Week 8 - Forecasting with Logistic Regression

# Introduction

Logistic regression extends the ideas of linear regression to the situation where the outcome variable, Y, is categorical. Logistic regression can be used for classifying a new record, where its class is unknown.

Logistic regression can be seen as a special case of generalized linear regression model and thus analogous to linear regression. The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression. In particular the key differences of these two models can be seen in the following two features of logistic regression

* First, the conditional distribution  is a Bernoulli distribution rather than a Gaussian distribution, because the dependent variable is binary.
* Second, the predicted values are probabilities and are therefore restricted to (0,1) through the logistic distribution function because logistic regression predicts the probability of particular outcomes.

In this section, we focus on the use of logistic regression for forecasting. We deal only with binary outcome variable having two possible classes. Popular example of binary outcomes are success/failure, yes/no, or similar. The following situation may require forecasting binary outcome:

* Whether or not an event will occur
* Whether a numerical measurement will go up or down
* Whether or not a numerical measurement will crosses a threshold

In some cases other situations we may choose to convert a continuous outcome variable or an outcome variable with multiple classes into a binary outcome variable to match the requirements.

# Naïve Forecast and Performance Evaluation

As in numerical forecasting, a naïve forecast can simply be the binary value in the previous period. An alternative naïve benchmark is the “majority-class” forecast. Which is the most popular outcome in the training period: event or non-event. When more than 50% of the training period outcomes are “event”, then a naïve majority-class forecast is “event”. When fewer than 50% of the training period outcomes are “event”, then the majority-class forecast is “non-event”

The performance of a binary outcome forecasting method is evaluated by comparing the binary forecast to the actual binary events. We will build a confusion matrix the way we did for classification methods outcome. Below is the format of such a matrix:

|  |  |  |
| --- | --- | --- |
|  | Forecasted events | Forecasted non-event |
| Actual events | # correctly forecasted events | # missed events |
| Actual non-events | # missed non-events | # correctly forecasted non-events |

1. The Logistic Regression Model

Why don’t we use the linear regression on a binary outcome? If we apply linear regression, we get the outcome and we get the performance measure evaluate the outcome! But you will see, if we apply linear regression on binary data, we will get very strange values, we won’t necessarily get zero or one, we don’t even get values between zero and one! That is why we don’t use linear regression for forecasting probability or binary event. That is why logistic regression comes in.

# Logistic Regression model

You remember this methods for classification from our CSDA 5330. Here, I will repeat the mathematics behind the method to refresh your memory.

The idea behind the logistic regression is straightforward. Instead of using *Y* directly as the outcome variable, we use a function of it, which is called ***logit.*** The logit, it turns out, can be modeled as a linear function of predictors. Once the logit has been predicted, it can be mapped back to a probability.

To understand the logit, we take several intermediate steps. These steps are:

* First, we look at the *p* = P(Y=1), the probability of belonging to class 1 (as opposed to class 0). In forecasting we can use “event” and “non-event” class. In contrast to the binary variable Y, which only takes the value 0 and 1, *p* (“event”) can take any value in the interval of [0,1]. However,, if we express *p* as a linear function of k predictors in the form

It is not guaranteed that the right-hand side will lead to values within the interval [0, 1] as I explain at the beginning of this section.

The solution is to use a nonlinear function of the predictors in the form

This is called the logistic response function, for any values of X1, X2, …, Xk , the right-hand side will be always lead to values in the [0, 1] interval.

* Next, we look at the a different measures of belonging a certain class, knowns as **Odds**. The odds of belonging to class 1 (“event”) are defined as the ratio of the probability of belonging to class 1 to the *probability of belonging to class 0*.

To explain better the meaning of Odds, let’s consider this scenario: Instead of talking about the *probability* of contracting a disease, people talking about the odds of contracting the disease. So, if the probability of contracting a disease is 35% the odds of contracting the disease is which is about 54%.

We can use the Odds to calculate the probability as follow:

Let’s use the above to calculate the odds based on the *p* function

Let’s assume Q = then we can write the Odds function as:

Which is turn is:

Substituting Q we will have:

And the log in base ***e*** of Odds is the regression function!

