Logestic regression

Sameh MAGDELDIN

Monday, January 30, 2025

This kind of model is used when your dependent variable (outcome) is categorial (usually binary, binomial) not continous. The predictors is either categorical or continous.

Logistic regression is a statistical method for modeling the probability of a binary outcome. It is widely used in classification problems, such as disease prediction, fraud detection, and customer analysis. This kind of model is used when your dependent variable (outcome) is categorical (usually binary, binomial) not continuous. the predictors is either categorical or continuous. It models the probability of a binary outcome (0 or 1) using the logistic (sigmoid) function

Packages needed in this tutorial

```
library("ggplot2")#draw the sigmoid curve for logistic regression
library("ISLR")# for Smarket dataset
library("plyr")# data manipulation
library("dplyr")# data manipulation
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("caret")# for training
```

```
## Loading required package: lattice
```

```
library ("randomForest")
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library ("e1071") # for sum
library("mlr3data") # to use heart disease dataset
```

==========

##Example 1: Dummy Data; predicting smokers from lung cap data.

A dummy data represents 2 variables of happiness and a binary variable represents the happiness status (0 no, 1 yes)

Question, can we predict if the individual is a happy or not from the 2 predictor variables?

###Example1

lets generate a dummy data

```
set.seed(52)
data1 <- data.frame(
  lungcap1 = rnorm(1500),
  lungcap2 = rnorm(1500),
  smoking = sample(c(0, 1), 1500, replace = TRUE)
)
```

Question, can we predict if the individual is a smoker or not from the lung capacity data values?

Explore your data

```
class(data1)
```

```
## [1] "data.frame"
```

```
dim(data1)
```

```
## [1] 1500    3
```

```
str(data1)
```

```
## 'data.frame':    1500 obs. of  3 variables:
##  $ lungcap1: num  -0.986 -0.395 1.309 0.955 -2.133 ...
##  $ lungcap2: num  -2.256 2.46 0.597 -1.189 -0.163 ...
##  $ smoking : num  1 1 0 1 0 0 1 1 0 1 ...
```

```r
summary(data1)
```

```
##     lungcap1           lungcap2            smoking
##  Min.   :-3.70032   Min.   :-3.166177   Min.   :0.0000
##  1st Qu.:-0.73361   1st Qu.:-0.677024   1st Qu.:0.0000
##  Median :-0.06936   Median :-0.009372   Median :0.0000
##  Mean   :-0.04952   Mean   : 0.002187   Mean   :0.4913
##  3rd Qu.: 0.64841   3rd Qu.: 0.644618   3rd Qu.:1.0000
##  Max.   : 3.84382   Max.   : 3.903227   Max.   :1.0000
```

Fitting the Logistic Regression Model

```r
model1 <- glm(smoking ~ lungcap1 + lungcap2, data = data1, family = binomial)
```

code explanation

#glm():This is the function used to fit Generalized Linear Models (GLMs) in R.

#GLMs are an extension of linear models that allow for different types of #response variables (e.g., binary, count, etc.).

#smoking ~ lungcap1 + lungcap2:This is the formula specifying the model.

#smoking is the response variable (dependent variable). It should be binary #(0 or 1) for logistic regression.

#lungcap1 and lungcap2 are the predictor variables (independent variables).

#data = data1: This specifies the dataset (data1) containing the variables #smoking, lungcap1, and lungcap2.

#family = binomial: This specifies the type of GLM to fit.

#For logistic regression, you use family = binomial because the response variable #is binary (e.g., smoker = 1, non-smoker = 0).

```r
summary(model1)
```

```
##
## Call:
## glm(formula = smoking ~ lungcap1 + lungcap2, family = binomial,
##     data = data1)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.03075    0.05176  -0.594    0.552
## lungcap1     0.07856    0.05138   1.529    0.126
## lungcap2    -0.04331    0.05296  -0.818    0.413
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2079  on 1499  degrees of freedom
## Residual deviance: 2076  on 1497  degrees of freedom
## AIC: 2082
##
## Number of Fisher Scoring iterations: 3
```

It seems that both variables care not significant. can not prodict the outcome..

