Cross-validation in Logistic Regression – A Machine Learning Tool to check model accuracy

A preamble: Let us ruminate a typical model building endeavor. We will have a response variable Y and several predictors. Estimate the parameters of the model using the entire data. Check the adequacy of the model using the entire data. This is like ‘You be the judge. You be the jury.’ In order to avoid the perception of bias, researchers have come up with several methods of validating the model. The focus now is on this.

Let us work in the environment of binary response variables and logistic regression models. Our Sepsis data is good to start with. Download the data from my pen drive.

> Sepsis <- read.delim("clipboard")

> dim(Sepsis)

[1] 106 6

> head(Sepsis)

Shock Malnutrition Alcoholism Age Infarction Outcome

1 0 0 0 56 0 0

2 0 0 0 80 0 0

3 0 0 0 61 0 0

4 0 0 0 26 0 0

5 0 0 0 53 0 0

6 0 1 0 87 0 1

Method 1: Select some percentage, say 80, of the data randomly and deem it as training data. Download and activate the ‘caret’ package. The function ‘createDataPartition’ comes under this package.

The ‘Outcome’ is the response variable. The length of this column is 106. Select 80% of the entries of the column randomly.

> train <- createDataPartition(Sepsis$Outcome, p = 0.8, list = FALSE)

Look at these entries. The output identifies the rows.

> train

Resample1

[1,] 1

[2,] 2

[3,] 3

[4,] 4

[5,] 5

[6,] 6

[7,] 7

[8,] 8

[9,] 9

[10,] 10

[11,] 11

[12,] 12

[13,] 13

[14,] 14

[15,] 15

[16,] 16

[17,] 17

[18,] 18

[19,] 19

[20,] 22

[21,] 23

[22,] 24

[23,] 25

[24,] 26

[25,] 27

[26,] 28

[27,] 31

[28,] 32

[29,] 33

[30,] 34

[31,] 36

[32,] 37

[33,] 38

[34,] 40

[35,] 41

[36,] 42

[37,] 43

[38,] 44

[39,] 46

[40,] 47

[41,] 50

[42,] 51

[43,] 53

[44,] 55

[45,] 56

[46,] 58

[47,] 59

[48,] 60

[49,] 62

[50,] 63

[51,] 65

[52,] 66

[53,] 67

[54,] 68

[55,] 69

[56,] 70

[57,] 72

[58,] 73

[59,] 74

[60,] 75

[61,] 76

[62,] 77

[63,] 78

[64,] 79

[65,] 80

[66,] 81

[67,] 82

[68,] 83

[69,] 84

[70,] 85

[71,] 87

[72,] 88

[73,] 90

[74,] 91

[75,] 92

[76,] 93

[77,] 94

[78,] 95

[79,] 96

[80,] 98

[81,] 99

[82,] 100

[83,] 104

[84,] 105

[85,] 106

Get the training data now. Pick up these rows.

> training <- Sepsis[train, ]

Set aside the ‘testing’ data.

> testing <- Sepsis[-train, ]

> dim(training)

[1] 85 6

> head(training)

Shock Malnutrition Alcoholism Age Infarction Outcome

1 0 0 0 56 0 0

2 0 0 0 80 0 0

3 0 0 0 61 0 0

4 0 0 0 26 0 0

5 0 0 0 53 0 0

6 0 1 0 87 0 1

How many patients in the training data have died after surgery? 18

> table(training$Outcome)

0 1

67 18

How many subjects in the testing data have died after surgery? 3

> table(testing$Outcome)

0 1

18 3

Fit the logistic regression model to the training data.

> Fit <- glm(Outcome ~ ., data = training, family = binomial)

> summary(Fit)

Call:

glm(formula = Outcome ~ ., family = binomial, data = training)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.2033 -0.4596 -0.0873 -0.0306 3.2274

Coefficients:

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

(Intercept) -9.49350 2.79077 -3.402 0.000670 \*\*\*

Shock 3.55512 1.25136 2.841 0.004497 \*\*

Malnutrition 0.98976 0.81379 1.216 0.223899

Alcoholism 3.63442 1.10178 3.299 0.000971 \*\*\*

Age 0.09130 0.03354 2.722 0.006496 \*\*

Infarction 1.91208 1.21019 1.580 0.114111

---

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 87.769 on 84 degrees of freedom

Residual deviance: 44.149 on 79 degrees of freedom

AIC: 56.149

Number of Fisher Scoring iterations: 7

Comments:

Our concern is how good this model is from prediction perspective.