We can see also, the log(Odds) is **logit**

# Lagged Predictors

When the binary data is the result of thresholding a continuous measurement, consider including lagged versions of the actual measurements rather than the binary values. For example, include the previous value rather than whether it exceeded the threshold. Passed(previous) actual measurements (observations) might be more predictive.

# Generating Binary Forecast

Once we estimated the model of our ***logit*** function, we obtain the predicted (forecasted) probability or odds for each time period of interest. Then, a binary forecast of “event” or “non-event” is obtained by comparing the forecasted probability to 0.5, equivalently, by comparing the forecasted odds to 1. A higher value leads to an “event” forecast, while a lower value leads to a “non-event” forecast. Playing with the cutoff value changes the fitting of the model. Remember we should avoid overfitting!

# Forecasting Rain Fall in Melbourne AUS.

Textbook’s example on page 183, is an example of using logistic regression to forecast the “event = Rainy” and “non-event = not Rainy”

The data set has record of rain amount on each day from 1/1/2000 until 10/31/2011, for 4322 days. How can we use this data to forecast if 11/1/2011 rains or not. Basically, if we consider the day with zero amount of rain fall as “non-event” and “event” otherwise, then we have a binary forecast case!

Based on the visualization of data (page 184) and noticing seasonality. We will use also lag- 1 rainfall values for capture observation correlation. This is done by introducing dummy predictor Rainyt-1. Note, we do not differencing. We just use the t-1 observation (Yt-1) to forecast the Ft

The forecast function is:

The following R codes show how this function is built and used in logistic regression to forecast.

setwd("Z:/CSDA 5410 Time Series Analytics/Week 8/Data and R-Codes")

library(readxl)

MRF.data<-read\_xls("MelbourneRainfall.xls")

#Set the date format to MM/DD/YY

library(caret)

MRF.data$Date<-as.Date(MRF.data$Date, format="%m/%d/%Y")

#Seting the binary data

MRF.data$Rainy<-ifelse(MRF.data$`Rainfall amount (millimetres)`> 0, 1, 0)

#specify time periods

nPeriods <- length(MRF.data$Rainy)

nPeriods

[1] 4322

The following is a sample of the data set after adding “Rainy” column

|  |  |  |
| --- | --- | --- |
| Date | Rainfall amount (millimeters) | Rainy |
| 1/1/00 | 0.4 | 1 |
| 1/2/00 | 0 | 0 |
| 1/3/00 | 0 | 0 |
| 1/4/00 | 3.4 | 1 |
| 1/5/00 | 1.4 | 1 |
| 1/6/00 | 0 | 0 |
| 1/7/00 | 0 | 0 |
| 1/8/00 | 0 | 0 |
| 1/9/00 | 0 | 0 |
| 1/10/00 | 2.2 | 1 |

# set the *sine* and *cosine* seasonality components of the data and uses the lag 1, because the function uses the Yt-1  to calculate Ft.

#we add the required columns to our time series data.

MRF.data$Lag1 <- c(NA, MRF.data$`Rainfall amount (millimetres)`[1:(nPeriods-1)])

MRF.data$t <- seq(1, nPeriods, 1)

MRF.data$Seasonal\_sine = sin(2 \* pi \* MRF.data$t / 365.25)

MRF.data$Seasonal\_cosine = cos(2 \* pi \* MRF.data$t / 365.25)

#select records from 1/1/2000 until 12/31/2009 as training dataset. Then we remove the first row since we use lag-1

train.data <- MRF.data[MRF.data$Date <= as.Date("12/31/2009", format="%m/%d/%Y"), ]

write.csv(train.data, "train\_data\_1.csv")

#removing the first record (lag 1)

train.data<-train.data[-1,]. ## removing the first row

write.csv(train.data, "train\_data\_2.csv")