I cant continue...

Lets move to another example

###2. Example2

read the data

```
data2 <- read.csv("C:/Users/magde/Desktop/logdata.csv")
```

The 2 variables smile scale and ensu. index were measured for 2447 individuals, the outcome (status) represent the happiness status (0=sad; 1 is happy)

can we fit a logistic model to predict happiness?

Explore your data

```
class(data2)
```

```
## [1] "data.frame"
```

```
dim(data2)
```

```
## [1] 2447    3
```

```
str(data2)
```

```
## 'data.frame':    2447 obs. of  3 variables:
##  $ status    : int  0 0 0 1 0 1 0 0 0 0 ...
##  $ smilescale: num  0.44 0.77 2.56 1.07 1.13 ...
##  $ ensuindex : num  -0.44493 0.34528 -0.00484 0.40637 1.7142 ...
```

fitting the model

```
model2<- glm(status  ~ smilescale + ensuindex, data = data2, family = binomial)
```

lets got the summary as usual

```
summary(model2)
```

```
##
## Call:
## glm(formula = status ~ smilescale + ensuindex, family = binomial,
##     data = data2)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.00067    0.09336  -21.43   <2e-16 ***
## smilescale   1.46851    0.06774   21.68   <2e-16 ***
## ensuindex   -0.54200    0.05226  -10.37   <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: 3314.5  on 2446  degrees of freedom
## Residual deviance: 2500.5  on 2444  degrees of freedom
## AIC: 2506.5
##
## Number of Fisher Scoring iterations: 5
```

It looks like both variables impacting the outcome significantly...

Tutors note

1. **Null Deviance**: This measures how well the response variable is predicted by a model that includes only the intercept (mean of the response variable). In this case, the null deviance is 3314.5 which means there's a lot of variation in the data that the he intercept-only model doesn't capture.
2. **Residual Deviance**: This measures how well the response variable is predicted by the model you have fitted. A lower residual deviance indicates a better fit. Here, the residual deviance is 2500.5
3. **Degrees of Freedom**: These are the number of values in the final calculation of a statistic.
4. **AIC (Akaike Information Criterion)**: This is a measure of the relative quality of a statistical model for a given set of data. It balances the goodness of fit of the model with the complexity of the model. A lower AIC indicates a better model. In this case, the AIC is 2506.5.
5. **Number of Fisher Scoring iterations:** 5 means that the model has been updated in 5 iterations to fit (fewer is better)

If the number of iterations is very high (e.g., 50 or more), it might indicate that the model is struggling to converge, possibly due to issues like multicollinearity, poorly scaled predictors, or a poorly specified model.

Predicting and Evaluating the Model

```
predictions <- predict(model2, type = "response")
```

type = "response"; ensures that the predictions are returned as probabilities

```
head(predictions)
```

```
##         1         2         3         4         5         6
## 0.2471135 0.2578229 0.8531412 0.3432464 0.2190205 0.8827950
```

As u can see, the model calculates the probability (of happiness) and return a value between 0-1.

##Now, i need to convert predicted probabilities(from a logistic regression model) ##into binary class labels (0 or 1). Let's break it down step by step:

I'll assume that

If the probability is greater than 0.5, it assigns 1 (positive class). If the probability is less than or equal to 0.5, it assigns 0 (negative class).

##This means:

If the predicted probability is > 0.5, the observation is classified as the positive class (1- happy).

If the predicted probability is < 0.5, the observation is classified as the negative class (0-sad).

```
data2$predicted_class <- ifelse(predictions > 0.5, 1, 0)
```

The **ifelse** function takes three arguments: 1. a condition 2. a value to return if condition is true 3. a value to return if condition of false

It checks each element of predictions:

If the probability is greater than 0.5, it assigns 1 (positive class).

