PREDICTIVE ANALYTICS

Northeastern University

ALY6020, SPRING 2020

MODULE 5 PROJECT ASSIGNMENT

WEEK 5: GENERALIZED LINEAR MODELS & LOGISTIC REGRESSIONS

SUBMITTED BY: SHIVANI ADSAR

NUID: 001399374

SUBMITTED TO: PROF. NA YU

DATE: 05/09/2020

Introduction

The assignment aims at providing practical experience on implementation of Generalized Linear Models (GLM) and Logistic Regressions. The linear regression uses an explanatory and dependent variable to establish a relationship between the two variables. Moreover, logistic regression uses a binary variable to predict the outcome. We have used the "Rats" and "Adult Data" datasets for performing analysis.

Analysis

Rats Dataset

- The Rats Dataset is an inbuilt dataset in R studio. This dataset has 300 instances and 5 attributes. The dataset is used for studying the tumors after effects of drugs and placebos on rats. [1]
- In order to perform the predictions, we have used the Generalized Linear Regression algorithm.

Analysis on the Rats Dataset

- The "Rats" dataset has been imported initially using the library, "survival". Using the "str" function, the structure of the dataset has been analyzed into 300 observations and 5 variables, namely, litter, rx, time, status and sex.

```
> library(survival)
> #Getting rats data and stored in vector rat
> rat<-rats
> #Checking structure of data
> str(rat)
'data.frame':   300 obs. of  5 variables:
 $ litter: int  1 1 1 2 2 2 3 3 3 4 ...
 $ rx    : num  1 0 0 1 0 0 1 0 0 1 ...
 $ time  : num  101 49 104 91 104 102 104 102 104 91 ...
 $ status: num  0 1 0 0 0 0 0 0 0 0 ...
 $ sex   : chr  "f" "f" "f" "m" ...
```

Fig.1: Structure of the Rats dataset

The "str" function is used to view the structure of the dataset for understanding the variables for further analysis.

- We have used the glm() function for building our linear model. Moreover, the cbind() combines the data frame by columns. Also, the linear regression is performed on the "status" column being used as the target variable and "rx" which shows the treatment, used as the dependent variable. We have used, "family = binomial" for categorical variables on our "rats" data.

```
> fit<-glm(cbind(status,1-status)~rx, family = binomial,data = rats)
> fit

Call:  glm(formula = cbind(status, 1 - status) ~ rx, family = binomial,
    data = rats)

Coefficients:
(Intercept)           rx
    -2.1429       0.8179

Degrees of Freedom: 299 Total (i.e. Null);  298 Residual
Null Deviance:      243
Residual Deviance: 237.2        AIC: 241.2
```

Fig.2: Output of glm() function

As observed, the output shows, "Coefficients" which is the effect of independent variable on the dependent variable. We can see -2.1429, the sign indicates direction of variable. The "Degree of Freedom" is 299, with null deviance as 243, which shows prediction of response variable and residual deviance shows the measure of goodness fit for our model. The AIC is 241.2, shows quality of our model.

- The AIC is known as Akaike information criterion, which helps in better model selection by comparing the AIC values amongst models. Hence, lesser the AIC value, better is the model. [1]
- In order to better understand the "coefficient" parameter, we have executed a summary() function. The coefficients are fitted give the default values for binomial family data, hence is a logistic regression.

We can interpret, that, Estimate, Standard Error and Z- value are parameters for comparisons of coefficients.

```
> summary(fit)$coef
             Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -2.1428634  0.2306523 -9.290447 1.536412e-20
rx           0.8179379  0.3368645  2.428092 1.517851e-02
```

Fig. 3: Summary() function

- We can find the standard error estimates by the square root of diagonal in variance function. This gives the estimates from an unscaled co-variance matrix.