Use the model to conduct prediction on the ‘testing’ data.

> Sepsis1 <- predict(Fit, newdata = testing, type = "response")

> Sepsis1

20 21 29 30 35 39

0.0012753132 0.1002100442 0.8087906108 0.0005611501 0.1837232951 0.6476007373

45 48 49 52 54 57

0.0429959232 0.1006763698 0.0774054607 0.0009700635 0.1164168257 0.0103183940

61 64 71 86 89 97

0.0784480161 0.0023789424 0.0028957423 0.2309558302 0.0004675415 0.0276742743

101 102 103

0.7923066786 0.4023319521 0.0007378222

Do hard prediction.

> HardPrediction <- ifelse(Sepsis1 >= 0.50, "Death", "Survival")

Create a data frame of observed and predicted outcomes of the ‘testing’ data.

> ObsPred <- data.frame(Observed = testing$Outcome, Predicted = HardPrediction)

> ObsPred

Observed Predicted

20 0 Survival

21 0 Survival

29 1 Death

30 0 Survival

35 0 Survival

39 0 Death

45 0 Survival

48 0 Survival

49 0 Survival

52 0 Survival

54 1 Survival

57 0 Survival

61 0 Survival

64 0 Survival

71 0 Survival

86 0 Survival

89 0 Survival

97 0 Survival

101 1 Death

102 0 Survival

103 0 Survival

Do cross-tabulation.

> table(ObsPred$Observed, ObsPred$Predicted)

Death Survival

0 1 17

1 2 1

> MisclassificationRate <- 2/21

> MisclassificationRate

[1] 0.0952381

Do you remember? When we used the entire data for model fitting and prediction, the misclassification rate was 8.5%.

The story does not end here. Repeat the process 100 times, say. We will have 100 misclassification rates. Build a histogram of these. Get a 95% confidence interval for the true misclassification rate. This is a machine learning aspect of the problem.

Method 2: k-fold cross validation

Divide the data randomly into k subgroups of almost equal size. Take k-1 subgroups and fit the model with the data from the k-1 subgroups pooled. (Each of these k-1 subgroups is called a fold. We will have k folds.) Use this model to do prediction using the left-out subgroup. Repeat this process over all possible k-1 subgroups. Let us do it. Typically, one takes k = 10.

Make ‘Outcome’ as factor.

> Sepsis$Outcome <- as.factor(Sepsis$Outcome)

Create 10 subgroups.

> CrossVal <- trainControl(method = "repeatedcv", number = 10, savePredictions = TRUE)

Fit the model each of 9 subgroups.

> Fit1 <- train(Outcome ~ ., data = Sepsis, method = "glm", family = "binomial",

+ trControl = CrossVal, tuneLength = 5)

Let us look at the output. The summary gives the fitted model using the entire data. Remember our focus is on model accuracy.

> summary(Fit1)

Call:

NULL

Deviance Residuals:

Min 1Q Median 3Q Max

-1.3277 -0.4204 -0.0781 -0.0274 3.2946

Coefficients:

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

(Intercept) -9.75391 2.54170 -3.838 0.000124 \*\*\*

Shock 3.67387 1.16481 3.154 0.001610 \*\*

Malnutrition 1.21658 0.72822 1.671 0.094798 .

Alcoholism 3.35488 0.98210 3.416 0.000635 \*\*\*

Age 0.09215 0.03032 3.039 0.002374 \*\*

Infarction 2.79759 1.16397 2.403 0.016240 \*

---

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 105.528 on 105 degrees of freedom

Residual deviance: 53.122 on 100 degrees of freedom

AIC: 65.122

Number of Fisher Scoring iterations: 7

What else is available? We are seeking information on model accuracy.