If the probability is less than or equal to 0.5, it assigns 0 (negative class).

note that i inserted a new variable called predicted_class as well.

now lets call the data2 again

```
head(data2, n=10)
```

```
##    status smilescale    ensuindex predicted_class
## 1       0  0.4395244 -0.444932544               0
## 2       0  0.7698225  0.345284187               0
## 3       0  2.5587083 -0.004844437               1
## 4       1  1.0705084  0.406366471               0
## 5       0  1.1292877  1.714198526               0
## 6       1  2.7150650 -0.060386554               1
## 7       0  1.4609162 -0.280702268               1
## 8       0 -0.2650612  0.485414461               0
## 9       0  0.3131471 -0.049344530               0
## 10      0  0.5543380  0.627765062               0
```

Tutors note

#When to Adjust the Threshold

The threshold of 0.5 is a default, but it may not always be optimal. You might want to adjust it based on:

#Class Imbalance:

If one class is much more frequent than the other, a threshold other than 0.5 might improve performance. 'in previous example„saddness encounter maybe 80% over happiness in egyptian population, 0.5 might lead to poor performance for the minority class.'

#Cost of Misclassification:

If false positives (e.g., predicting 1 when the true class is 0) are more costly than false negatives, you might use a higher threshold (e.g., 0.7).

If false negatives are more costly, you might use a lower threshold (e.g., 0.3).

Now we need to make what is called a **confusion matrix**. simply it evalutes the model (if prediction is correct for each observation) or not

```r
conf_matrix <- table(data2$status, data2$predicted_class)
```

note that i called confusion matrix between the prediction and real status

```r
conf_matrix
```

```
##
##        0    1
##   0 1181  260
##   1  384  622
```

but what does this means??

	predicted 0	predicted 1
actual 0	TN	FP
actual 1	FN	TP

Now ill turn it into percentage (remember, biostat session of categorial data?)

```r
pro.table <- prop.table(conf_matrix) * 100
```

can u predict the happiness possiblity if u have a set of these 2 parameters on the model.....? try to build a function that gets the 2 parameters and check it with the model to predict..

#calculating accuracy and percision (Model metrics)

Accuracy: This can be calculated as ({(TN + TP)}/{(TN + FP + FN + TP)}). In this case, it would be: ({48.26318 + 25.41888}/{48.26318 + 10.62526 + 15.69268+ 25.41888}). =0.736

Precision: This is ({TP}/{(TP + FP)}), which tells you how many of the predicted positives are actually positive.

=0.70

overall impression: the model looks good!

Bonus points

use the following 2 functions to calculate accuracy and percision

calculate_accuracy <- function(true_labels, predicted_labels) { # Create a confusion matrix conf_matrix <- table(true_labels, predicted_labels)

Calculate accuracy accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)

return(accuracy) }

calculate_precision <- function(true_labels, predicted_labels) { # Create a confusion matrix conf_matrix <- table(true_labels, predicted_labels)

Calculate precision precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])

return(precision) }

###Example3

In this example, ill use a mtcars dataset

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

The question... can we predict if the car is automatic transmission or not from the mpg (miles/gallon) and wt (weight)as predictors? (automatic transmission: am=0)

```
mtcars$am <- factor(mtcars$am)  # Convert to factor (0 = automatic, 1 = manual)
```

convert the am to factor not number

```
dim(mtcars) # get the dimensions of the dataset
```

```
## [1] 32 11
```

here,ill split my data into traning and validation set

ill do 80-20

Split data (80% training, 20% testing)

```
set.seed(123)
train_index <- createDataPartition(mtcars$am, p = 0.8, list = FALSE)
```

1.createDataPartition() from the caret package creates stratified random splits of your data **(in caret package)**

2.It preserves the class distribution of your outcome variable (am in this case)

