Binomial logistic regression

Author:
Sebastian Balle (016187546)

Alice Huynh (006955661)

Erik Tostado (013879487)

Aaron Wilson (016159999)

Supervisor: Dr. Hojin Moon

Department of Mathematics and Statistics
California State University, Long Beach
Long Beach, California, USA
January 6, 2019

Abstract

A binomial logistic regression analysis was performed to predict if an email is spam by using predictors that were based off emails that were sent to George Forman in 1999 from June to July. There are 57 predictor variables in the data: 48 predictor variables measure the frequency of certain words in percentage, 6 predictor variables measure the frequency of certain character in percentage and 3 predictor variables measures the average and longest length of uninterrupted sequences of capital letters as well as the total number of capital letters in continuous integers. The response variable is valued at 1 (spam) or 0 (non-spam). In the final logistic regression model, the predictor variables were reduced to 12. The result in the final model has 96.53 % ability to determine whether an email is spam or not. The key finding is that the selected 12 predictor variables are significant to identifying a spam email.

Contents

1 Data description

1.1 Description

Our data set is based off 4601 emails that were sent to George Forman in June-July of 1999. We used this data set to help us build a binomial logistic regression model. The data measures number frequency of selected words, selected characters, and capital letters in each e-mail. Each email is an Y_i, $1 \leq i \leq 4061$, given the value of 1 (spam) or 0 (non-spam). This is to help us identify the chances of an email being a spam mail.

1.2 Data types

We have 57 predictor variables in our data set. 48 are continuous variables that measure the recurrence of words (in percentage) that appears in the email. 6 are continuous variables that measure the recurrence of the characters in an email (in percentage). 1 is continuous real number, that measures the average length of uninterrupted sequences of capital letters that are selected in an e-mail. 1 is continuous integer that measures the length of uninterrupted sequences of capital letter. Our response variable is binary 1 means spam and and 0 means non spam.

2 | Methodology / Theory

2.1 Simple logistic regression

Simple logistic regression has a dichotomous response variable with the value of 1 or 0. $0 \leq \pi \geq$. We used the simple logistic regression to determine whether or not the predictor are significant enough to keep before continuing on to cross validation which will be explained in detail later on. 1

$$(Y|X_1) = \frac{e^{\beta_0+\beta_1 X_1}}{1+e^{\beta_0+\beta_1 X_1}} = \pi$$

where

$$Y = \begin{cases} 1 \\ 0 \end{cases}$$

Transformation

$$\left[\frac{\pi(X)}{1-\pi(X)}\right] = \beta_0 + \beta_1 X$$

2.2 Multiple logistic regression

Multiple logistic regression has a dichotomous response variable with the value of 1 or 0 and there are n predictor variables. $0 \leq \pi \geq$. If $\pi > 0.5$ there is a higher chance it's a spam because it closer to 1. If $\pi < 0.5$ there is a higher chance of being a non-spam. We use the equation below to create our model.

$$(Y|X_1, X_2 + ... + X_n) = \frac{e^{\beta_0+\beta_1 X_1+X_2+...+X_n}}{1+e^{\beta_0+\beta_1 X_1+\beta_X 2+...+\beta_n X_n}} = \pi$$

where

$$Y = \begin{cases} 1 & \text{spam} \\ 0 & \text{non spam} \end{cases}$$

Transformation

$$\left[\frac{\pi(X)}{1-\pi(X)}\right] = \beta_0 + \beta_1 X_1 + X_2 + ... + X_n$$

2.3 VIF

VIF is a formal method of detecting the presence of multicollinearity. This measures how much the variances of the estimated regression coefficients are inflated as compared to when the predictor variables are not linearly related.

We used VIF in the prescreening process and the cross validation to help use determine if the predictor in our model is linearly related and we do not want the predictor to be linearly related.

2.4 Cross validation

Cross validation is a machine learning technique that let you build a model more efficient and robust. Cross validation is often used when the dataset is not large enough. The method let you re sample the

data so many times that the model building can be done over and over again.

