Classification

Umaid Ahmed

2/18/2023

source:https://www.kaggle.com/datasets/whenamancodes/credit-card-customers-prediction (https://www.kaggle.com/datasets/whenamancodes/credit-card-customers-prediction)

In linear regression the target is quantitative while in logistical regression target is qualitative. In logistic regression the target is divided in to different classification. These classification can be binomial or it can be multiclass. In logistic regression the observations are separated into regions with each region having a same class.

Reading The Data

```
df <- read.csv("Bank Churners.csv")
```

Dividing Data into Training and Testing

```
set.seed(1234)
i <- sample(1:nrow(df), 0.80*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

First lets take a look at the structure of the data

```
str(train)
```

```
## 'data.frame':    8101 obs. of  23 variables:
##  $ CLIENTNUM
: int  758178558 721020858 809378133 708323208 751607058 717580458 711628458 708162558 718174
908 719273733 ...
##  $ Attrition_Flag
: chr  "Existing Customer" "Existing Customer" "Existing Customer" "Attrited Customer" ...
##  $ Customer_Age
: int  44 34 43 49 45 43 45 64 53 37 ...
##  $ Gender
: chr  "F" "F" "F" "F" ...
##  $ Dependent_count
: int  3 1 3 3 2 4 3 0 3 2 ...
##  $ Education_Level
: chr  "Graduate" "College" "Uneducated" "Uneducated" ...
##  $ Marital_Status
: chr  "Single" "Single" "Single" "Married" ...
##  $ Income_Category
: chr  "Unknown" "Unknown" "Less than $40K" "Less than $40K" ...
##  $ Card_Category
: chr  "Blue" "Blue" "Blue" "Blue" ...
##  $ Months_on_book
: int  37 20 38 36 38 36 31 54 36 36 ...
##  $ Total_Relationship_Count
: int  3 2 6 4 4 2 4 6 4 5 ...
##  $ Months_Inactive_12_mon
: int  2 2 1 2 3 3 1 1 3 3 ...
##  $ Contacts_Count_12_mon
: int  1 2 2 4 3 2 0 2 2 5 ...
##  $ Credit_Limit
: num  4703 3190 1521 1522 1686 ...
##  $ Total_Revolving_Bal
: int  1555 1959 0 0 0 0 901 1992 2417 998 ...
##  $ Avg_Open_To_Buy
: num  3148 1231 1521 1522 1686 ...
##  $ Total_Amt_Chng_Q4_Q1
: num  0.932 0.622 0.791 0.482 0.812 0.969 0.825 0.715 0.522 0.773 ...
##  $ Total_Trans_Amt
: int  4127 4220 4880 2241 4413 7847 1902 1355 1458 1855 ...
##  $ Total_Trans_Ct
: int  89 66 79 36 77 63 45 36 29 29 ...
##  $ Total_Ct_Chng_Q4_Q1
: num  0.648 0.61 0.549 0.8 0.75 ...
##  $ Avg_Utilization_Ratio
: num  0.331 0.614 0 0 0 0 0.102 0.613 0.572 0.125 ...
##  $ Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_cou
nt_Education_Level_Months_Inactive_12_mon_1: num  7.12e-05 1.04e-04 5.88e-05 9.97e-01 4.38e-0
4 ...
##  $ Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_cou
nt_Education_Level_Months_Inactive_12_mon_2: num  0.99993 0.9999 0.99994 0.00286 0.99956 ...
```

As it can be noted that there are 23 features in this data. Some of them are characters while others are numeric. For logistic regression we would be interested in the features that can be classified and predictors that can be good for the logistic regression.

Now, lets take a look some head values in the training dataframe and see if there are any features that can be used for logistic regression:

```
head(train)
```

	CLIENTNUM <int>	Attrition_Flag <chr>	Customer_Age <int>	Gender <chr>	Dependent_count <int>
7452	758178558	Existing Customer	44	F	3
8016	721020858	Existing Customer	34	F	1
7162	809378133	Existing Customer	43	F	3
8086	708323208	Attrited Customer	49	F	3
7269	751607058	Existing Customer	45	M	2
9196	717580458	Attrited Customer	43	M	4

6 rows | 1-6 of 24 columns

Now, lets take a look some tail values in the training dataframe and see if there are any features that can be used for logistic regression:

```
tail(train)
```

	CLIENTNUM <int>	Attrition_Flag <chr>	Customer_Age <int>	Gender <chr>	Dependent_count <int>
7408	715672458	Existing Customer	47	M	5
1230	809959683	Existing Customer	55	M	2
92	714070758	Existing Customer	49	M	4
9375	708770883	Existing Customer	43	M	2
2835	714011358	Existing Customer	38	M	3
6391	790164408	Existing Customer	59	F	1

6 rows | 1-6 of 24 columns

As we can see here there are columns that can be good predictor. For this dataset our target can be Income_Category and our predictors can be Customer_Age, Gender, Education_Level, Card_Category and Credit_Limit

since we are only interested in 6 columns we can reduce the dataset to:

```
train <- train[,c(3,4,6,8,9,14)]
test <- test[,c(3,4,6,8,9,14)]
str(train)
```

```
## 'data.frame':    8101 obs. of  6 variables:
##  $ Customer_Age   : int  44 34 43 49 45 43 45 64 53 37 ...
##  $ Gender         : chr  "F" "F" "F" "F" ...
##  $ Education_Level: chr  "Graduate" "College" "Uneducated" "Uneducated" ...
##  $ Income_Category: chr  "Unknown" "Unknown" "Less than $40K" "Less than $40K" ...
##  $ Card_Category  : chr  "Blue" "Blue" "Blue" "Blue" ...
##  $ Credit_Limit   : num  4703 3190 1521 1522 1686 ...
```

```
str(test)
```

```
## 'data.frame':    2026 obs. of  6 variables:
##  $ Customer_Age   : int  49 40 51 32 56 61 47 53 57 38 ...
##  $ Gender         : chr  "F" "F" "M" "M" ...
##  $ Education_Level: chr  "Graduate" "High School" "Unknown" "High School" ...
##  $ Income_Category: chr  "Less than $40K" "Less than $40K" "$120K +" "$60K - $80K" ...
##  $ Card_Category  : chr  "Blue" "Blue" "Gold" "Silver" ...
##  $ Credit_Limit   : num  8256 3313 34516 29081 11751 ...
```

storing some columns as factor so that we can perform logistic regression

(Note:The only reason I am reusing code for test train is because Professor specified in the Assignment details to perform data exploration on training data. If I am making changes to training data then I have to make changes to testing data as well)

```
train$Income_Category <- factor(train$Income_Category)
train$Gender <- factor(train$Gender)
train$Card_Category <- factor(train$Card_Category)
train$Education_Level <- factor(train$Education_Level)

test$Income_Category <- factor(test$Income_Category)
test$Gender <- factor(test$Gender)
test$Card_Category <- factor(test$Card_Category)
test$Education_Level <- factor(test$Education_Level)

str(train)
```

```
## 'data.frame':    8101 obs. of  6 variables:
##  $ Customer_Age   : int  44 34 43 49 45 43 45 64 53 37 ...
##  $ Gender         : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 1 1 1 2 ...
##  $ Education_Level: Factor w/ 7 levels "College","Doctorate",..: 3 1 6 6 2 6 3 6 6 6 ...
##  $ Income_Category: Factor w/ 6 levels "$120K +","$40K - $60K",..: 6 6 5 5 1 4 5 6 5 3 ...
##  $ Card_Category  : Factor w/ 4 levels "Blue","Gold",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Credit_Limit   : num  4703 3190 1521 1522 1686 ...
```

```
str(test)
```

```
## 'data.frame':    2026 obs. of  6 variables:
##  $ Customer_Age   : int  49 40 51 32 56 61 47 53 57 38 ...
##  $ Gender         : Factor w/ 2 levels "F","M": 1 1 2 2 2 2 2 2 1 1 ...
##  $ Education_Level: Factor w/ 7 levels "College","Doctorate",..: 3 4 7 4 1 4 2 6 3 3 ...
##  $ Income_Category: Factor w/ 6 levels "$120K +","$40K - $60K",..: 5 5 1 3 4 2 3 3 2 6 ...
##  $ Card_Category  : Factor w/ 4 levels "Blue","Gold",..: 1 1 2 4 1 1 1 1 1 1 ...
##  $ Credit_Limit   : num  8256 3313 34516 29081 11751 ...
```

learning the names of different levels in income category to have a better understanding of the target