3.mtcars$am: The outcome/target variable i'm trying to predict

4.p = 0.8: Specifies 80% of data for training, 20% for testing

5.list = FALSE: Returns indices as a matrix instead of a list

```
train_data <- mtcars[train_index, ]
test_data <- mtcars[-train_index, ]
```

lets see

```
dim(train_data)
```

```
## [1] 27 11
```

```
dim(test_data)
```

```
## [1]  5 11
```

Logistic regression model (using mpg and wt as predictors)

```
model <- glm(am ~ mpg + wt, data = train_data, family = binomial)
#fitting the model, as usual
```

```
summary(model)
```

```
##
## Call:
## glm(formula = am ~ mpg + wt, family = binomial, data = train_data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  26.9381    13.4376   2.005   0.0450 *
```

```
## mpg          -0.3891     0.2899  -1.342   0.1795
## wt           -6.3527     2.6785  -2.372   0.0177 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36.499  on 26  degrees of freedom
## Residual deviance: 15.864  on 24  degrees of freedom
## AIC: 21.864
##
## Number of Fisher Scoring iterations: 6
```

#cross validation

```
ctrl <- trainControl(method = "cv", number = 5, savePredictions = TRUE)
```

method = "cv": tands for cross-validation

n=5: Indicates the number of folds used in the cross-validation process.

#In this example, it uses 5-fold cross-validation, which means the data is split #into 5 parts, and the model is trained and tested 5 times, each time using a #different part as the validation set and the remaining parts as the training #set.

#savePredictions = TRUE: as it says to save the preidections

Cross-validation (5-fold)

```
cv_model <- train(am ~ mpg + wt, data = train_data, method = "glm",
                  family = binomial, trControl = ctrl)
```

Tutors' note: you might get this warning if u did not convert your outcome as factor

"Warning: You are trying to do regression and your outcome only has two possible values Are you trying to do classification? If so, use a 2 level factor as your outcome column.Warning: glm.fit: algorithm did not convergeWarning: glm.fit: fitted probabilities numerically 0 or 1 occurred"

lets convert it

```
train_data$am <- as.factor(train_data$am)
```

rerun again the code

#Now we know the model but the new part is trControl=ctrl

#trControl = ctrl: This links the trainControl object (ctrl) to the model. #This means the model will use 5-fold cross-validation while training.

Lets see the cross-validation results

```
cv_model$results
```

```
##   parameter  Accuracy     Kappa AccuracySD   KappaSD
## 1      none 0.9314286 0.8622074 0.09604421 0.1908026
```

Tutors note parameter: Some models have tuning parameters (like lambda for glmnet), but glm has no tunable parameters, so this is NA.[**ignore in glm model**] **Accuracy:** The average accuracy across the cross-validation folds. **highest is the best**

Kappa: Cohen's Kappa statistic (measures agreement beyond chance). 1=perfect,0= no agreement, less than 0 worst than chance **AccuracySD:** The standard deviation of accuracy across the folds (shows variability).*high is bad* **KappaSD:** Standard deviation of Kappa across the folds.

bonus point for GURU

family=binomial in logistic regression in **small number entery** family=poisson for count data glmnet: when u have large data entery instead of glm

other models

```
rf_model <- randomForest(am ~ mpg + wt, data = mtcars, ntree = 100)
#using random forest model
#handle missing values but less interpretable than glm
```

```
svm_model <- svm(am ~ mpg + wt, data = mtcars, kernel = "linear")
# using support vector machine
```

Predict on test set

```
test_pred <- predict(model, newdata = test_data, type = "response")
```

#in logistic regression to get probabilities.

```
test_class <- ifelse(test_pred > 0.5, 1, 0)
```

Confusion matrix (another way rather than before)

```
confusionMatrix(factor(test_class), test_data$am)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 3 0
##          1 0 2
```

```
##
##                Accuracy : 1
##                  95% CI : (0.4782, 1)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 0.07776
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.6
##          Detection Rate : 0.6
##    Detection Prevalence : 0.6
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : 0
##
```

What does this means?