First the data is split into a training set and a test set. The training set is used to build the model and the test set is used later on to test the models accuracy. The k-fold cross validation is used in our example where $k = 10$. The k-fold cross validation is where the training set is divided into k randomly sampled subset. We then fit a model for each of these subset. When we have fitted 10 model to each of these subset we re sample the whole training set and does it all over again. We do this 20 times so we fit in total 200 model to a random sample of the training set. For each of these 200 models we extract 55 pvalues (β_0 and all the other 54 explanatory variables) and put these into a big matrix. We can then sum up how many times each of the individual variable are significant. We choose to include all variables in our model that had a count on 100 or greater

2.5 Forward / Backward selection

Forward selection is a model selection procedure where you start with the simplest model and add the variable that reduce the sum of square residuals (SSE). You keep on adding variable until no more variable can be added to the model that will improve the models accuracy.

Backward selection is a model selection procedure where you start with the largest model possible. This model include all the variables in the data set. For each round you remove a the variable that influences the model in a negative way. The SSE should decrease for each variable you remove. When no more variables can be removed that influence the model in a negative way the procedure is done.

The step function is used in this project with a direction of both ways. This means that R use a combination of forward and backward selection in order to choose the best model.

3 | Model building

3.1 Prescreening

To determine if the predictor variables are significant we split the data into 80% training set and 20% into test set. We used the training set and regressing the response variable against each predictor variable to obtain the p-value which is shown in Table 3.1. We sorted the p-value from highest to lowest and identified the p-values > 0.05. We were able to identify 3 predictor variables that were higher than 0.05. These are `word_freq_will`, `word_freq_parts`, and `word_freq_address`, and we took them out the data set. Then we check the variable inflation factor (VIF) of the 54 predictor variables that are left. The VIF was calculated to be 6.5, which does not gives us much information because it greater than 5 and less than 10. We continued to the cross-validation.

i	Perdictor Variable	p-value
12	word_freq_will	5.048146e-01
38	word_freq_parts	1.611175e-01
2	word_freq_address	1.569197e-01
41	word_freq_cs	2.580458e-02
47	word_freq_table	1.874331e-02
14	word_freq_report	2.937581e-03
4	word_freq_3d	2.798328e-03
49	char_freq_;	.934298e-03
...	...	...
7	word_freq_remove	9.856843e-50
24	word_freq_money	3.656744e-50
19	word_freq_you	2.025637e-53
16	word_freq_free	5.797688e-62
55	capital_run_length_average	1.164926e-66
52	char_freq_!	1.104771e-70
56	capital_run_length_longest	2.833842e-76
21	word_freq_your	6.735986e-94
53	char_freq_$	5.366225e-100

Table 3.1: p-value of $Y \sim X_i$

3.2 Cross Validation

By doing the cross validation we get the following count of each variable.

β_0	β_{50}	β_{49}	β_6	β_{14}	β_{25}	β_4	β_{23}	β_{52}	β_{43}	β_{53}
186	138	132	125	123	121	120	119	108	107	104
β_{26}	β_{42}	β_{15}	β_{48}	β_{21}	β_7	β_{22}	β_{24}	β_5	β_{17}	β_{10}
103	102	101	101	100	98	98	98	97	96	95
β_8	β_{39}	β_9	β_{19}	β_{33}	β_{16}	β_{31}	β_{46}	β_2	β_{11}	β_{54}
93	93	92	92	92	91	91	91	89	89	89
β_{18}	β_{34}	β_{20}	β_{28}	β_{51}	β_{13}	β_{40}	β_{12}	β_{41}	β_1	β_{36}
88	88	87	87	87	86	86	85	84	83	83
β_{47}	β_{29}	β_{35}	β_{37}	β_{38}	β_{45}	β_{27}	β_{44}	β_{30}	β_{32}	β_3
83	82	81	79	78	77	75	75	71	70	69

Table 3.2: Cross validation count

From table 3.2 that the variable with a count equal or greather than 100 that we will use in our model is as follow

$$\begin{aligned}\text{glm}(y_{train} \sim &\beta_0 + \beta_4 + \beta_6 + \beta_{14} + \beta_{15} + \beta_{21} + \beta_{23} + \beta_{25} \\ &+ \beta_{26} + \beta_{42} + \beta_{43} + \beta_{48} + \beta_{49} + \beta_{50} + \beta_{52} + \beta_{53}, \text{family} = \text{"binomial"})\end{aligned}$$