As see, the standard error estimates for the "rx" parameter are, 33% and 23%.

```
> sqrt(diag(summary(fit)$cov.scaled))
(Intercept)          rx
  0.2306523   0.3368645
```

Fig.4: Error estimation

- The quantities such as the deviance and log-likelihood can be retrieved from a fitted object through functions like loglik().
  Where, Deviance is equal to the Residual Deviance, which equals -2*logLik

The "loglik" function returns number of attributes. Degrees of freedom which account to parameters in the model. We see, 237.1642 is the returned fitted parameter.

```
> c(fit$deviance, -2*logLik(fit))
[1] 237.1642 237.1642
```

Fig.5: loglik() function

- We have used the "update" function to refit the previously fitted model.
  As seen, the formula shows, a "." Which means that the previous model has been used, and instead of "rx", predictor has changed to an intercept of 1. [1]

```
> c(fit$null.deviance, -2*logLik(update(fit, formula = .~1)))
[1] 242.9781 242.9781
```

Fig.6: Update() function

- Further, we have used the Update() function to change the link present in the binomial family. We have used the rbind() function that combines the matrix by rows.

```
> fit2<-update(fit, family=binomial(link = "probit"))
> rbind(logit=fit$coefficients, probit=fit2$coefficients, rescale.probit=fit2$coefficients/0.5513)
              (Intercept)        rx
logit           -2.142863 0.8179379
probit          -1.253565 0.4471442
rescale.probit  -2.273835 0.8110724
```

Fig.7: rbind() function

- According to Wilk's Theorem, there exists a difference of 2 times between loglik of a model with 'p' parameter dimensions vs. loglik for a base model, which is equal to chi-square (p,df), when base model is true.
- Moreover, we have analysed the model quality with a predictor, log (time) with 'rx'

```
> fit3<-update(fit, formula=.~. + I(log(time)))
```

- We have used the I(log(time))) to ensure the evaluation of the predictor within the glm() function. We have used the same data as we did for the "fitB1", hence "time" denotes one of the columns in the dataset. As we have seen earlier, log(time) is our created predictor in "glm". [1]
- Since the time range does not vary drastically for different rats, we would not use the "time" variable as our response variable.

```
> fit3<-update(fit, formula=.~. + I(log(time)))
> summary(rats$time[rats$status==1])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  34.00   67.25   79.50   77.07   91.25  104.00
> summary(rats$time[rats$status==0])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  23.00   86.25  102.00   92.62  104.00  104.00
```

The summary() function shows the minimum, median, mean and maximum values for survival status of rats as 1 and 0. As see, the values are higher for status of survival 1 than 0.

Fig.8: The model quality analysis

- We have used the cbind() function to view the updated columns in our matrix.

```
> cbind(rats[1:10,], model.matrix(fit3)[1:10,])
   litter rx time status sex (Intercept) rx I(log(time))
1       1  1  101      0   f           1  1     4.615121
2       1  0   49      1   f           1  0     3.891820
3       1  0  104      0   f           1  0     4.644391
4       2  1   91      0   m           1  1     4.510860
5       2  0  104      0   m           1  0     4.644391
6       2  0  102      0   m           1  0     4.624973
7       3  1  104      0   f           1  1     4.644391
8       3  0  102      0   f           1  0     4.624973
9       3  0  104      0   f           1  0     4.644391
10      4  1   91      0   m           1  1     4.510860
```

As we notice, the first 4 columns ie. Litter, rx, time, status and sex are the columns in the original data. The columns, Intercept, rx and I(log(time)), last 3 columns are created by the glm() function.

Fig.9: cbind() function for viewing updated columns()

- The coefficients of the new model 'fit3' are:

```
> summary(fit3)$coefficients
                Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  10.1398138  2.7847889  3.641143 2.714306e-04
rx            0.8106187  0.3552044  2.282119 2.248231e-02
I(log(time)) -2.7717764  0.6339362 -4.372327 1.229291e-05
```

The coefficients as estimate, standard error, z-vale and Pr (>|z|) are viewed for rx and I(log(time)).

Fig.10: Coefficients of the 'fit3' model

- The deviances of the fitB1 and logLik models can be compared using:

```
> c(2*(logLik(fit3)-logLik(fit)),fit$deviance-fit3$deviance)
[1] 20.22656 20.22656
```

Fig11: Comparison of fitB1 and logLik

We can see the values of deviances are 20.22656.

- The likelihood ratio test has been compared with chisquare 1 df.

```
> 1-pchisq(20.226,1)
[1] 6.881132e-06
```

Fig.12: Comparison with Chi-square values

The p-value for chi-square is 6.881132e-06, which is different p-value from the I(log(time)) coefficient. The model still needs improvements.

