. Ideally, no information from the test data should be used to scale the training data or should be used to direct the training process of a machine learning model. Hence, we first split the data and then apply the scaling.

Also, features like DriversLicense and ZipCode are not as important as the other features in the dataset for predicting credit card approvals. We should drop them to design our machine learning model with the best set of features.

```
In [8]: # Drop the features 'DriversLicense', 'ZipCode'
        credit_df = credit_df[-c(12,14)]

        # Split data in training a testing datasets
        library(caTools)
        set.seed(123)
        split<- sample.split(credit_df$ApprovalStatus, SplitRatio=0.75)
        Train<- subset(credit_df,split==TRUE)
        Test <- subset(credit_df, split==FALSE)

        # New dataframes shape
        dim(Train)
        dim(Test)
```

1. 517 2. 14
1. 173 2. 14

```
In [9]: # Get success rates in training set
        table(Train$ApprovalStatus)
```

```
  0   1
287 230
```

1.5 8. Preprocessing the data (Rescaling Data to an uniform range)

We are only left with one final preprocessing step of scaling data between 0-1 before we can fit a machine learning model to the data.

For example, the credit score, CreditScore, of a person is their creditworthiness based on their credit history. The higher this number, the more financially trustworthy a person is considered to be. So, a CreditScore of 1 is the highest since we're rescaling all the values to the range of 0-1.

```
In [10]: # We dont need to rescale for R
```

1.6 9. Fitting a logistic regression model to the train set

Essentially, predicting if a credit card application will be approved or not is a classification task. According to UCI, our dataset contains more instances that correspond to "Denied" status than instances corresponding to "Approved" status. Specifically, out of 690 instances, there are 383 (55.5%) applications that got denied and 307 (44.5%) applications that got approved.

This gives us a benchmark. A good machine learning model should be able to accurately predict the status of the applications with respect to these statistics.

Which model should we pick? A question to ask is: are the features that affect the credit card approval decision process correlated with each other? they indeed are correlated. Because of this correlation, we'll take advantage of the fact that generalized linear models perform well in these cases. Let's start our machine learning modeling with a Logistic Regression model (a generalized linear model).

```
In [11]: # Create logrithmic regresion model = base model
         logreg<- glm(ApprovalStatus~., data=Train,family=binomial)

         summary(logreg)
```

```
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Call:
glm(formula = ApprovalStatus ~ ., family = binomial, data = Train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4169  -0.2962  -0.1246   0.4502   3.0000

Coefficients: (2 not defined because of singularities)
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.140e+01  1.455e+03   0.008 0.993750
Gender1           2.079e-01  3.700e-01   0.562 0.574188
Age               7.729e-05  1.943e-03   0.040 0.968271
Debt             -1.825e-02  3.354e-02  -0.544 0.586474
Marriedu         -1.550e+01  1.455e+03  -0.011 0.991504
Marriedy         -1.617e+01  1.455e+03  -0.011 0.991136
BankCustomergg           NA         NA      NA       NA
BankCustomerp            NA         NA      NA       NA
EducationLevelc   2.142e-01  5.904e-01   0.363 0.716725
EducationLevelcc  1.281e+00  8.825e-01   1.452 0.146566
EducationLeveld   2.205e-01  8.935e-01   0.247 0.805092
EducationLevele   1.817e+00  1.296e+00   1.402 0.160867
EducationLevelff -3.940e+00  2.298e+00  -1.715 0.086398 .
EducationLeveli  -3.819e-01  8.318e-01  -0.459 0.646130
EducationLevelj  -3.775e+00  2.374e+00  -1.590 0.111871
EducationLevelk  -3.282e-01  7.130e-01  -0.460 0.645333
EducationLevelm   4.999e-01  7.865e-01   0.636 0.525098
```

```
EducationLevelq   7.737e-01  6.435e-01   1.202 0.229245
EducationLevelr  -9.230e+00  1.679e+02  -0.055 0.956146
EducationLevelw   6.325e-01  6.593e-01   0.959 0.337384
EducationLevelx   1.043e+00  8.831e-01   1.182 0.237368
Ethnicitydd      -7.459e-01  1.941e+00  -0.384 0.700799
Ethnicityff       2.542e+00  2.166e+00   1.173 0.240623
Ethnicityh        4.233e-01  6.824e-01   0.620 0.535012
Ethnicityj        5.395e+00  2.258e+00   2.390 0.016870 *
Ethnicityn        3.040e+00  1.634e+00   1.860 0.062920 .
Ethnicityo       -5.754e+01  2.058e+03  -0.028 0.977698
Ethnicityv        4.043e-01  6.283e-01   0.643 0.519966
Ethnicityz       -2.662e+00  1.936e+00  -1.375 0.169080
YearsEmployed     3.642e-02  5.351e-02   0.681 0.496106
PriorDefault1     4.010e+00  4.343e-01   9.233  < 2e-16 ***
Employed1         4.520e-01  4.305e-01   1.050 0.293690
CreditScore       1.418e-01  7.260e-02   1.952 0.050887 .
Citizenp          3.393e+00  9.893e-01   3.429 0.000606 ***
Citizens         -2.246e-01  5.440e-01  -0.413 0.679734
Income            6.265e-04  2.101e-04   2.982 0.002864 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 710.42  on 516  degrees of freedom
Residual deviance: 297.45  on 483  degrees of freedom
AIC: 365.45

Number of Fisher Scoring iterations: 14
```