In variable name the model above correspond to the following model

$$\begin{aligned}\text{glm}(y_{train} \sim &\text{word\_freq\_our} + \text{word\_freq\_remove} + \text{word\_freq\_internet} + \text{word\_freq\_receive} \\ &+ \text{word\_freq\_free} + \text{word\_freq\_business} + \text{word\_freq\_you} + \text{word\_freq\_money} \\ &+ \text{word\_freq\_hp} + \text{word\_freq\_george} + \text{word\_freq\_technology} + \text{word\_freq\_re} \\ &+ \text{word\_freq\_edu} + \text{`char\_freq\_;`} + \text{`char\_freq\_¡} + \text{`char\_freq\_\$`} \\ &+ \text{capital\_run\_length\_average} + \text{capital\_run\_length\_longest}, \text{family} = \text{"binomial"})\end{aligned}$$

3.3 Forward / Backward selection

We then use the step function in R to calculate the final model. The implemented R-function gives us the following expression for the final model:

$$\begin{aligned}\text{glm}(y_{train} \sim &\text{word\_freq\_our} + \text{word\_freq\_remove} + \text{word\_freq\_free} + \text{word\_freq\_business} \\ &+ \text{word\_freq\_hp} + \text{word\_freq\_george} + \text{word\_freq\_re} + \text{word\_freq\_edu} \\ &+ \text{`char\_freq\_¡} + \text{`char\_freq\_\$`} + \text{capital\_run\_length\_average}, \text{family} = \text{"binomial"})\end{aligned}$$

It can be seen that the variable `word_freq_internet.word_freq_receive, word_freq_you, word_freq_money, word_freq_technology, ‘char_freq_;‘, capital_run_length_longest` was removed from the model.

3.4 Presentation of Final Model

Now, we wish to use our results from R seen in the below table to give a final presentation of our model. Therefore, we first define

$$Y = \begin{cases} 1 & \text{if the email is spam} \\ 0 & \text{if the email is not spam} \end{cases}$$

$$X_1 = \text{word\_freq\_our}$$
$$X_2 = \text{word\_freq\_remove}$$
$$X_3 = \text{word\_freq\_free}$$
$$X_4 = \text{word\_freq\_business}$$
$$X_5 = \text{word\_freq\_font}$$
$$X_6 = \text{word\_freq\_hp}$$
$$X_7 = \text{word\_freq\_george}$$
$$X_8 = \text{word\_freq\_re}$$
$$X_9 = \text{word\_freq\_edu}$$
$$X_{10} = \text{`char\_freq\_¡}$$
$$X_{11} = \text{`char\_freq\_\$`}$$
$$X_{12} = \text{capital\_run\_length\_longest}$$

Now, we can see in the below table that the estimates of the coefficients of each of the predictor variables listed above, as well as the intercept, are all significant and there they can all be included in our final model.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-1.4262	0.0923	-15.45	< 2e-16
word_freq_our	0.4445	0.0883	5.03	4.86e-07
word_freq_remove	3.0683	0.3979	7.71	1.24e-14
word_freq_free	1.1635	0.1494	7.79	6.87e-15
word_freq_business	1.4015	0.2135	6.57	5.18e-11
word_freq_font	0.1032	0.0447	2.31	0.0208
word_freq_hp	-3.0454	0.3447	-8.84	< 2e-16
word_freq_george	-13.5291	1.9646	-6.89	5.72e-12
word_freq_re	-0.9042	0.1712	-5.28	1.27e-07
word_freq_edu	-1.9303	0.3215	-6.00	1.93e-09
`char_freq_¡	0.5679	0.1002	5.67	1.43e-08
`char_freq_$`	10.5994	0.8764	12.09	< 2e-16
capital_run_length_longest	0.0150	0.0017	8.99	< 2e-16

Table 3.3: Title

Thus, using the above estimates we obtain the following logistic mean response function

$$\pi_Y = \frac{\exp(-1.443+0.4445X_1+3.836X_2+1.240X_3+1.812X_4+0.103X_5-2.682X_6-16.905X_7-0.824X_8-2.144X_9+0.409X_{10}+8.767X_{11}+0.017X_{12})}{1+\exp(-1.443+0.4445X_1+3.836X_2+1.240X_3+1.812X_4+0.103X_5-2.682X_6-16.905X_7-0.824X_8-2.144X_9+0.409X_{10}+8.767X_{11}+0.017X_{12})}$$

4 Prediction

4.1 Confusion matrix

Now we will use our final model on our 20% Test set with a cut-off of 0.5. This means we will predict a given email to be spam (and thus assign it a value of 1) if our model returns a value greater then 0.5 (and return a value of zero otherwise).
Therefore, in doing so we were able to obtain the following confusion matrix in Figure 4.1 for our Test set and then from that we can calculate the values for the accuracy, sensitivity, specificity, precision and negative predicted value.