- In addition, we have tried to add some parameters like log(time)^2 or rx*log(time) for better comparisons of coefficients.

```
> fit4<-update(fit3, .~. + I(rx*log(time))+I(log(time)^2))
> #get summary of ths fit4 model
> summary(fit4)$coefficients
                  Estimate Std. Error    z value   Pr(>|z|)
(Intercept)     -64.273349  31.968344 -2.0105311 0.04437501
rx               -5.209379   5.833052 -0.8930795 0.37181458
I(log(time))     33.842043  15.211011  2.2248385 0.02609208
I(rx * log(time))  1.401760   1.335106  1.0499238 0.29375316
I(log(time)^2)    -4.466465   1.807551 -2.4710041 0.01347343
> #Check deviance
> fit3$deviance-fit4$deviance
[1] 10.64098
```

Fig.13: Summary of fit4 model

As observed, the coefficients do not look significant. We have tried this method for getting a different p-value, however, the values do not seem significant. The deviance for fit4 model shows 10.64 as compared to the previous models.

- In order to perform better analysis of the deviance tables, we have used the anova(). The Analysis of Variance analyses the differences in the group means and variances.

```
> anova(fit4)
Analysis of Deviance Table

Model: binomial, link: logit

Response: cbind(status, 1 - status)

Terms added sequentially (first to last)


                 Df Deviance Resid. Df Resid. Dev
NULL                                299     242.98
rx                1   5.8139       298     237.16
I(log(time))      1  20.2266       297     216.94
I(rx * log(time)) 1   2.8112       296     214.13
I(log(time)^2)    1   7.8298       295     206.30
```

We can observe, the RSS values, which is Residual Sum of Squares which is used for measuring the variance that cannot be identifies by regression models. It measures the variance between the regression model and the dataset.

Fig.14: Checking the deviance through anova()

- We have further analysed the differences in the deviances for better comparisons.

```
> devs<-c(fit$null.deviance, fit$deviance, fit3$deviance, update(fit3, .~.+I(rx*log(time)))$deviance, fit4$deviance)
> devs
[1] 242.9781 237.1642 216.9377 214.1265 206.2967
> round(-diff(devs), 3)
[1]  5.814 20.227  2.811  7.830
```

Fig.15: Analysis of Deviances

- It can be noted, that residuals have replaced the deviance values, with highest deviance being 242.9781. [1]
- After performing analysis on the Rats dataset, it can be concluded that ‘status’ and ‘time’ have been significant variables in predicting the correlation for existence of tumor in rats.

Adult Dataset

- The Adult Dataset has been imported from the website, Kaggle. This dataset has 32561 instances and 15 attributes. The dataset is used for studying if the person earns more than 50k in a year. [2]
- In order to perform the predictions, we have used the Logistic Regression algorithm.

Importing the Data

- The "Adult" dataset has been imported by reading the csv file. Using the "str" function, the structure of the dataset has been analyzed into 32561 observations and 15 variables.

```
> incomedata <- read.csv("C:/Users/Shivani Adsar/OneDrive/Desktop/Northeastern University/Predictive Analytics/Module 5/adult.data.csv")
> #Check structure of data
> str(incomedata)
'data.frame':   32561 obs. of  15 variables:
 $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
 $ workclass     : Factor w/ 9 levels "?","Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
 $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
 $ education     : Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 13 10 ...
 $ education.num : int  13 13 9 7 13 14 5 9 14 13 ...
 $ marital.status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
 $ occupation    : Factor w/ 15 levels "?","Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
 $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
 $ sex           : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
 $ capital.gain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ capital.loss  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hours.per.week: int  40 13 40 40 40 40 16 45 50 40 ...
 $ native.country: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
 $ salary        : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

The "str" function is used to view the structure of the dataset for understanding the variables for further analysis.

Fig.16: Structure of 'Adult' dataset

- We have installed the "dplyr" package for using the mutate function.
  The mutate function will convert the salary lesser than 50K into 0's and greater than 50K into 1's, ie. into categorical variables so that it is easier for performing logistic regression.