#True Negatives (TN): 5 (correctly predicted class 0) #False Positives (FP): 0 (predicted 1 when true class was 0) #False Negatives (FN): 0 (predicted 0 when true class was 1) #True Positives (TP): 1 (correctly predicted class 1) #Mcnemar's Test P-Value: Tests if FP and FN are equally likely #in this example nothing to compare

##The p-value (0.3349) suggests the model's performance is not significant.. ##we cant predict am (outcome) from both predictors##

##Homework1##

In this presentation, i will try to predict the stock direction (which is a categorial; up or down) using logestic regression.

lets start!

```
library("ISLR")
attach(Smarket)
```

Figure out your data

```
str(Smarket)
```

```
## 'data.frame':    1250 obs. of  9 variables:
##  $ Year     : num  2001 2001 2001 2001 2001 ...
##  $ Lag1     : num  0.381 0.959 1.032 -0.623 0.614 ...
##  $ Lag2     : num  -0.192 0.381 0.959 1.032 -0.623 ...
##  $ Lag3     : num  -2.624 -0.192 0.381 0.959 1.032 ...
##  $ Lag4     : num  -1.055 -2.624 -0.192 0.381 0.959 ...
##  $ Lag5     : num  5.01 -1.055 -2.624 -0.192 0.381 ...
##  $ Volume   : num  1.19 1.3 1.41 1.28 1.21 ...
##  $ Today    : num  0.959 1.032 -0.623 0.614 0.213 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 2 2 1 2 2 2 1 2 2 2 ...
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2                Lag3          
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000  
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000  
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500  
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716  
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750  
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000  
##       Lag4                Lag5              Volume           Today          
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000  
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500  
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500  
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138  
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750  
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000  
##  Direction
##  Down:602
##  Up  :648
##
##
##
##
```

as you see, direction is a categorial variable.

first i will generate a correlation matrix to have a look on the data. note that i need to exclude the direction variable because it is catgorial (can not do correlation with that!)

I will create another dataset named Smarket1 without Direction column

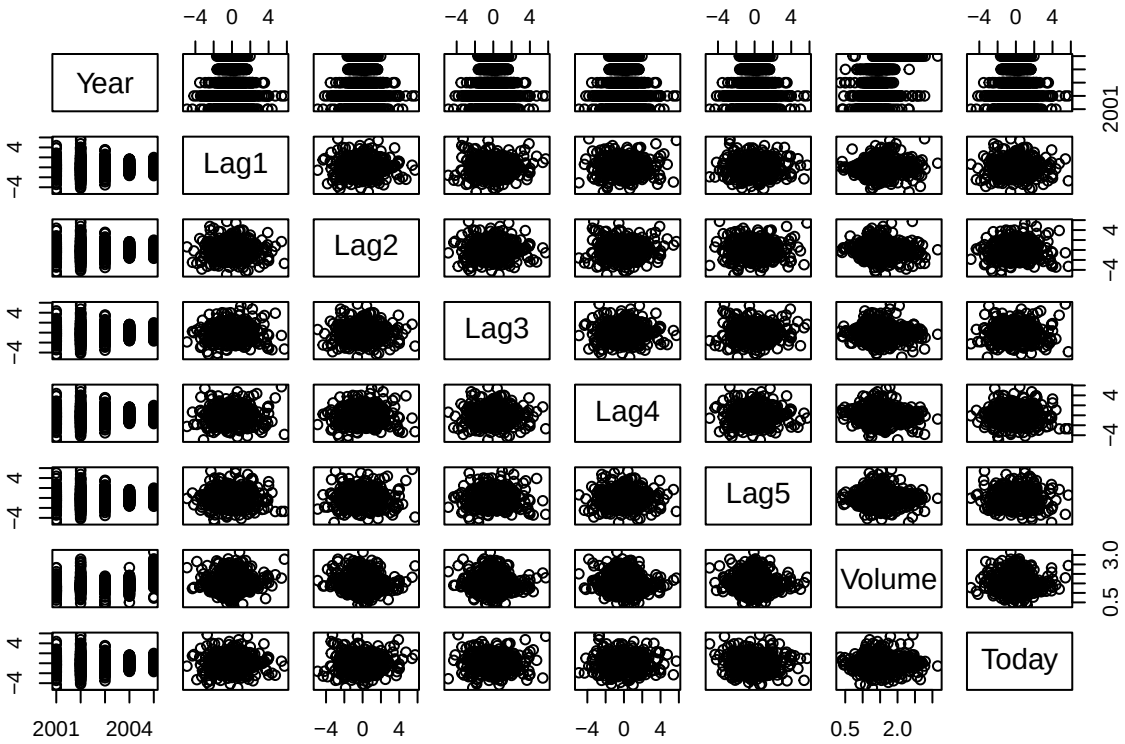
```
cor(Smarket[,-9])# i excluded the direction column
```

```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

as you can see there is no strong correlation between lags, if there is a one we would be rich.

lets visiualize the data