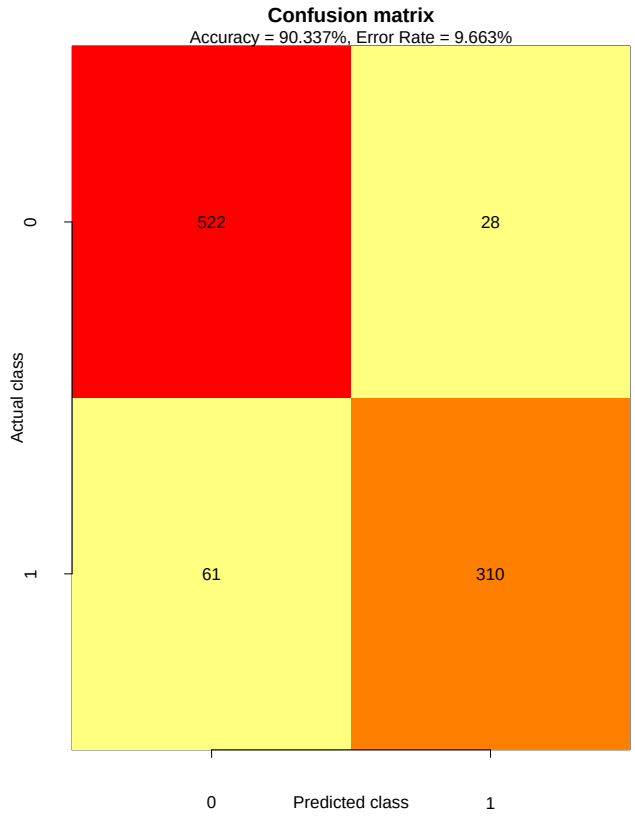

Figure 4.1: Confusion matrix

Now, we can calculate the accuracy using

$$Accuracy = \frac{522 + 310}{921} = 90.337\%$$

Thus we can see that we have an accuracy of 90.337%. This means that, using the data for the emails in the test set, our model correctly predicted whether the emails were spam or not 90% of the time, which is pretty good.

We now calculate the sensitivity and specificity by

$$Sensitivity = \frac{310}{61+310} = 83.558\%$$
$$Specificity = \frac{522}{522+28} = 94.909\%$$

Therefore, we firstly see that we obtained a sensitivity of 83.558%. This means that out of all the emails in the test set which were actually spam, our model was able to detect 84% of them. This value is important as it shows that only approximately 16% percent of spam emails in our test set were not able to be detected by our model.
Another value of significance we should note is that the specificity we obtained was 94.909%. We especially want this value to be close to 100% because it represents the percentage of time that an email that wasn't spam was correctly predicted to be not spam. So if we implement our model in practice and we are deleting emails that our model detects as spam then it is really bad if we are deleting emails which are in fact 'not spam' just because our model incorrectly predicted them as spam.
We also wish to calculate the precision and negative predicted value

$$Precision = \frac{310}{28+310} = 91.716\%$$
$$Negative\ Predicted\ Value = \frac{522}{522+28} = 89.537\%$$

Thus, we see that we have a precision of 91.716%. This means that approximately 92% of all the emails in our test set which our modeled predicted as spam were actually spam.
And on the other hand we have a negative predicted value of 89.537%. This basically means that approximately 90% of all the emails in our test set which our model predicted were not spam were indeed in actuality not spam.

4.2 Area Under the ROC Curve

We now move on to plotting our receiver operating characteristic (ROC) curve in R. Recall that a model with perfect discrimination ability will have an ROC curve which is a horizontal line at 1 on the true positive rate axis and a model that is completely random will have an ROC curve which is a diagonal line from (0,0) to (1,1). We see in Figure 4.2 that our ROC curve is quite close to the top left corner of the plot which means our model is quite good.
We also note that the area under the ROC curve for our model is 0.9653639.
Therefore, the fact that the ROC curve is quite close to the top left corner of the plot and that the area under the curve is relatively close to 1 signifies that our model has a high discrimination ability in that it is able to tell whether an email is spam or not quite well.

