Chapter 5 – Evaluating Classification & Predictive Performance

Data Mining for Business Analytics in R

Shmueli, Bruce, Yahav, Patel & Lichtendahl

Why Evaluate?

- Multiple methods are available to classify or predict
- For each method, multiple choices are available for settings
- To choose best model, need to assess each model's performance

Evaluating Predictive Performance

Measuring Predictive error

- Not the same as "goodness-of-fit"
- We want to know how well the model predicts **new data**, not how well it fits the data it was trained with
- Key component of most measures is difference between actual *y* and predicted *y* ("error")

Some measures of error

MAE or MAD: Mean absolute error (deviation)
Gives an idea of the magnitude of errors

Average error
Gives an idea of systematic over- or under-prediction

MAPE: Mean absolute percentage error

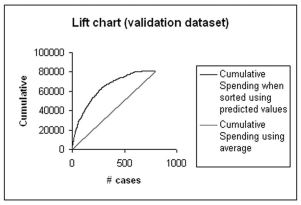
RMSE (root-mean-squared-error): Square the errors, find their average, take the square root

Total SSE: Total sum of squared errors

Lift Chart for Predictive Error

- Y axis is cumulative value of numeric target variable (e.g., revenue), instead of cumulative count of "responses"
- X axis is cumulative number of cases, sorted left to right in order of predicted value
- Benchmark is average numeric value per record, i.e. not using model

Lift chart example – spending



Accuracy Measures (Classification)

Misclassification error

- Error = classifying a record as belonging to one class when it belongs to another class.
- Error rate = percent of misclassified records out of the total records in the validation data

Naïve Rule

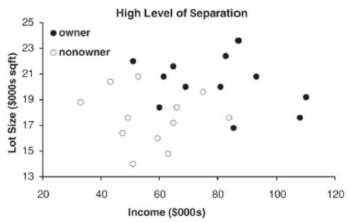
Naïve rule: classify all records as belonging to the most prevalent class

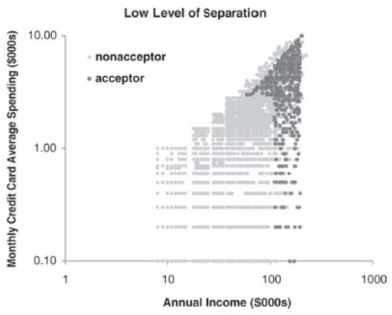
- Often used as benchmark: we hope to do better than that
- Exception: when goal is to identify high-value but rare outcomes, we may do well by doing worse than the naïve rule (see "lift" – later)

Separation of Records

"High separation of records" means that using predictor variables attains low error

"Low separation of records" means that using predictor variables does not improve much on naïve rule





Confusion Matrix

(R function: "`confusionMatrix`")

TABLE 5.2 CONFUSION MATRIX BASED ON 3000 RECORDS AND TWO CLASSES

		Actual Class 0	Actual Class 1
Predicted Class	0	2689	85
	1	25	201

201 1's correctly classified as "1"
85 1's incorrectly classified as "0"
25 0's incorrectly classified as "1"
2689 0's correctly classified as "0"

Error Rate

TABLE 5.2 CONFUSION MATRIX BASED ON 3000 RECORDS AND TWO CLASSES

		Actual Class 0	Actual Class 1
Predicted Class	0	2689	85
	1	25	201

Overall error rate = (25+85)/3000 = 3.67%
Accuracy = 1 – err = (201+2689) = 96.33%
If there are multiple classes, the error rate is:
(sum of misclassified records)/(total records)

Cutoff for classification

Most DM algorithms classify via a 2-step process:
For each record,

1. Compute **probability of belonging to class "1"**
2. Compare to cutoff value, and classify accordingly

- Default cutoff value is 0.50
  - If >= 0.50, classify as "1"
  - If < 0.50, classify as "0"
- Can use different cutoff values
- Typically, error rate is lowest for cutoff = 0.50

Cutoff Table

Actual Class	Prob. of "1"	Actual Class	Prob. of "1"
1	0.996	1	0.506
1	0.988	0	0.471
1	0.984	0	0.337
1	0.980	1	0.218
1	0.948	0	0.199
1	0.889	0	0.149
1	0.848	0	0.048
0	0.762	0	0.038
1	0.707	0	0.025
1	0.681	0	0.022
1	0.656	0	0.016
0	0.622	0	0.004