1.7 10. Making predictions and evaluating performance

But how well does our model perform?

We will now evaluate our model on the test set with respect to classification accuracy. But we will also take a look the model's confusion matrix. In the case of predicting credit card applications, it is equally important to see if our machine learning model is able to predict the approval status of the applications as denied that originally got denied. If our model is not performing well in this aspect, then it might end up approving the application that should have been approved. The confusion matrix helps us to view our model's performance from these aspects.

- Our model was pretty good! It was able to yield an accuracy score of almost 84%.
- For the confusion matrix, the first element of the of the first row of the confusion matrix denotes the true negatives meaning the number of negative instances (denied applications) predicted by the model correctly. And the last element of the second row of the confusion matrix denotes the true positives meaning the number of positive instances (approved applications) predicted by the model correctly.

```
In [12]: # Apply the model to the test set
         logreg_predict<-predict(logreg, newdata=Test,type="response")


         # Create a confusion Matrix
         table(Test$ApprovalStatus,logreg_predict>0.5)
```

```
Warning message in predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type == :
prediction from a rank-deficient fit may be misleading
```

```
    FALSE TRUE
  0    81   15
  1     9   68
```

1.8 11. Stepwise and making the model perform better

Let's see if we can do better. We can perform a stepwise selection of the model parameters to improve the model's ability to predict credit card approvals.

```
In [13]: # Use the step function to simplify the model with a function
         backwards = step(logreg) # Backwards selection is the default
```

```
Start:  AIC=365.45
ApprovalStatus ~ Gender + Age + Debt + Married + BankCustomer +
    EducationLevel + Ethnicity + YearsEmployed + PriorDefault +
    Employed + CreditScore + Citizen + Income



Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Step:  AIC=365.45
ApprovalStatus ~ Gender + Age + Debt + Married + EducationLevel +
    Ethnicity + YearsEmployed + PriorDefault + Employed + CreditScore +
    Citizen + Income


Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred

                 Df Deviance    AIC
- EducationLevel 13   309.90 351.90
- Ethnicity       8   307.97 359.97
- Age             1   297.45 363.45
- Debt            1   297.74 363.74
- Gender          1   297.76 363.76
- YearsEmployed   1   297.92 363.92
- Employed        1   298.54 364.54
<none>                297.45 365.45
- Married         2   301.54 365.54
- CreditScore     1   302.34 368.34
- Citizen         2   308.14 372.14
- Income          1   313.76 379.76
- PriorDefault    1   440.49 506.49


Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Step:  AIC=351.9
ApprovalStatus ~ Gender + Age + Debt + Married + Ethnicity +
    YearsEmployed + PriorDefault + Employed + CreditScore + Citizen +
    Income



Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
```

```
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred

                Df Deviance    AIC
- Gender         1   309.90 349.90
- Age            1   309.91 349.91
- Debt           1   310.31 350.31
- YearsEmployed  1   310.84 350.84
- Ethnicity      8   325.60 351.60
<none>               309.90 351.90
- Employed       1   312.33 352.33
- CreditScore    1   314.67 354.67
- Citizen        2   320.67 358.67
- Married        2   322.36 360.36
- Income         1   326.96 366.96
- PriorDefault   1   467.89 507.89


Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Step:  AIC=349.9
ApprovalStatus ~ Age + Debt + Married + Ethnicity + YearsEmployed +
    PriorDefault + Employed + CreditScore + Citizen + Income



Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred

                Df Deviance    AIC
- Age            1   309.92 347.92