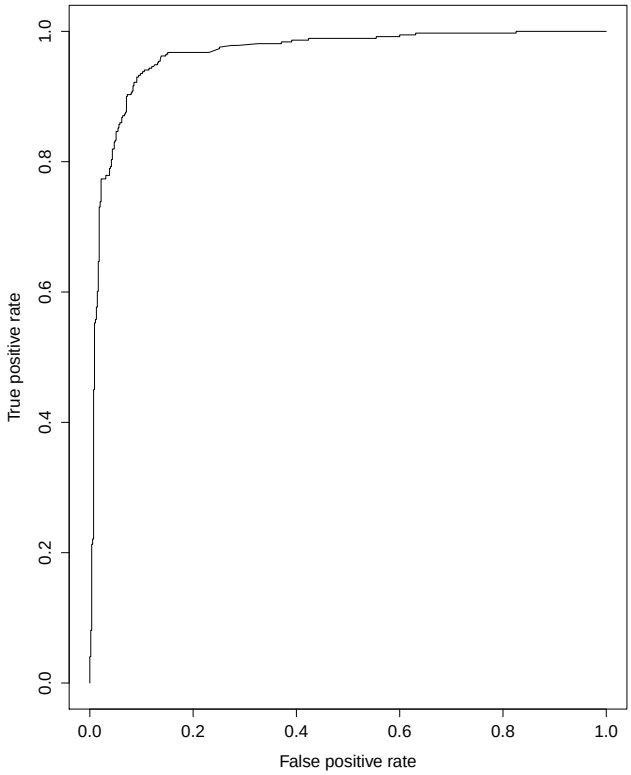

Figure 4.2: AUC plot

4.3 Interpretation of final model

Since we are using a binomial logistic regression for this model. Much of the interpretation follows suit for the amount of frequency involved in the continuous variables. We interpret the continuous variables on the given percentage of words that appear in the given email. In the final model we are able to interpret log-odds and probabilities of how likely the email could be considered spam; with the given amount of words that are frequent in the email. To give a basic interpretation for log-odds we simply input these figures in the logistic mean response function. For the logistic regression to be considered as spam, the given total must be over .50 .

4.4 Confidence interval

We obtained the following confidence intervals for each and every significant parameter in our final model. In general interpretation; the logistic regression coefficients are the given change for the log-odds for each and every one unit increase in frequency for the predictor variables. Since the data set is only a sample of one individual "George Forman" the model only takes upon the specific frequencies of words that are concluded to be spam or not spam for this individual's judgment. The word "George" itself distinguishes as non-spam with a log-odds interval between negative 21 and negative 13 because of this given matter. Another word such as edu is taken into account for this individual's back ground as an academic. Much of the frequencies are purely determined by a person's judgment and background on their consideration of what is usually spam mail or not.

4.5 Odds ratio

In the given confidence intervals for the odds ratio, the odds ratio only distinguishes on what odds only determine as spam. For basic interpretation; for every one percent increase in word frequency the odds would increase on these given intervals. On another account the continuous variables that were not considered to be spam in the previous confidence intervals have a near zero to minimal change to the overall odds in the given odds ratio. The George variable for instance has a similar effect into this because the George variable in the log-odds intervals showed a negative response determining if it is spam or not. The odds ratio only shows the true effect only taking true-spam categorization into account which is only minimal.

	Odd Ratio	Lowerbound	Upperbound
(Intercept)	0.236	0.19694	0.28217
word_freq_our	1.445	1.24557	1.69344
word_freq_remove	46.363	10.51076	136.12087
word_freq_free	3.455	2.604449	4.70396
word_freq_business	6.123	3.81714	10.20959
word_freq_font	1.108	1.01174	1.22554
word_freq_hp	0.68	0.3668	0.1179
word_freq_george	4.55E-08	6.00E-10	1.96E-06
word_freq_re	0.439	0.31358	0.59345
word_freq_edu	0.177	0.05727	0.21593
‘char_freq_¡	1.506	1.27359	1.82976
‘char_freq_$‘	6418.495	1561.42429	28907.76101
capital_run_length_longest	1.017	1.01329	1.02018

Table 4.1: Odd Ratio & Confidence Interval

5 Conclusion

The given goal was to determine what given variables would suggest an email is spam or not. The final results suggest that the words and characters that imply spam are our,remove,free,business,font,! and the dollar sign. The Greater frequency of these words in emails would categorize as spam. The other set of words that suggest as non spam are hp, george, re as response and edu. From our examination ; the non-spam words are taken in reference to a direct individual or organization as a given reference. While the set of words that imply as non-spam are words that usually demonstrate false opportunity for the individual receiving the spam mail.Spam orientated words lack direction and reference to their given meaning. In conclusion we can suggest that these results can determine some important aspects in behavior involving computer usage.