```
levels(train$Income_Category)
```

```
## [1] "$120K +"        "$40K - $60K"    "$60K - $80K"    "$80K - $120K"
## [5] "Less than $40K" "Unknown"
```

Since we are only interested in the customer’s age and credit limit let’s take a look at the summary of those features:

```
summary(train[, c(1,6)])
```

```
##   Customer_Age    Credit_Limit
##  Min.   :26.00   Min.   : 1438
##  1st Qu.:41.00   1st Qu.: 2544
##  Median :46.00   Median : 4546
##  Mean   :46.31   Mean   : 8575
##  3rd Qu.:52.00   3rd Qu.:10979
##  Max.   :73.00   Max.   :34516
```

We can also use contrast to see how the target variable is classified

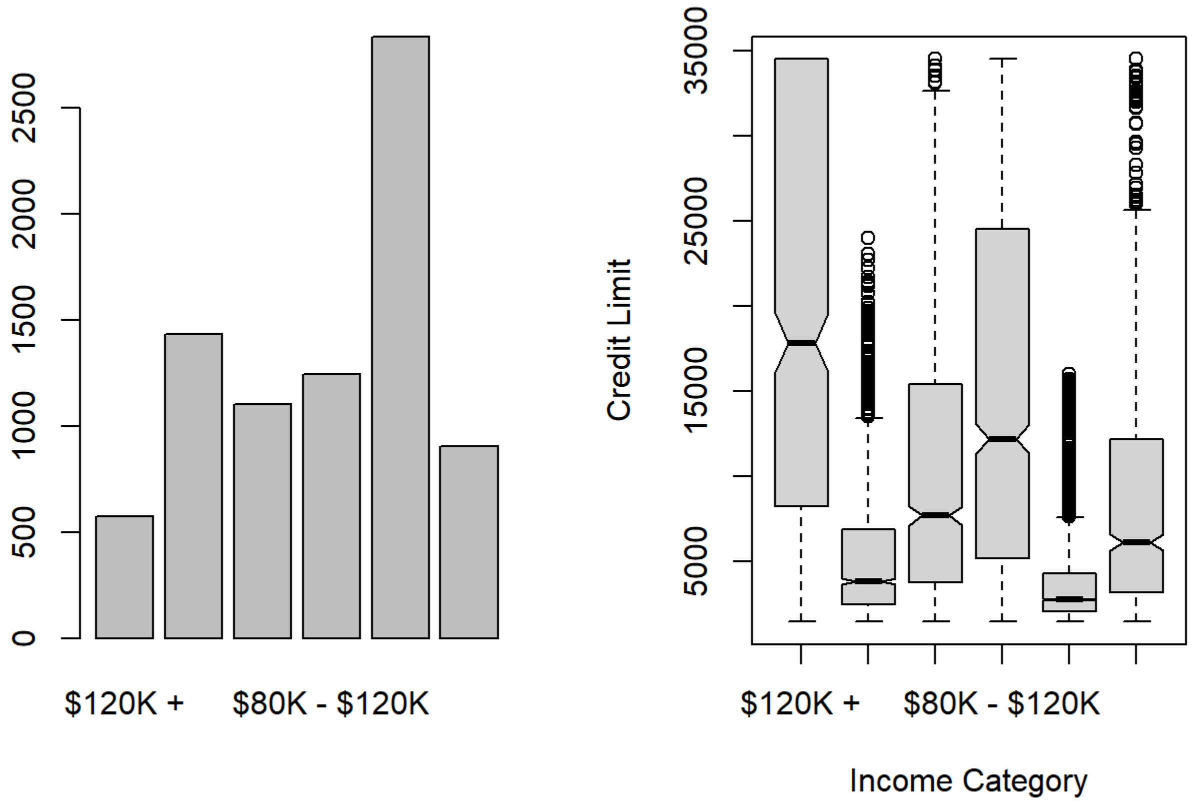
```
contrasts(train$Income_Category)
```

```
##                $40K - $60K $60K - $80K $80K - $120K Less than $40K Unknown
## $120K +                  0           0            0              0       0
## $40K - $60K              1           0            0              0       0
## $60K - $80K              0           1            0              0       0
## $80K - $120K             0           0            1              0       0
## Less than $40K           0           0            0              1       0
## Unknown                  0           0            0              0       1
```

The base case here is “120k +” while others are classified as there own string.

Informative Graph