```
> incomedata<-incomedata %>%
+   mutate(salary=ifelse(salary=="<=50K",0, 1))
```

Fig.17: Mutate function

Check the Class Bias

- We have checked the class bias to view the proportion of I's and 0's in our target variable. [2]

```
> table(incomedata$salary)

    0     1
24720  7841
```

As observed, there is a class bias with 0 having 24720 values and 1 having 7841 values. Hence, we need to sample them equally.

Fig.18: Checking the Class Bias

Creating Testing and Training Samples

- In order to avoid the class bias, we would be allocating most of the input data in the testing sample / validation sample, and lesser proportion into the training sample, as our data contains many values. [2]

```
> # Create Training Data
> income_ones <- incomedata[which(incomedata$salary == 1), ]  # all 1's
> income_zeros <- incomedata[which(incomedata$salary == 0), ]  # all 0's
> set.seed(100)  # for repeatability of samples
> income_ones_training_rows <- sample(1:nrow(income_ones), 0.7*nrow(income_ones))  # 1's for training
> income_zeros_training_rows <- sample(1:nrow(income_zeros), 0.7*nrow(income_ones))  # 0's for training. Pick as many 0's as 1's
> income_training_ones <- income_ones[income_ones_training_rows, ]
> income_training_zeros <- income_zeros[income_zeros_training_rows, ]
> income_trainingData <- rbind(income_training_ones, income_training_zeros)  # row bind the 1's and 0's
```

Fig.19: Allocation of Training Sample

- The 1's and 0's have been collected form the salary column for having equal proportion of 0's and 1's in our training data. [2]
- The set.seed(100) function takes 100 random samples and repeats them in a sequence.
- The 1's and 0's have been combined using the rbind() function.
- Similarly, we have created the testing sample, with equal proportion of 1's and 0's and then the data has been combined using the rbind() function.

```
> # Create Test Data
> income_test_ones <- income_ones[-income_ones_training_rows, ]
> income_test_zeros <- income_zeros[-income_zeros_training_rows, ]
> income_testData <- rbind(income_test_ones, income_test_zeros)  # row bind the 1's and 0's
```

Fig.20: Allocation of Testing Sample

Information Values

- We have used the 'smbinning' package which performs score modeling and discretization, ie. conversion of continuous variables into bins. This gives a better understanding of the distribution with a binary variable.
- The continuous and factor variables are segregated and then combined in a dataframe.

```
> library(smbinning)
> factor_vars <- c ("workclass", "education", "marital.status", "occupation", "relationship", "race", "sex", "native.country")
> continuous_vars <- c("age", "fnlwgt","education.num", "hours.per.week", "capital.gain", "capital.loss")
> iv_df <- data.frame(VARS=c(factor_vars, continuous_vars), IV=numeric(14))  # init for IV results
```

Fig.21: Segregation of Continuous and Factor variables

- Further, we have computed the information values for categorical variables.
  This has been performed for salary values, greater than 50K.
  Moreover we have used the WOE, Weight of Evidence and IV, Information Values as they help in performing effective analysis for binary classifiers. They help to establish linear and non-linear relationships amongst variables. Thus, finding correlations between the dependent and independent variables. [2]

```
> for(factor_var in factor_vars){
+   smb <- smbinning.factor(income_trainingData, y="salary", x=factor_var)  # WOE table
+   if(class(smb) != "character"){ # heck if some error occured
+     iv_df[iv_df$VARS == factor_var, "IV"] <- smb$iv
+   }
+ }
```

Fig.22: Information Value for Categorical Variables

We have created the WOE, 'Weight of Evidence' table that transforms the factors into binary classifications by calculation of weights.

- Similarly, we have calculated the information values for continuous variables.

```
> for(continuous_var in continuous_vars){
+   smb <- smbinning(income_trainingData, y="salary", x=continuous_var)  # WOE table
+   if(class(smb) != "character"){  # any error while calculating scores.
+     iv_df[iv_df$VARS == continuous_var, "IV"] <- smb$iv
+   }
+ }
```

Fig.23: Information Value for Continuous Variables

- We have performed sorting of the information values by using the order() function, in the decreasing order.