> names(Fit1)

[1] "method" "modelInfo" "modelType" "results" "pred"

[6] "bestTune" "call" "dots" "metric" "control"

[11] "finalModel" "preProcess" "trainingData" "resample" "resampledCM"

[16] "perfNames" "maximize" "yLimits" "times" "levels"

[21] "terms" "coefnames" "xlevels"

> Fit1$results

parameter Accuracy Kappa AccuracySD KappaSD

1 none 0.8860606 0.6008252 0.06032257 0.1859254

The following is the meat of 10-fold cross validation. There are 10 folds. Each left-out subgroup has 10 or 11 observations.

Take Fold 1. The left-out subgroup has 11 observations. Fit the model to Fold 1 data. Do the prediction on the left-out subgroup. Accuracy = 10/11 = 90.9%

Take Fold 2. The left-out subgroup has 12 observations. Fit the model to Fold 2 data. Do the prediction on the left-out subgroup. Accuracy = 10/12 = 83.3%

Take Fold 3. The left-out subgroup has 10 observations. Fit the model to Fold 3 data. Do the prediction on the left-out subgroup. Accuracy = 9/10 = 90.0%

And so on …

We will have 10 such accuracies. Calculate the mean and standard deviation.

Average accuracy = 88.6% SD = 0.06

Kappa is another way to measure accuracy.

Comment: This exercise sets misclassification rate at 11.4% not the rosy 8.5%.