```
par(mfrow=c(1,2))
plot(train$Income_Category)
plot(train$Credit_Limit~train$Income_Category, notch=TRUE, xlab="Income Category", ylab="Cred
it Limit")
```



The graph on the left is an histogram of Income_category. It shows that data is spread out evenly except that there are less people in the 120k+ category and there are more people in less than 40K. The graph on the right is a box plot. The box plot shows how well the data is distributed. Income categories are on the x axis and credit limit is on the y axis. The bar in middle shows the median value. As it can be seen that most of the categories have credit limit more than the median range. It can also be noted that some of the data point have a higher y-value, this can be due to large number of dataset.

Logistic Regression Model, Summary And Prediction

(Note: parts related to logistical regression are consolidated together in this section because there are multiple classification of target so to make a model for these multiple classification I used a function. The model inside the function was inaccessible outside the function)

First we would divide the data set into different categories. Dataset in a category of itself will have the value of 1 while other categories will have a value of 0

```
moreThan120k <- train
moreThan120k$Income_Category <- as.factor(ifelse (moreThan120k$Income_Category == "$120K +",
1,0))

between40to60k <- train
between40to60k$Income_Category <- as.factor(ifelse (between40to60k$Income_Category == "$40K -
$60K",1,0))

between60to80k <- train
between60to80k$Income_Category <- as.factor(ifelse (between60to80k$Income_Category == "$60K -
$80K",1,0))

between80to120k <- train
between80to120k$Income_Category <- as.factor(ifelse (between80to120k$Income_Category == "$80K
- $120K",1,0))

lessthan40k <- train
lessthan40k$Income_Category <- as.factor(ifelse (lessthan40k$Income_Category == "Less than $4
0K",1,0))
```

Creating a logistical regression function for different categories of data. This function will create a logistical regression model and outputs the summary of the regression. This function also used to predict on the test data. For our logistic regression model the target is the Income category while all the others features in the dataset are the predictors. Than the function makes prediction on the test data. if the probability is higher than 0.5 it is registered as a 1 and if it is less that 0.5 than 0. Next we take out the mean which will tell us how accurate our model was in predicting **(Note: part 4d and 4f of the assignment are consolidated together in this function because the regression mode was inaccessible outside the function)**

```
logistical_regression <- function(df, i){
    train <- df[i,]
    test <- df[-i,]
    glm1 <- glm(Income_Category~., data=train, family="binomial")
    print("===================printing Summary==============================")
    print(summary(glm1))
    print("=============Testing the model by making prediction==============")
    probs <- predict(glm1, newdata=test, type="response")
    pred <- ifelse(probs>0.5, 1, 0)
    acc <- mean(pred==test$Income_Category)
    print("===================printing accuracay===========================")
    print(paste("accuracy = ", acc))
    table(pred, test$Income_Category)
}
```

Using our logistical regression Function

```
print("****************** Using moreThan120k ********************************")
```

```
## [1] "****************** Using moreThan120k ********************************"
```

```
logistical_regression(moreThan120k,i)
```

```
## [1] "==================printing Summary============================="
##
## Call:
## glm(formula = Income_Category ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.50980  -0.39990  -0.00005  -0.00004   2.60500
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -2.254e+01  2.956e+02  -0.076  0.93921
## Customer_Age                  3.026e-02  7.269e-03   4.164 3.13e-05 ***
## GenderM                       1.807e+01  2.956e+02   0.061  0.95125
## Education_LevelDoctorate      5.059e-01  2.962e-01   1.708  0.08772 .
## Education_LevelGraduate      -4.877e-03  2.037e-01  -0.024  0.98090
## Education_LevelHigh School    6.183e-02  2.142e-01   0.289  0.77289
## Education_LevelPost-Graduate -4.037e-01  3.266e-01  -1.236  0.21641
## Education_LevelUneducated     2.044e-01  2.246e-01   0.910  0.36280
## Education_LevelUnknown        3.653e-01  2.210e-01   1.653  0.09842 .
## Card_CategoryGold            -1.103e+00  3.662e-01  -3.013  0.00259 **
## Card_CategoryPlatinum        -7.781e-01  7.136e-01  -1.090  0.27554
## Card_CategorySilver          -1.503e+00  2.325e-01  -6.463 1.03e-10 ***
## Credit_Limit                  8.646e-05  5.369e-06  16.104  < 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: 3243.6  on 6487  degrees of freedom
## Residual deviance: 2222.3  on 6475  degrees of freedom
##   (1613 observations deleted due to missingness)
## AIC: 2248.3
##
## Number of Fisher Scoring iterations: 19
##
## [1] "=============Testing the model by making prediction============="
## [1] "==================printing accuracay=========================="
## [1] "accuracy =  0.924984500929944"
```