```
> iv_df <- iv_df[order(-iv_df$IV), ]  # sort
> iv_df
             VARS     IV
5    relationship 1.5342
9             age 1.2203
7             sex 0.2910
1       workclass 0.1665
6            race 0.0600
2       education 0.0000
3  marital.status 0.0000
4      occupation 0.0000
8  native.country 0.0000
10         fnlwgt 0.0000
11  education.num 0.0000
12 hours.per.week 0.0000
13   capital.gain 0.0000
14   capital.loss 0.0000
```

As observed, the Information Values for variables like, relationship, age, sex, workclass etc. show how well they are able to classify the response of above 50K or less 50K in the salary variable. So, relationship has the highest Information Value of 1.53 and capital.loss has the lowest value. Also, these variables would be significant for our predictions.

Fig.24: Sorted Information Values

Building Logit Models and Prediction

- We have used the glm() function with family='binomial' to build the logistic regression model in our training data. The left side of the '~' sign shows our independent variable, Salary and the variables on the right side of the sign show the dependent variables, like Relationship, Age, Capitalgain, Occupation and Educationnum. [2]

```
> logitMod <- glm(salary ~ relationship + age + capital.gain + occupation + education.num, data=income_trainingData, family=binomial
(link="logit"))
```

Fig.25: Logistic Model using glm()

- We have used the predict function which is used for predicting the logs of the salary variable.

```
> predicted <- predict(logitMod, income_testData, type="response")  # predicted scores
```

Fig.26: Prediction on model

The predict() function gives us the values between 0 and 1.

- In order to perform optimization on our model, we have used the optimal prediction cutoff. The cutoff score is 0.5 for models by default, for improving the accuracy in training and testing samples, we have used the optimalcutoff() function. This function provides optimal cutoff by improving prediction of 1's and 0's and minimize misclassification errors.
- We have installed 'InformationValue' library for performing analysis on performance of model and binary classifications.

```
> library(InformationValue)
> optcutoff <- optimalCutoff(income_testData$salary, predicted)[1]
> optcutoff
[1] 0.91
```

Fig.27: Optimisation on model

The optimal cutoff is seen to be 0.91 for our model.

Model Diagnostics

- Now, we have analysed the diagnostic using the summary() function.

```
> summary(logitMod)

Call:
glm(formula = salary ~ relationship + age + capital.gain + occupation +
    education.num, family = binomial(link = "logit"), data = income_trainingData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6410  -0.5380  -0.0058   0.6307   3.3190

Coefficients:
                          Estimate  Std. Error z value             Pr(>|z|)
(Intercept)             -4.73857173  0.24333539 -19.473 < 0.0000000000000002 ***
relationshipNot-in-family -2.23970817  0.07148961 -31.329 < 0.0000000000000002 ***
relationshipOther-relative -2.62526922  0.25579107 -10.263 < 0.0000000000000002 ***
relationshipOwn-child     -3.50571726  0.18178372 -19.285 < 0.0000000000000002 ***
relationshipUnmarried     -2.38371504  0.11290572 -21.112 < 0.0000000000000002 ***
relationshipWife           0.48824214  0.11074481   4.409  0.00001039865174799 ***
age                        0.02856987  0.00238190  11.995 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 15216.0  on 10975  degrees of freedom
Residual deviance:  8779.5  on 10953  degrees of freedom
AIC: 8825.5

Number of Fisher Scoring iterations: 8
```

We can interpret the beta coefficients, standard error, z-value and p-value. We can view entries for each category as the glm() function considers each category to be independent binary variable. The AIC is 8825.5 and fisher iterations are 8.

Fig.28: Summary of logitMod

- Since we are performing regression analysis, we need to check the multicollinearity where there exists a collinearity between more than1 variable. The multicollinearity between variables can be viewed by the Variance Inflation Factor, vif() function.

```
> library(car)
> vif(logitMod)
                    GVIF Df GVIF^(1/(2*Df))
relationship    1.323488  5        1.028424
age             1.121895  1        1.059196
capital.gain    1.023553  1        1.011708
occupation      1.732991 14        1.019832
education.num   1.463801  1        1.209876
```

As observed, the values of dependent variables are lesser than 4, which is appropriate.

Fig.29: Variance Inflation Factor

- We need to view the misclassification error which shows the percentage mismatch between actual and predicted variables.