- If cutoff is 0.50: eleven records are classified as "1"
- If cutoff is 0.80: seven records are classified as "1"

Confusion Matrix for Different Cutoffs

Function `confusionMatrix` requires library `caret`

```
## cutoff = 0.6
> confusionMatrix(ifelse(owner.df$Probability>0.5,
# note: "reference" = "actual"
Confusion Matrix and Statistics

          Reference
Prediction nonowner owner
  nonowner        10     1
  owner            2    11

               Accuracy : 0.875
```

```
## cutoff = 0.25
> confusionMatrix(ifelse(owner.df$Probability>0.25,
Confusion Matrix and Statistics

          Reference
Prediction nonowner owner
  nonowner        8     1
  owner           4    11

               Accuracy : 0.7916667
```

```
## cutoff = 0.75
> confusionMatrix(ifelse(owner.df$Probability>0.75,
Confusion Matrix and Statistics

          Reference
Prediction nonowner owner
  nonowner        11     5
  owner            1     7

               Accuracy : 0.75
```

When One Class is More Important

In many cases it is more important to identify members of one class

- Tax fraud
- Credit default
- Response to promotional offer
- Detecting electronic network intrusion
- Predicting delayed flights

In such cases, we are willing to tolerate greater overall error, in return for better identifying the important class for further attention

Alternate Accuracy Measures

If "C_1" is the important class,

Sensitivity (also called "recall) = % of "C_1" class correctly classified =

$$\text{Sensitivity} = \frac{\#\text{ of samples with the event and predicted to have event}}{\#\text{ of samples having event}} = \frac{n_{1,1}}{n_{1,1}+n_{1,2}}$$

Alternate Accuracy Measures

Specificity = % of "C_0" class correctly classified

$$\text{Specificity} = \frac{\#\text{ of samples without the event and predicted so}}{\#\text{ of samples without event}} = \frac{n_{2,2}}{n_{2,1}+n_{2,2}}$$

Precision= % of predicted "C_1's" that are actually"C_1's"

Alternate Accuracy Measures

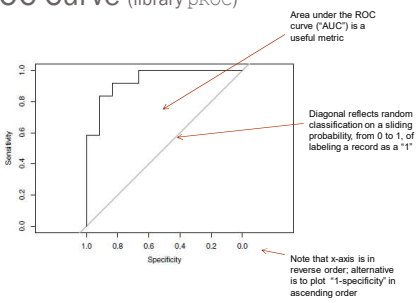
Precision= % of predicted "C_1's" that are actually"C_1's"

False Positive Rate= 1 - specificity

$$=1-\frac{n_{2,2}}{n_{2,1}+n_{2,2}}$$

$$=\frac{n_{2,1}}{n_{2,1}+n_{2,2}}$$

ROC Curve (library pROC)



Area under the ROC curve ("AUC") is a useful metric

Diagonal reflects random classification on a sliding probability, from 0 to 1, of labeling a record as a "1"

Note that x-axis is in reverse order; alternative is to plot "1-specificity" in ascending order

Lift (gains)

(separating the "wheat from the chaff")

Lift (also termed "gains"): Goal

Evaluates how well a model identifies the most important class

Helps evaluate, e.g.,

- How many tax records to examine
- How many loans to grant
- How many customers to mail offer to

Lift (gains) and Decile Charts – Cont.

Compare performance of DM model to "no model, pick randomly"

Measures ability of DM model to identify the important class, relative to the average prevalence of the class

Charts give explicit assessment of results over a large number of cutoffs

Lift and Decile Charts: How to Use

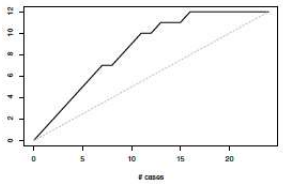
Sort records by predicted probability of belonging to the important class ("1's")

Move down the list, noting actual class

As you go, compare the number of actual 1's to the number of 1's you would expect with no model