```
##
## pred    0    1
##    0 1477  113
##    1    8   15
```

```
print("***************** Using between40to60k ********************************")
```

```
## [1] "****************** Using between40to60k *******************************"
```

```
logistical_regression(between40to60k,i)
```

```
## [1] "==================printing Summary=============================="
## 
## Call:
## glm(formula = Income_Category ~ ., family = "binomial", data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.2546  -0.7059  -0.6094  -0.3105   2.3120  
## 
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                 -1.004e+00  2.155e-01  -4.659 3.18e-06 ***
## Customer_Age                -9.630e-04  4.004e-03  -0.240    0.810    
## GenderM                      3.384e-01  7.123e-02   4.750 2.03e-06 ***
## Education_LevelDoctorate    -2.369e-01  1.983e-01  -1.195    0.232    
## Education_LevelGraduate      4.735e-02  1.172e-01   0.404    0.686    
## Education_LevelHigh School  -7.413e-03  1.261e-01  -0.059    0.953    
## Education_LevelPost-Graduate  2.340e-01  1.695e-01   1.380    0.167    
## Education_LevelUneducated   -1.836e-02  1.344e-01  -0.137    0.891    
## Education_LevelUnknown      -5.082e-02  1.341e-01  -0.379    0.705    
## Card_CategoryGold            1.888e+00  3.586e-01   5.266 1.39e-07 ***
## Card_CategoryPlatinum        7.810e-01  1.057e+00   0.739    0.460    
## Card_CategorySilver          1.466e+00  1.850e-01   7.922 2.33e-15 ***
## Credit_Limit                -1.008e-04  7.225e-06 -13.945  < 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: 6150.5  on 6487  degrees of freedom
## Residual deviance: 5865.8  on 6475  degrees of freedom
##   (1613 observations deleted due to missingness)
## AIC: 5891.8
## 
## Number of Fisher Scoring iterations: 5
## 
## [1] "============Testing the model by making prediction============="
## [1] "==================printing accuracay=========================="
## [1] "accuracy =  0.841289522628642"
```

```
## 
## pred    0    1
##    0 1356  255
##    1    1    1
```

```
print("****************** Using between60to80k ********************************")
```

```
## [1] "****************** Using between60to80k ********************************"
```

```
logistical_regression(between60to80k,i)
```

```
## [1] "===================printing Summary=============================="
## 
## Call:
## glm(formula = Income_Category ~ ., family = "binomial", data = train)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.22571  -0.74124  -0.00005  -0.00005   1.84945  
## 
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                  -2.022e+01  3.009e+02  -0.067  0.94642    
## Customer_Age                 -5.036e-03  4.927e-03  -1.022  0.30674    
## GenderM                       1.987e+01  3.009e+02   0.066  0.94735    
## Education_LevelDoctorate     -9.968e-02  2.410e-01  -0.414  0.67912    
## Education_LevelGraduate       5.006e-04  1.465e-01   0.003  0.99727    
## Education_LevelHigh School    8.392e-02  1.543e-01   0.544  0.58648    
## Education_LevelPost-Graduate  3.463e-02  2.125e-01   0.163  0.87055    
## Education_LevelUneducated    -2.540e-02  1.685e-01  -0.151  0.88022    
## Education_LevelUnknown        1.207e-02  1.662e-01   0.073  0.94214    
## Card_CategoryGold             1.279e+00  3.224e-01   3.968 7.24e-05 ***
## Card_CategoryPlatinum         1.305e+00  6.597e-01   1.977  0.04799 *  
## Card_CategorySilver           5.975e-01  1.872e-01   3.192  0.00141 ** 
## Credit_Limit                 -3.359e-05  4.901e-06  -6.853 7.23e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 5125.0  on 6487  degrees of freedom
## Residual deviance: 3583.9  on 6475  degrees of freedom
##   (1613 observations deleted due to missingness)
## AIC: 3609.9
## 
## Number of Fisher Scoring iterations: 19
## 
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy =  0.854308741475512"
```