```
> misClassError(income_testData$salary, predicted, threshold = optcutoff)
[1] 0.09
```

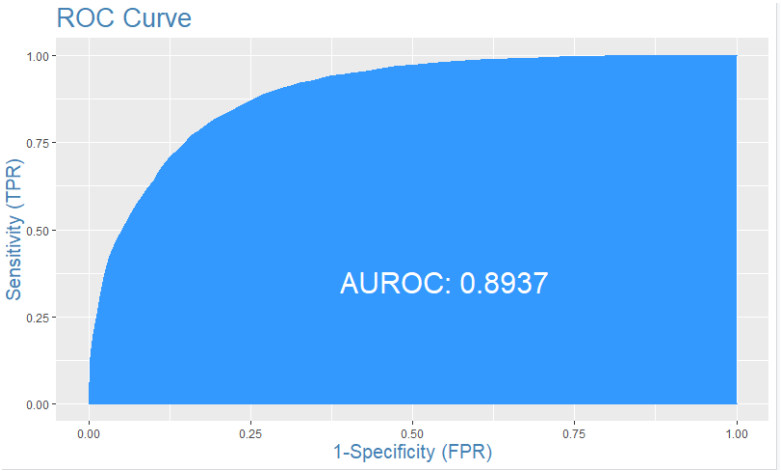
Fig.30: Misclassification error

We can see that, our model has 0.09 error, lower the error, better is the prediction.

- Further, we have plotted the ROC Curve, Receiver Operating Characteristics Curve which shows the diagnostics of a binary classifier by plotting the true positives against false positives.

- In order for a ROC curve to be good for a model, the curve should rise steeply and should have greater area under the curve as the cutoff is lowered, by indicating True Positive Rate increases faster than False Positive Rate.

```
plotROC(income_testData$salary, predicted)
```



It can be seen that area under the ROC curve is 89.37% which is good for our model.

Fig.31: ROC Curve

- We have computed the Concordance for our model, which gives the percentage of number of actual positive's (1's) in comparison to the negative's (0's), if the positive's are more than the negative's, the model is said to be concordant. Higher the concordance, better is the model. [2]

```
> Concordance(income_testData$salary, predicted)
$Concordance
[1] 0.8938266
```

The concordance in our model, is 89.38%, which is good for our model.

Fig.32: Concordance

- The Sensitivity is the percentage of 1's or positive's that are correctly predicted by our model. While, the Specificity is the percentage of 0's or Negative's correctly predicted by our model.

```
> sensitivity(income_testData$salary, predicted, threshold = optcutoff)
[1] 0.2881428
> #Specificity
> specificity(income_testData$salary, predicted, threshold = optcutoff)
[1] 0.9860649
```

Fig.33: Sensitivity and Specificity

As seen, the Sensitivity is 28.81% and Specificity is 98.60%.

- Confusion Matrix shows the performance of a classification model.

```
> confusionMatrix(income_testData$salary, predicted, threshold = optcutoff)
      0    1
0 18964 1675
1   268  678
```

Fig.34: Confusion Matrix

The True Negatives ie. 18964 show we predicted No, and the income is less than 50K. While, False Positives is 1675 as they were predicted to be greater than 50K but actually have lesser than 50K. Moreover, we predicted 268 as False Negatives and their salary

was more than 50K. In addition, 678 were True Positives, as their cases were predicted correctly as greater than 50K. The accuracy of the model is predicted to be 95%.

- After performing analysis on the Adult dataset for predicting the number of people with greater income than 50K using Logistic Regression, it was noted that the accuracy of the model is highest as 98% with the lowest classification error of 0.09%.

Conclusion

- After performing linear regression analysis on the Rats dataset, it can be concluded that 'status' and 'time' have been significant variables in predicting the correlation for existence of tumor in rats.
- After performing analysis on the Adult dataset for predicting the number of people with greater income than 50K using Logistic Regression, it was noted that the accuracy of the model is highest as 98% with the lowest classification error of 0.09%.

References

[1] STRICKLAND, PRESIDENT JEFFREY. PREDICTIVE ANALYTICS USING R. LULU COM, 2015.

[2] Prabhakaran, S. (n.d.). eval(ez_write_tag([[728,90],'r_statistics_co-box-3','ezslot_4',109,'0','0']));Logistic Regression. Retrieved from http://r-statistics.co/Logistic-Regression-With-R.html

[3] Hrek, I. (2019, November 28). UCI Adult. Retrieved from https://www.kaggle.com/ivanhrek/uci-adult