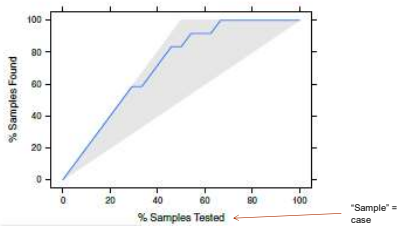
- In lift chart: compare step function to straight line
- In decile chart compare to ratio of 1

Lift (Gains) Chart – cumulative performance
(using R "Gains" package)



After examining (e.g.,) 10 cases (x-axis), 9 owners (y-axis) have been correctly identified

Lift (Gains) Chart – cumulative performance
R "caret" package uses %



After examining (e.g.,) 40% = 10 of the cases (x-axis), 75% of the owners (y-axis) have been correctly identified

Decile Chart

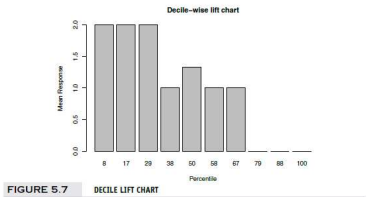


FIGURE 5.7 DECILE LIFT CHART

In "most probable" (top) decile, model is twice as likely to identify the important class compared to avg. prevalence. Percentiles do not match deciles exactly due to small sample of discrete data, with multiple records sharing same decile boundary.

Lift (Gains): How to Compute

- Using the model's output, sort records from most likely to least likely members of the important class
- Compute lift: Accumulate the correctly classified "important class" records (Y axis) and compare to number of total records (X axis)

Lift vs. Decile Charts

Both embody concept of "moving down" through the records, starting with the most probable 1's

Decile chart does this in decile chunks of data
- Y axis shows ratio of decile mean to overall mean

Lift chart shows continuous cumulative results
- Y axis shows number of important class records identified

Oversampling and Asymmetric Costs

Rare Cases

Asymmetric costs/benefits typically go hand in hand with presence of rare but important class

- Responder to mailing
- Someone who commits fraud
- Debt defaulter

- Often we oversample rare cases to give model more information to work with

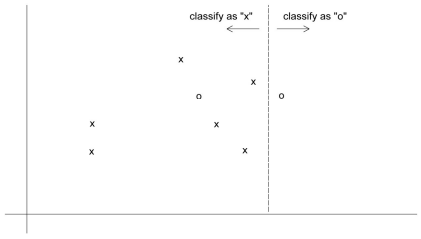
- Typically use 50% “1” and 50% “0” for training

Example

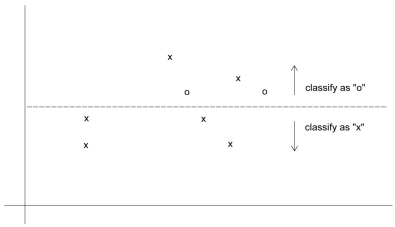
Following graphs show optimal classification under three scenarios:

- assuming equal costs of misclassification
- assuming that misclassifying “o” is five times the cost of misclassifying “x”
- Oversampling scheme allowing DM methods to incorporate asymmetric costs

Classification: equal costs

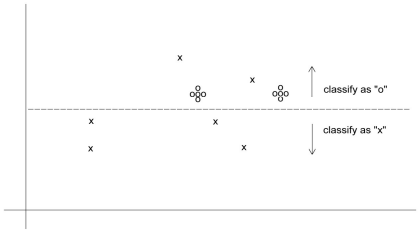


Classification: Unequal costs



Oversampling Scheme

Oversample “o” to appropriately weight misclassification costs



An Oversampling Procedure

1. Separate the responders (rare) from non-responders
2. Randomly assign half the responders to the training sample, plus equal number of non-responders
3. Remaining responders go to validation sample
4. Add non-responders to validation data, to maintain original ratio of responders to non-responders
5. Randomly take test set (if needed) from validation

Classification Using Triage

Take into account a gray area in making classification decisions

- Instead of classifying as C_1 or C_0, we classify as
  - C_1
  - C_0
  - Can't say

The third category might receive special human review

Summary

- Evaluation metrics are important for comparing across DM models, for choosing the right configuration of a specific DM model, and for comparing to the baseline ("no model")
- Major metrics: confusion matrix, error rate, predictive error
- Other metrics when
  - one class is more important
  - asymmetric costs
- When important class is rare, use oversampling
- In all cases, metrics computed from validation data