A R code

```
library(car)
setwd("~/Google Drev/DTU/5. semester CSULB/STAT 410/Final project/SpamDatabase")
spambase.data <- read.csv("~/Google Drev/DTU/5. semester CSULB/STAT 410/Final project/
    SpamDatabase/spambase.data.txt", header=FALSE)
names(spambase.data) <- c("word_freq_make","word_freq_address","word_freq_all","word_
    freq_3d","word_freq_our","word_freq_over",
                          "word_freq_remove","word_freq_internet","word_freq_order","
                              word_freq_mail","word_freq_receive",
                          "word_freq_will","word_freq_people","word_freq_report","word_
                              freq_addresses","word_freq_free",
                          "word_freq_business","word_freq_email","word_freq_you","word_
                              freq_credit","word_freq_your",
                          "word_freq_font","word_freq_000","word_freq_money","word_freq
                              _hp","word_freq_hpl","word_freq_george",
                          "word_freq_650","word_freq_lab","word_freq_labs","word_freq_
                              telnet","word_freq_857","word_freq_data",
                          "word_freq_415","word_freq_85","word_freq_technology","word_
                              freq_1999","word_freq_parts","word_freq_pm",
                          "word_freq_direct","word_freq_cs","word_freq_meeting","word_
                              freq_original","word_freq_project","word_freq_re",
                          "word_freq_edu","word_freq_table","word_freq_conference","
                              char_freq_;","char_freq_(","char_freq_[",
                          "char_freq_!","char_freq_$","char_freq_#","capital_run_length
                              _average","capital_run_length_longest",
                          "capital_run_length_total","spam")
seed <- 99
set.seed(seed)

####################
## Training set ##
####################

#80% training set / 20 % test set
spambase.data$spam <- as.factor(spambase.data$spam)
y <- spambase.data$spam
X <- spambase.data[-58]

#First step is to shuffle data
N_train <- floor(0.8*nrow(X))
train_ind <- sample(seq_len(nrow(X)), size = N_train, replace = FALSE)
X_train <- X[train_ind, ]
y_train <- spambase.data$spam[train_ind]

#Change index to be easier to acces in loop
rownames(X_train) <- 1:nrow(X_train)

###############
## Test set ##
###############

X_test <- X[-train_ind, ]
y_test <- spambase.data$spam[-train_ind]

#Prescreening
#Extract p-value from each glm for each variable
n <- 57
```

```
my.pvalue <- rep(0,n)
for( i in 1:n){
  data1 <- X_train[,i]
  spam <- y_train
  data.model <- data.frame(spam,data1)
  my.pvalue[i] <- summary(glm(spam ~ data1, data=data.model, family = "binomial"))$coef
      [8]
}

my.pvalue <- data.frame(names(spambase.data)[-58],my.pvalue)
names(my.pvalue) <- c("Attribute","P-value of beta1")

#Which variable
my.pvalue[which(my.pvalue[,2]>0.05),]

#Remove variable  which are not significant
pvalue.reduced <- my.pvalue[my.pvalue[,2]<0.05,]
pvalue.reduced.sorted <- pvalue.reduced[order(pvalue.reduced$`P-value of beta1`,
    decreasing = TRUE),]

stripchart(sort(round(pvalue.reduced.sorted$`P-value of beta1`,digits=1000)), method =
    "stack", offset = 1, at = .05, pch = 20)

########################
## Cross validation ##
########################

#Remove variable from which are not significant
X_train <- X_train[,-which(names(X_train)=="word_freq_will")]
X_train <- X_train[,-which(names(X_train)=="word_freq_parts")]
X_train <- X_train[,-which(names(X_train)=="word_freq_address")]

dim(X_train)

# VIF check
data.vif <- cbind(y_train,X_train)
fit.vif <- glm(y_train ~ ., data=data.vif, family = "binomial")
max(vif(fit.vif))

#Create matrix to hold each pvalues from first 1/10 regression pvalues to last 1/10
    regresson pvalues.
#M = How many resampling
M <- 20
N <- ncol(X_train)