The following is a sample of the data set prepared for model building (first row is removed)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Rainfall amount (millimetres) | Rainy | Lag1 | t | Seasonal\_sine | Seasonal\_cosine |
| 1/2/00 | 0 | 0 | 0.4 | 2 | 0.03439806 | 0.99940821 |
| 1/3/00 | 0 | 0 | 0 | 3 | 0.05158437 | 0.99866864 |
| 1/4/00 | 3.4 | 1 | 0 | 4 | 0.06875541 | 0.99763355 |
| 1/5/00 | 1.4 | 1 | 3.4 | 5 | 0.0859061 | 0.99630324 |
| 1/6/00 | 0 | 0 | 1.4 | 6 | 0.10303138 | 0.99467811 |
| 1/7/00 | 0 | 0 | 0 | 7 | 0.12012617 | 0.99275863 |
| 1/8/00 | 0 | 0 | 0 | 8 | 0.1371854 | 0.99054539 |
| 1/9/00 | 0 | 0 | 0 | 9 | 0.15420405 | 0.98803902 |
| 1/10/00 | 2.2 | 1 | 0 | 10 | 0.17117706 | 0.98524028 |
| 1/11/00 | 0 | 0 | 2.2 | 11 | 0.18809942 | 0.98214999 |

#validation set: all dates greater than 12/31/2009

valid.data<- MRF.data[MRF.data$Date > as.Date("12/31/2009", format="%m/%d/%Y"), ]

#get only the required columns

xvalid <- valid.data[, c(4,6,7)] ## we don’t need t column since it is also used to calculate SINE and COSINE

# run the logistic method and build the forecast model: Rainy is the target, lag-1 is the trend, sin and cosine are seasonality

MRF.lr <- glm(Rainy ~ Lag1 + Seasonal\_sine + Seasonal\_cosine, data = train.data, family = "binomial")

#Check the fitting training data measures

summary(MRF.lr)

Call:

glm(formula = Rainy ~ Lag1 + Seasonal\_sine + Seasonal\_cosine,

family = "binomial", data = train.data)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.8566 -0.9321 -0.7514 1.3118 1.7230

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.76888 0.03858 -19.927 < 2e-16 \*\*\*

Lag1 0.11187 0.01137 9.843 < 2e-16 \*\*\*

Seasonal\_sine -0.26885 0.05049 -5.324 1.01e-07 \*\*\*

Seasonal\_cosine -0.37134 0.05067 -7.328 2.33e-13 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 4751.8 on 3651 degrees of freedom

Residual deviance: 4533.7 on 3648 degrees of freedom

(1 observation deleted due to missingness)

AIC: 4541.7

Number of Fisher Scoring iterations: 4

#predicting validation

MRF.lr.pred <- predict(MRF.lr, xvalid, type = "response")

value1<-ifelse(MRF.lr$fitted.values > 0.5, 1, 0)

length(value1)

[1] 3652

value2<-train.data$Rainy

length(value2)

[1] 3653

value2<-value2[-1]

#training data confusion matrix (fitting evaluation)

confusionMatrix(as.factor(value1), as.factor(value2), positive="1")

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 2251 1115

1 104 182

Accuracy : 0.6662

95% CI : (0.6507, 0.6815)

No Information Rate : 0.6449

P-Value [Acc > NIR] : 0.003566

Kappa : 0.1166

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.14032

Specificity : 0.95584

Pos Pred Value : 0.63636

Neg Pred Value : 0.66875

Prevalence : 0.35515

Detection Rate : 0.04984

Detection Prevalence : 0.07831

Balanced Accuracy : 0.54808

'Positive' Class : 1

#validation data confusion matrix (model evaluation)

# confusionMatrix(as.factor(ifelse(MRF.lr.pred > 0.5, 1, 0)), as.factor(valid.data$Rainy), positive="1")

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 373 220

1 21 55

Accuracy : 0.6398

95% CI : (0.6021, 0.6762)

No Information Rate : 0.5889

P-Value [Acc > NIR] : 0.004043

Kappa : 0.1647

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.20000

Specificity : 0.94670

Pos Pred Value : 0.72368

Neg Pred Value : 0.62901

Prevalence : 0.41106

Detection Rate : 0.08221

Detection Prevalence : 0.11360

Balanced Accuracy : 0.57335

'Positive' Class : 1

Plots of the training and validation