```
##
## pred    0    1
##    0 1378  232
##    1    3    0
```

```
print("****************** Using between80to120 ********************************")
```

```
## [1] "****************** Using between80to120 ********************************"
```

```
logistical_regression(between80to120k,i)
```

```
## [1] "==================printing Summary=============================="
## 
## Call:
## glm(formula = Income_Category ~ ., family = "binomial", data = train)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.45770  -0.72859  -0.00005  -0.00004   1.78881  
## 
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                 -2.065e+01  2.995e+02  -0.069   0.9450    
## Customer_Age                 5.176e-04  4.979e-03   0.104   0.9172    
## GenderM                      1.947e+01  2.995e+02   0.065   0.9482    
## Education_LevelDoctorate    -1.974e-01  2.315e-01  -0.853   0.3939    
## Education_LevelGraduate     -1.910e-01  1.413e-01  -1.352   0.1765    
## Education_LevelHigh School  -1.279e-01  1.495e-01  -0.856   0.3922    
## Education_LevelPost-Graduate -1.178e-01  2.067e-01  -0.570   0.5686    
## Education_LevelUneducated   -2.634e-01  1.628e-01  -1.618   0.1057    
## Education_LevelUnknown      -2.297e-01  1.616e-01  -1.422   0.1552    
## Card_CategoryGold           -1.595e+00  3.609e-01  -4.419 9.93e-06 ***
## Card_CategoryPlatinum       -1.780e+00  7.990e-01  -2.228   0.0259 *  
## Card_CategorySilver         -7.584e-01  1.733e-01  -4.377 1.20e-05 ***
## Credit_Limit                 5.206e-05  4.285e-06  12.149  < 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: 5536.2  on 6487  degrees of freedom
## Residual deviance: 3669.4  on 6475  degrees of freedom
##   (1613 observations deleted due to missingness)
## AIC: 3695.4
## 
## Number of Fisher Scoring iterations: 19
## 
## [1] "=============Testing the model by making prediction============="
## [1] "==================printing accuracay==========================="
## [1] "accuracy =  0.843149411035338"
```

```
##     
## pred    0    1
##    0 1318  217
##    1   36   42
```

```
print("***************** Using lessthan40k ********************************")
```

```
## [1] "***************** Using lessthan40k ********************************"
```

```
logistical_regression(lessthan40k,i)
```

```
## [1] "===================printing Summary=============================="
## 
## Call:
## glm(formula = Income_Category ~ ., family = "binomial", data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.7672  -0.5425  -0.1106   0.7780   2.8142  
## 
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                   1.634e+00  2.305e-01   7.087 1.37e-12 ***
## Customer_Age                 -2.094e-03  4.222e-03  -0.496    0.620    
## GenderM                      -2.619e+00  8.862e-02 -29.556  < 2e-16 ***
## Education_LevelDoctorate     -1.633e-01  1.963e-01  -0.832    0.405    
## Education_LevelGraduate       1.550e-01  1.227e-01   1.263    0.207    
## Education_LevelHigh School    5.672e-02  1.320e-01   0.430    0.667    
## Education_LevelPost-Graduate -1.200e-02  1.863e-01  -0.064    0.949    
## Education_LevelUneducated     4.460e-02  1.399e-01   0.319    0.750    
## Education_LevelUnknown        1.790e-01  1.388e-01   1.289    0.197    
## Card_CategoryGold             4.734e+00  5.366e-01   8.823  < 2e-16 ***
## Card_CategoryPlatinum         5.496e+00  1.321e+00   4.160 3.18e-05 ***
## Card_CategorySilver           3.702e+00  2.559e-01  14.466  < 2e-16 ***
## Credit_Limit                 -2.904e-04  1.339e-05 -21.697  < 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: 8380.3  on 6487  degrees of freedom
## Residual deviance: 5140.4  on 6475  degrees of freedom
##   (1613 observations deleted due to missingness)
## AIC: 5166.4
## 
## Number of Fisher Scoring iterations: 7
## 
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy =  0.806571605703658"
```