p1 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p2 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p3 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p4 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p5 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p6 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p7 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p8 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p9 <- as.data.frame(matrix(NA, nrow = M, ncol=N))
p10 <- as.data.frame(matrix(NA, nrow = M, ncol=N))

#Create loop to make 20 different combination of data set

#Create data frame and change colnames. Add 1 to column because we want to storage
    beta0 as well.
#Each beta has been shifted one when removing a column like before
pvalue <- as.data.frame(matrix(nrow = 200,ncol = (N+1)))

for(i in 1:(N+1)){
```

```
  colnames(pvalue)[i] <- paste("beta",i-1,collapse = "",sep = "")
}
colnames(pvalue)[1] <- "beta0"

for( k in 1:M){
  set.seed(seed)
  train_ind <- sample(seq_len(nrow(X_train)), size = N_train, replace = FALSE)
  X_train <- X_train[train_ind, ]
  y_train <- y_train[train_ind]

  #Change index to be easier to acces in loop
  rownames(X_train) <- 1:nrow(X_train)

  #Create data frame for each 1/10 of data set
  data.train1 <- cbind(y_train[1:368],X_train[1:368,])
  data.train2 <- cbind(y_train[369:736],X_train[369:736,])
  data.train3 <- cbind(y_train[737:1104],X_train[737:1104,])
  data.train4 <- cbind(y_train[1105:1472],X_train[1105:1472,])
  data.train5 <- cbind(y_train[1473:1840],X_train[1473:1840,])
  data.train6 <- cbind(y_train[1841:2208],X_train[1841:2208,])
  data.train7 <- cbind(y_train[2209:2576],X_train[2209:2576,])
  data.train8 <- cbind(y_train[2577:2944],X_train[2577:2944,])
  data.train9 <- cbind(y_train[2945:3312],X_train[2945:3312,])
  data.train10 <- cbind(y_train[3313:3680],X_train[3313:3680,])

  #Change name because cbind is measy
  names(data.train1)[names(data.train1)=="y_train[1:368]"] <- "y_train"
  names(data.train2)[names(data.train2)=="y_train[369:736]"] <- "y_train"
  names(data.train3)[names(data.train3)=="y_train[737:1104]"] <- "y_train"
  names(data.train4)[names(data.train4)=="y_train[1105:1472]"] <- "y_train"
  names(data.train5)[names(data.train5)=="y_train[1473:1840]"] <- "y_train"
  names(data.train6)[names(data.train6)=="y_train[1841:2208]"] <- "y_train"
  names(data.train7)[names(data.train7)=="y_train[2209:2576]"] <- "y_train"
  names(data.train8)[names(data.train8)=="y_train[2577:2944]"] <- "y_train"
  names(data.train9)[names(data.train9)=="y_train[2945:3312]"] <- "y_train"
  names(data.train10)[names(data.train10)=="y_train[3313:3680]"] <- "y_train"

  #Fit models for each subset of data. Had to set max iteration to 10 to make the model
      converge.
  FI <- 10
  mod1 <- glm(y_train~. , data=data.train1, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod2 <- glm(y_train~. , data=data.train2, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod3 <- glm(y_train~. , data=data.train3, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod4 <- glm(y_train~. , data=data.train4, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod5 <- glm(y_train~. , data=data.train5, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod6 <- glm(y_train~. , data=data.train6, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod7 <- glm(y_train~. , data=data.train7, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod8 <- glm(y_train~. , data=data.train8, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod9 <- glm(y_train~. , data=data.train9, family = binomial(link = "logit"),control =
      list(maxit = FI))
  mod10 <- glm(y_train~. , data=data.train10, family = binomial(link = "logit"),control
      = list(maxit = FI))
  #Extract pvalues into vector
  for (i in 1:(N+1)){
    p1[k,i] <- summary(mod1)$coefficient[,4][i]
```

```
    p2[k,i] <- summary(mod2)$coefficient[,4][1]