```
pairs(Smarket[,-9])
```



the Smarket data includes some observations from 2001-2005 [type: unique(Smarket$Year)]

we will separate the data into training (2001-2004 and testing data 2005)

or i can use the code stated earlier!

```
training <- (Smarket$Year<2005)
testing <- !training

training_data<- filter(Smarket,Smarket$Year<2005)
testing_data <- filter(Smarket,Smarket$Year==2005)
```

selecting only direction column from testing

```
direction <- testing_data$Direction
```

Fit our logistic regression model

```
log_model <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=training_data,
                 family=binomial)
#glm stands for generalized linear model
```

get a summary

```
summary(log_model)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = training_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.191213   0.333690   0.573    0.567
## Lag1        -0.054178   0.051785  -1.046    0.295
## Lag2        -0.045805   0.051797  -0.884    0.377
## Lag3         0.007200   0.051644   0.139    0.889
## Lag4         0.006441   0.051706   0.125    0.901
## Lag5        -0.004223   0.051138  -0.083    0.934
## Volume      -0.116257   0.239618  -0.485    0.628
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.3  on 997  degrees of freedom
## Residual deviance: 1381.1  on 991  degrees of freedom
## AIC: 1395.1
##
## Number of Fisher Scoring iterations: 3
```

eariler, we could not see a strong correlation thats why also p value here is not that significant

lets use this fitted model to do prediction "for probability" of the testing data (2005)

```
predicted_prob <- predict(log_model,testing_data,type="response")
```

I'll use another approach here„take care

the predicted probabilities needed to be changed again to class (up or down)

Ok , we have 252 directions (to be tested)

```
log_model_direction <- rep("Down",252)# repeat doum 252 times then change the
#value to up when it probability meets 0.05 or less
```

```
log_model_direction[predicted_prob>0.5]="Up"
# missclassification rate cutoff is 0.5
```

Create a confusion matrix to check accuracy of our fitted model

```
conf_matrix5 <- table(log_model_direction,direction)
```

tabulate direction with our predict the result

```
conf_matrix5
```

```
##                    direction
## log_model_direction Down Up
##                Down   77 97
##                Up     34 44
```

The confusion matrix says that prediction model is BAD

to calculate missclassification

```
pro.tablex <- prop.table(conf_matrix) * 100
```

43% OF PREDICTION IS FALSE

##Homework2##

Im going to use mtcars dataset to predict VS value (very fast engine) (dependent, binomial) against independent variables (wt and disp; weight and displacement of engine)

after fitting the model , we will try to figure out the probability of vs (being a very fast engine or not) if we have a certain weight and displacement value.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

fitting a logistic model