```
## 
## pred   0   1
##    0 839 121
##    1 191 462
```

Summary

The first thing the summary shows is the **formula**. This formula is used to predict the Income_category as

function of other features as the predictor.
The second thing the summary shows is the **Deviance Residuals**. This is the measure of deviance calculated from each observation made on the dataset.
Next is the **Coefficients**. The Coefficients in logistic regression is calculated by using log odds of Y. As it can be seen in the model above (Model for lessthan40k) that the P value of GenderM, different card Categories and credit limit is very low which is really good for the model.
The bottom part of the summary shows the **Null deviance** and **Residual deviance**. Null deviance is based on the Intercept. If the predictors have an affect on the model we would see a decrease from Null Deviance to Residual Deviance. In the model above (lessthan40k) we can see there is about a 40% decrease from Null deviance to Residual deviance meaning the model is good. Apart from this model, all the model above shows a decrease in Residual deviance.
AIC stands for Akaike Information Criterion. It is used to compare model.
Number of Fisher scoring iterations refers to the number of iteration it took for optimizating the algorithm.

Prediction result

The prediction was made by using predict function. predict function uses the logistic regression model that was trained for set of values. We store the predict function value into probs (short for probability) variable. After that we sort out the probability using if else in which probabilities less than 0.5 were stored as 0 and probabilities greater than 0.5 were stored as one. After that we took the mean to find the accuracy of our model. As it turns out for models for different classification the least accuracy was 0.80 meaning our model was able to predict 80% of the value with 20% error while the highest accuracy was 0.92. A table is also printed showing the actual numbers. This table is also called confusion matrix. The table can be divided into this manner:
True_Positive False_Positive False_Negative True_Negative
For this table we look at the diagnol value (True_Positive and True_Negative). For the model above (lessthan40k) we can see that the model was able to predict that 839 customers falls into the category of less than 40k yearly income while 462 doesn’t. While on the other hand our model predicted that 121 customers fall under less than 40k yearly income which was not true.

Naive Bayes, Summary And Prediction

(Note: parts related to Naive Bayes are consolidated together in this section because there are multiple classification of target so to make a model for these multiple classification I used a function. The model inside the function was inaccessible outside the function)

Summary

For Naive Bayes we have to first load the library e1071 and then run the naive bayes model on the training data. Here we are trying to predict Income_Category based on the Customer_Age, Gender, Education_Level, Card_Category and Credit_Limit

```
library(e1071)
nb1 <- naiveBayes(Income_Category~ ., data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        $120K +    $40K - $60K    $60K - $80K   $80K - $120K Less than $40K
##     0.07073201     0.17713862     0.13640291     0.15393161     0.35020368
##        Unknown
##     0.11159116
##
## Conditional probabilities:
##                 Customer_Age
## Y                    [,1]     [,2]
##   $120K +        47.55323 6.792585
##   $40K - $60K    46.18815 8.377645
##   $60K - $80K    46.04887 7.584868
##   $80K - $120K   46.30874 6.852384
##   Less than $40K 46.26190 8.477682
##   Unknown        46.22013 8.690392
##
##                 Gender
## Y                         F          M
##   $120K +        0.00000000 1.00000000
##   $40K - $60K    0.56376307 0.43623693
##   $60K - $80K    0.00000000 1.00000000
##   $80K - $120K   0.00000000 1.00000000
##   Less than $40K 0.91963342 0.08036658
##   Unknown        0.95132743 0.04867257
##
##                 Education_Level
## Y                   College  Doctorate   Graduate High School Post-Graduate
##   $120K +        0.09424084 0.05584642 0.28446771  0.19895288    0.04013962
##   $40K - $60K    0.10522648 0.03832753 0.31428571  0.19930314    0.06202091
##   $60K - $80K    0.09683258 0.04253394 0.30678733  0.21990950    0.05339367
##   $80K - $120K   0.11948677 0.03528468 0.30954290  0.20048115    0.05372895
##   Less than $40K 0.09728587 0.04300317 0.31970391  0.19034191    0.04723299
##   Unknown        0.10287611 0.06637168 0.30309735  0.19247788    0.04203540
##                 Education_Level
## Y                Uneducated    Unknown
##   $120K +        0.16928447 0.15706806
##   $40K - $60K    0.13797909 0.14285714
##   $60K - $80K    0.13755656 0.14298643
##   $80K - $120K   0.13953488 0.14194066
##   Less than $40K 0.14839619 0.15403595
##   Unknown        0.16150442 0.13163717
##
##                 Card_Category
## Y                        Blue         Gold     Platinum       Silver
##   $120K +        0.9022687609 0.0244328098 0.0069808028 0.0663176265
##   $40K - $60K    0.9365853659 0.0090592334 0.0006968641 0.0536585366
##   $60K - $80K    0.9085972851 0.0208144796 0.0036199095 0.0669683258
##   $80K - $120K   0.9093825180 0.0128307939 0.0016038492 0.0761828388
##   Less than $40K 0.9531194924 0.0074021854 0.0014099401 0.0380683821
##   Unknown        0.9402654867 0.0077433628 0.0044247788 0.0475663717
##
##                 Credit_Limit
## Y                     [,1]      [,2]
##   $120K +        19454.403 12019.697
##   $40K - $60K     5516.629  4556.732
##   $60K - $80K    10811.296  8935.500
##   $80K - $120K   15517.168 11292.735
##   Less than $40K  3759.196  2800.310
##   Unknown         9339.504  8586.559
```

The Naive bayes uses laplace smoothing. The summary of the model shows us the **A-priori probablities** on the training data. It tells us that there is 0.35020368 prior probability that customer falls under Less than $40K and we can see other prior probability under the classified groups. We also have the **conditional probability**. lets take gender for an example. Our model shows that the probability that a person that falls into income category of 120k+ is mostly male. It also tells as that there is a 0.56 probability that a person who falls under 40K - 60K is female. For quantitative predictors like Customer_Age and Credit_Limit we get the standard deviation and the mean