    p3[k,i] <- summary(mod3)$coefficient[,4][i]
    p4[k,i] <- summary(mod4)$coefficient[,4][i]
    p5[k,i] <- summary(mod5)$coefficient[,4][i]
    p6[k,i] <- summary(mod6)$coefficient[,4][i]
    p7[k,i] <- summary(mod7)$coefficient[,4][i]
    p8[k,i] <- summary(mod8)$coefficient[,4][i]
    p9[k,i] <- summary(mod9)$coefficient[,4][i]
    p10[k,i] <- summary(mod10)$coefficient[,4][i]
  }

#Transfer pvalues from vector to matrix
  for(i in 1:(N+1)){
    pvalue[1+10*(k-1),i] <- p1[k,i]
    pvalue[2+10*(k-1),i] <- p2[k,i]
    pvalue[3+10*(k-1),i] <- p3[k,i]
    pvalue[4+10*(k-1),i] <- p4[k,i]
    pvalue[5+10*(k-1),i] <- p5[k,i]
    pvalue[6+10*(k-1),i] <- p6[k,i]
    pvalue[7+10*(k-1),i] <- p7[k,i]
    pvalue[8+10*(k-1),i] <- p8[k,i]
    pvalue[9+10*(k-1),i] <- p9[k,i]
    pvalue[10+10*(k-1),i] <- p10[k,i]
  }
}

pvalue[,5]
#Setting all pvalue below 0.05 as 1 and below as 0
pvalue <- ifelse(pvalue<=0.05,1,0)
#Setting all NA to 0.
pvalue[is.na(pvalue)] <- 0

#Format as dataframe
pvalue <- as.data.frame(pvalue)

#Total count of each variable.
colSums(pvalue)

colSums(pvalue)[order(colSums(pvalue),decreasing = TRUE)]

##########################################
## New model with count greater 100 ##
##########################################

#First resample the data again
train_ind <- sample(seq_len(nrow(X_train)), size = N_train, replace = FALSE)
X_train <- X_train[train_ind, ]
y_train <- y_train[train_ind]
data <- cbind(y_train,X_train)

#Model
mod <- glm(y_train ~ word_freq_our + word_freq_remove + word_freq_free + word_freq_
    business +
              word_freq_font + word_freq_hp + word_freq_george + word_freq_650 +  word_
                  freq_re +
              word_freq_edu + `char_freq_[` + `char_freq_!` + `char_freq_$` + capital_
                  run_length_average +
              capital_run_length_longest, data = data, family = "binomial")

#Use step function to do forward and backward selection on model
step(mod, direction = "both")

#Remove variable with pvalues above 0.05. (largest above first)
```

```
mod.final <- glm(formula = y_train ~ word_freq_our + word_freq_remove + word_freq_free
    +
                 word_freq_business + word_freq_font + word_freq_hp + word_freq_george +
                      word_freq_re +
                 word_freq_edu + `char_freq_!` + `char_freq_$` + capital_run_length_
                     longest,
                 family = "binomial", data = data)
summary(mod.final1)

#Check vif
max(vif(mod.final))
#None above 10

######################
## Prediction part ##
######################

#Accuracy
test.set <- cbind(y_test,X_test)
summary(mod.final)
fitted.mod <- predict(mod.final, newdata = subset(test.set,select = c
    (6,8,17,18,23,26,28,46,47,53,54,57)), type = "response")
fitted.y <- ifelse(fitted.mod > .5, 1, 0)
error <- mean(fitted.y != test.set$y_test)
print(paste('Accuracy = ', 1-error))

#Area under the curve
library(ROCR)
pr=prediction(fitted.mod, test.set$y_test)
prf=performance(pr, measure="tpr", x.measure="fpr")
plot(prf)

auc <- performance(pr,measure="auc")
auc@y.values[[1]]

#Confusion matrix
source("confmatplot.R")
#Confusion matrix
tr <- as.numeric(test.set$y_test)
tr[which(tr==1)] <- 0
tr[which(tr==2)] <- 1
pr <- as.numeric(fitted.y)

confmatplot(tr, pr)

summary(mod.final)

#Confidence intervals
ci <- round(confint(mod.final),5)
ci

#Odds ratio
odds.ratio <- rep(0,12)
for(i in 1:12){
  odds.ratio[i] <- round(exp(mod.final$coefficients[i+1]),10)
}

names(odds.ratio) <-c("word_freq_our","word_freq_remove","word_freq_free","word_freq_
    business",
                      "word_freq_font","word_freq_hp","word_freq_george","word_freq_re"
                          ,"word_freq_edu",
                      "`char_freq_!`","`char_freq_$` ","capital_run_length_longest ")
odds.ratio
```

```
#Odds ratio confidence interval
ci.odds.ratio <- round(exp(confint(mod.final)),10);
ci.odds.ratio
```