```
model <- glm(vs~wt+disp,data=mtcars,family=binomial)
```

```
summary(model)
```

```
##
## Call:
## glm(formula = vs ~ wt + disp, family = binomial, data = mtcars)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  1.60859    2.43903   0.660    0.510
## wt           1.62635    1.49068   1.091    0.275
## disp        -0.03443    0.01536  -2.241    0.025 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.86  on 31  degrees of freedom
## Residual deviance: 21.40  on 29  degrees of freedom
## AIC: 27.4
##
## Number of Fisher Scoring iterations: 6
```

null deviance (43.86): how does the dependent variable predicted by this model without considering explanatory variables (here wt and disp) (null devience should be as low as possible)

residual deviance (21.40):after including explanatory variables. note it decreased (better after including the independent variables)

AIC;Akaike information criterion (27.4): how good the quality of the model is in comparision to other logistic models [usefull if you are comparing several logistic models , you need to choose the one with low AIC]. not used here.

Coefficients:

as you see, wt (1.6) influence the vs positively and disp (-0.03) influence the vs negatively

you can understand if there is a significant or not from probability.

intercept: is a regression taking in account no pridectors (consider x as 0), in some cases it might be non meaningful ——— More detailed

when weight (wt) increase 1 unit, the log of odds[odds are number of favour outcome to non favour outcome], log number of odd of vs=1^1.62 -> the odd here will equal E^1.62 (E is 2.17)

‘

---

now lets pridict the vs if we have an engine with weigh of 2.1 and disp of 180

```
newdata <- data.frame(wt=4.7,disp=160)
```

```
predict(model,newdata,type="response")
```

```
##         1
## 0.9768662
```

as you see, this means that the probability of an engine which have wt 4.7 and disp 160 to be a very fast car is 97%

Undertanding and getting odds ratio:

Lets begin with probability. Probabilities range between 0 and 1. Lets say that the probability of success is .8, thus

p = .8

Then the probability of failure is

q = 1 - p = .2

Odds are determined from probabilities and range between 0 and infinity. Odds are defined as the ratio of the probability of success and the probability of failure. The odds of success are

odds(success) = p/(1-p) = .8/.2 = 4

that is, the odds of success are 4 to 1. The odds of failure would be

odds(failure) = .2/.8 = .25

This looks a little strange but it is really saying that the odds of failure are 1 to 4. The odds of success and the odds of failure are just reciprocals of one another, i.e., 1/4 = .25 and 1/.25 = 4. .

```
exp(cbind(OR = coef(model), confint(model)))
```

```
## Waiting for profiling to be done...
```

```
##                    OR      2.5 %       97.5 %
## (Intercept) 4.9957752 0.04849156 1262.2610624
## wt          5.0852960 0.31758592  147.4390638
## disp        0.9661524 0.92966248    0.9895088
```

```
Odds <- exp(cbind(OR = coef(model), confint(model)))
```

```
## Waiting for profiling to be done...
```

exporting CSV table of Odds ratio

```
write.csv(Odds, file = "Odds.csv", row.names = TRUE)
```

drawing a graph data:mtcars what we need to do, predicting vs from an mpg reads

```
lreg <- glm(vs~mpg,data=mtcars,family=binomial)
```

```
lreg
```

```
##
## Call:  glm(formula = vs ~ mpg, family = binomial, data = mtcars)
##
## Coefficients:
## (Intercept)          mpg
##     -8.8331       0.4304
##
## Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
## Null Deviance:       43.86
## Residual Deviance: 25.53     AIC: 29.53
```

```
summary(lreg)
```

```
##
## Call:
## glm(formula = vs ~ mpg, family = binomial, data = mtcars)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.8331     3.1623  -2.793  0.00522 **
## mpg           0.4304     0.1584   2.717  0.00659 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.860  on 31  degrees of freedom
## Residual deviance: 25.533  on 30  degrees of freedom
## AIC: 29.533
##
## Number of Fisher Scoring iterations: 6
```

```
par(mfrow=c(2,2))
ggplot(lreg, aes(x=mpg, y=vs)) + geom_point() + stat_smooth(method="glm",
       method.args = list(family = "binomial"), se=FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```