> Fit1$pred

pred obs rowIndex parameter Resample

1 0 0 7 none Fold01.Rep1

2 0 0 17 none Fold01.Rep1

3 0 0 23 none Fold01.Rep1

4 1 1 40 none Fold01.Rep1

5 0 0 52 none Fold01.Rep1

6 0 1 54 none Fold01.Rep1

7 0 0 55 none Fold01.Rep1

8 0 0 74 none Fold01.Rep1

9 0 0 92 none Fold01.Rep1

10 0 0 95 none Fold01.Rep1

11 0 0 103 none Fold01.Rep1

12 0 0 5 none Fold02.Rep1

13 0 0 20 none Fold02.Rep1

14 0 0 31 none Fold02.Rep1

15 0 1 32 none Fold02.Rep1

16 0 0 34 none Fold02.Rep1

17 0 0 46 none Fold02.Rep1

18 0 0 48 none Fold02.Rep1

19 0 0 71 none Fold02.Rep1

20 1 1 91 none Fold02.Rep1

21 0 0 93 none Fold02.Rep1

22 0 1 100 none Fold02.Rep1

23 0 0 102 none Fold02.Rep1

24 1 1 19 none Fold03.Rep1

25 0 0 28 none Fold03.Rep1

26 0 0 30 none Fold03.Rep1

27 0 0 44 none Fold03.Rep1

28 0 0 45 none Fold03.Rep1

29 0 0 49 none Fold03.Rep1

30 0 0 53 none Fold03.Rep1

31 0 1 58 none Fold03.Rep1

32 0 0 59 none Fold03.Rep1

33 0 0 106 none Fold03.Rep1

34 0 0 18 none Fold04.Rep1

35 1 1 24 none Fold04.Rep1

36 1 1 29 none Fold04.Rep1

37 0 0 36 none Fold04.Rep1

38 0 0 37 none Fold04.Rep1

39 1 0 39 none Fold04.Rep1

40 0 0 47 none Fold04.Rep1

41 0 0 64 none Fold04.Rep1

42 0 0 86 none Fold04.Rep1

43 0 0 88 none Fold04.Rep1

44 0 0 4 none Fold05.Rep1

45 0 0 13 none Fold05.Rep1

46 0 0 14 none Fold05.Rep1

47 0 0 15 none Fold05.Rep1

48 0 0 22 none Fold05.Rep1

49 0 0 66 none Fold05.Rep1

50 1 1 75 none Fold05.Rep1

51 0 1 85 none Fold05.Rep1

52 0 0 104 none Fold05.Rep1

53 0 0 105 none Fold05.Rep1

54 0 1 6 none Fold06.Rep1

55 0 0 9 none Fold06.Rep1

56 1 1 11 none Fold06.Rep1

57 0 0 25 none Fold06.Rep1

58 0 0 27 none Fold06.Rep1

59 0 0 50 none Fold06.Rep1

60 0 0 76 none Fold06.Rep1

61 0 0 78 none Fold06.Rep1

62 0 0 82 none Fold06.Rep1

63 0 0 89 none Fold06.Rep1

64 0 0 94 none Fold06.Rep1

65 0 0 2 none Fold07.Rep1

66 0 0 10 none Fold07.Rep1

67 0 0 12 none Fold07.Rep1

68 0 0 21 none Fold07.Rep1

69 0 0 26 none Fold07.Rep1

70 0 0 41 none Fold07.Rep1

71 0 0 60 none Fold07.Rep1

72 0 0 68 none Fold07.Rep1

73 1 1 69 none Fold07.Rep1

74 1 1 87 none Fold07.Rep1

75 0 0 97 none Fold07.Rep1

76 0 0 3 none Fold08.Rep1

77 0 1 8 none Fold08.Rep1

78 0 0 35 none Fold08.Rep1

79 1 1 38 none Fold08.Rep1

80 0 0 43 none Fold08.Rep1

81 0 0 51 none Fold08.Rep1

82 1 0 56 none Fold08.Rep1

83 0 0 77 none Fold08.Rep1

84 0 0 80 none Fold08.Rep1

85 0 0 81 none Fold08.Rep1

86 0 0 1 none Fold09.Rep1

87 0 0 42 none Fold09.Rep1

88 0 0 57 none Fold09.Rep1

89 0 0 62 none Fold09.Rep1

90 1 1 65 none Fold09.Rep1

91 0 0 67 none Fold09.Rep1

92 0 0 73 none Fold09.Rep1

93 0 0 79 none Fold09.Rep1

94 0 1 83 none Fold09.Rep1

95 0 0 84 none Fold09.Rep1

96 0 0 96 none Fold09.Rep1

97 0 0 16 none Fold10.Rep1

98 0 0 33 none Fold10.Rep1

99 0 0 61 none Fold10.Rep1

100 0 0 63 none Fold10.Rep1

101 0 0 70 none Fold10.Rep1

102 1 0 72 none Fold10.Rep1

103 0 0 90 none Fold10.Rep1

104 0 1 98 none Fold10.Rep1

105 0 0 99 none Fold10.Rep1

106 1 1 101 none Fold10.Rep1

The summary of all these labors:

> Fit1$resample

Accuracy Kappa Resample

1 0.9090909 0.6206897 Fold01.Rep1

2 0.8333333 0.4285714 Fold02.Rep1

3 0.9000000 0.6153846 Fold03.Rep1

4 0.9000000 0.7368421 Fold04.Rep1

5 0.9000000 0.6153846 Fold05.Rep1

6 0.9090909 0.6206897 Fold06.Rep1

7 1.0000000 1.0000000 Fold07.Rep1

8 0.8000000 0.3750000 Fold08.Rep1

9 0.9090909 0.6206897 Fold09.Rep1

10 0.8000000 0.3750000 Fold10.Rep1

> Fit1$finalModel

Call: NULL

Coefficients:

(Intercept) Shock Malnutrition Alcoholism Age

-9.75391 3.67387 1.21658 3.35488 0.09215

Infarction

2.79759

Degrees of Freedom: 105 Total (i.e. Null); 100 Residual

Null Deviance: 105.5

Residual Deviance: 53.12 AIC: 65.12

Final word: Go ahead and use the model built on the entire data. Be warned that the accuracy of the model is 88.6% not the rosy 91.5%.

Method 3: Leave one out Cross Validation

How this works? Remove one observation. Fit the model to the data. Use the fitted model to predict the ‘Outcome’ of the left-out observation. Do it 106 times.

Let us do it.

Set the stage for LOOCV.

> TrainControl <- trainControl(method = "LOOCV")

Fit the model following the paradigm of LOOCV.