```
naive_bayes <- function(df,i){
  train <- df[i,]
  test <- df[-i,]
  nb1 <- naiveBayes(Income_Category~ ., data=train)
  print("=============Testing the model by making prediction==============")
  p1 <- predict(nb1, newdata=test, type="class")
  print("==================printing accuracay==========================")
  accuracy <- mean(p1==test$Income_Category)
  print(paste("accuracy:", accuracy))
  print(table(p1, test$Income_Category))
}
```

```
print("***************** Using moreThan120k ********************************")
```

```
## [1] "***************** Using moreThan120k ********************************"
```

```
naive_bayes(moreThan120k,i)
```

```
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy: 0.893366398016119"
##
## p1     0    1
##   0 1394   81
##   1   91   47
```

```
print("****************** Using between40to60k ********************************")
```

```
## [1] "****************** Using between40to60k ********************************"
```

```
naive_bayes(between40to60k,i)
```

```
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy: 0.841289522628642"
##
## p1     0    1
##   0 1357  256
##   1    0    0
```

```
print("****************** Using between60to80k ********************************")
```

```
## [1] "****************** Using between60to80k ********************************"
```

```
naive_bayes(between60to80k,i)
```

```
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy: 0.836949783013019"
##
## p1     0    1
##   0 1334  216
##   1   47   16
```

```
print("****************** Using between80to120 ********************************")
```

```
## [1] "****************** Using between80to120 ********************************"
```

```
naive_bayes(between80to120k,i)
```

```
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy: 0.830750154990701"
##
## p1     0    1
##    0 1248  167
##    1  106   92
```

```
print("****************** Using lessthan40k *********************************")
```

```
## [1] "****************** Using lessthan40k *********************************"
```

```
naive_bayes(lessthan40k,i)
```

```
## [1] "=============Testing the model by making prediction=============="
## [1] "===================printing accuracay==========================="
## [1] "accuracy: 0.813391196528208"
##
## p1    0   1
##    0 804  75
##    1 226 508
```

Evaluate and Predict:

Evaluation was made using predict function on nb1 (naive bayes) model. We also print the confusion matrix. The accuracy was calculated as the mean accuracy. **(for more information check prediction result under Regression model. It follows the same patern)**

Strengths and Weaknesses of Logistic Regression and Naive Bayes

Logistic regression makes the estimate on the probability directly of Y on the give value of X and such and is refer as discriminative classifier. Naive bayes makes the estimate on the probability directly of Y, classification of the features and the likelihood of the data on given Y and refer to as a generative classifier. Naive Bayes will outperform Logistic regression if the data used is small. If you have more data then logistic regression outperforms Naive Bayes. Naive Bayes have lower variance but higher bias than logistic regression. Naive Bayes is easier to implement and interpret. Naive bayes makes guesses on the data set which it wasn’t train for.