- Debt           1   310.31 348.31
- YearsEmployed  1   310.84 348.84
- Ethnicity      8   325.60 349.60
<none>               309.90 349.90
- Employed       1   312.35 350.35
- CreditScore    1   314.67 352.67
- Citizen        2   320.69 356.69
- Married        2   322.53 358.53
- Income         1   326.96 364.96
- PriorDefault   1   467.89 505.89


Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Step:  AIC=347.92
ApprovalStatus ~ Debt + Married + Ethnicity + YearsEmployed +
    PriorDefault + Employed + CreditScore + Citizen + Income



Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred

                Df Deviance    AIC
- Debt           1   310.32 346.32
- YearsEmployed  1   310.87 346.87
<none>               309.92 347.92
- Employed       1   312.41 348.41
- Ethnicity      8   326.55 348.55
- CreditScore    1   314.68 350.68
- Citizen        2   320.76 354.76
- Married        2   322.70 356.70
- Income         1   326.96 362.96
- PriorDefault   1   469.85 505.85


Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Step:  AIC=346.32
ApprovalStatus ~ Married + Ethnicity + YearsEmployed + PriorDefault +
    Employed + CreditScore + Citizen + Income



Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
                Df Deviance    AIC
- YearsEmployed  1   311.18 345.18
<none>               310.32 346.32
- Employed       1   312.96 346.96
- Ethnicity      8   327.59 347.59
- CreditScore    1   314.84 348.84
- Citizen        2   321.54 353.54
- Married        2   323.22 355.22
- Income         1   327.36 361.36
- PriorDefault   1   472.15 506.15


Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred


Step:  AIC=345.18
ApprovalStatus ~ Married + Ethnicity + PriorDefault + Employed +
    CreditScore + Citizen + Income



Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurredWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred

               Df Deviance    AIC
<none>              311.18 345.18
- Employed      1   313.53 345.53
- Ethnicity     8   328.06 346.06
- CreditScore   1   316.60 348.60
- Citizen       2   322.24 352.24
- Married       2   325.10 355.10
- Income        1   327.60 359.60
- PriorDefault  1   494.31 526.31
```

```
In [14]: formula(backwards)

         summary(backwards)
```

```
ApprovalStatus ~ Married + Ethnicity + PriorDefault + Employed +
    CreditScore + Citizen + Income



Call:
```

```
glm(formula = ApprovalStatus ~ Married + Ethnicity + PriorDefault +
    Employed + CreditScore + Citizen + Income, family = binomial,
    data = Train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3000  -0.2910  -0.1414   0.5048   3.0910

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.685e+01  1.455e+03   0.012 0.990760
Marriedu      -2.093e+01  1.455e+03  -0.014 0.988527
Marriedy      -2.163e+01  1.455e+03  -0.015 0.988143
Ethnicitydd    9.049e-01  1.549e+00   0.584 0.559167
Ethnicityff   -1.293e+00  8.973e-01  -1.441 0.149453
Ethnicityh     9.698e-01  5.736e-01   1.691 0.090898 .
Ethnicityj     2.784e+00  1.241e+00   2.244 0.024843 *
Ethnicityn     3.097e+00  1.443e+00   2.146 0.031846 *
Ethnicityo    -6.358e+01  2.058e+03  -0.031 0.975360
Ethnicityv     7.346e-01  5.025e-01   1.462 0.143759
Ethnicityz    -7.631e-01  1.347e+00  -0.566 0.571091
PriorDefault1  4.034e+00  4.011e-01  10.058  < 2e-16 ***
Employed1      6.206e-01  4.038e-01   1.537 0.124327
CreditScore    1.338e-01  6.745e-02   1.984 0.047242 *
Citizenp       3.339e+00  9.107e-01   3.666 0.000246 ***
Citizens       4.880e-03  5.053e-01   0.010 0.992295
Income         5.895e-04  1.989e-04   2.963 0.003045 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 710.42  on 516  degrees of freedom
Residual deviance: 311.18  on 500  degrees of freedom
AIC: 345.18

Number of Fisher Scoring iterations: 14
```

```
In [15]: # Apply the model to the test set
         backwards_predict<-predict(backwards, newdata=Test,type="response")


         # Create a confusion Matrix
         table(Test$ApprovalStatus,backwards_predict>0.5)
```

```
   FALSE TRUE
 0    79   17
 1    10   67
```

```
In [ ]:
```