> Fit3 <- train(Outcome ~ ., data = Sepsis, trControl = TrainControl, method = "glm", family = binomial)

> summary(Fit3)

Call:

NULL

Deviance Residuals:

Min 1Q Median 3Q Max

-1.3277 -0.4204 -0.0781 -0.0274 3.2946

Coefficients:

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

(Intercept) -9.75391 2.54170 -3.838 0.000124 \*\*\*

Shock 3.67387 1.16481 3.154 0.001610 \*\*

Malnutrition 1.21658 0.72822 1.671 0.094798 .

Alcoholism 3.35488 0.98210 3.416 0.000635 \*\*\*

Age 0.09215 0.03032 3.039 0.002374 \*\*

Infarction 2.79759 1.16397 2.403 0.016240 \*

---

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 105.528 on 105 degrees of freedom

Residual deviance: 53.122 on 100 degrees of freedom

AIC: 65.122

Number of Fisher Scoring iterations: 7

Comment: Use the model based on all the observations.

Assess accuracy as per this method.

> names(Fit3)

[1] "method" "modelInfo" "modelType" "results" "pred"

[6] "bestTune" "call" "dots" "metric" "control"

[11] "finalModel" "preProcess" "trainingData" "resample" "resampledCM"

[16] "perfNames" "maximize" "yLimits" "times" "levels"

[21] "terms" "coefnames" "xlevels"

This is the crux.

> Fit3$results

parameter Accuracy Kappa

1 none 0.8584906 0.5107692

Go deeper.

> Fit3$pred

pred obs rowIndex parameter

1 0 0 1 none

2 0 0 2 none

3 0 0 3 none

4 0 0 4 none

5 0 0 5 none

6 0 1 6 none

7 0 0 7 none

8 0 1 8 none

9 0 0 9 none

10 0 0 10 none

11 1 1 11 none

12 0 0 12 none

13 0 0 13 none

14 0 0 14 none

15 0 0 15 none

16 0 0 16 none

17 0 0 17 none

18 0 0 18 none

19 1 1 19 none

20 0 0 20 none

21 0 0 21 none

22 0 0 22 none

23 0 0 23 none

24 1 1 24 none

25 0 0 25 none

26 0 0 26 none

27 0 0 27 none

28 0 0 28 none

29 1 1 29 none

30 0 0 30 none

31 0 0 31 none

32 0 1 32 none

33 0 0 33 none

34 0 0 34 none

35 0 0 35 none

36 0 0 36 none

37 0 0 37 none

38 1 1 38 none

39 1 0 39 none

40 0 1 40 none

41 0 0 41 none

42 0 0 42 none

43 0 0 43 none

44 0 0 44 none

45 0 0 45 none

46 0 0 46 none

47 0 0 47 none

48 0 0 48 none

49 0 0 49 none

50 0 0 50 none

51 0 0 51 none

52 0 0 52 none

53 0 0 53 none

54 0 1 54 none

55 0 0 55 none

56 1 0 56 none

57 0 0 57 none

58 0 1 58 none

59 0 0 59 none

60 0 0 60 none

61 0 0 61 none

62 0 0 62 none

63 0 0 63 none

64 0 0 64 none

65 1 1 65 none

66 0 0 66 none

67 0 0 67 none

68 0 0 68 none

69 1 1 69 none

70 0 0 70 none

71 0 0 71 none

72 1 0 72 none

73 0 0 73 none

74 0 0 74 none

75 1 1 75 none

76 0 0 76 none

77 0 0 77 none

78 0 0 78 none

79 0 0 79 none

80 0 0 80 none

81 0 0 81 none

82 0 0 82 none

83 0 1 83 none

84 1 0 84 none

85 0 1 85 none

86 0 0 86 none

87 1 1 87 none

88 0 0 88 none

89 0 0 89 none

90 0 0 90 none

91 1 1 91 none

92 0 0 92 none

93 0 0 93 none

94 0 0 94 none

95 0 0 95 none

96 1 0 96 none

97 0 0 97 none

98 0 1 98 none

99 0 0 99 none

100 0 1 100 none

101 1 1 101 none

102 0 0 102 none

103 0 0 103 none

104 0 0 104 none

105 0 0 105 none

106 0 0 106 none

Discuss.