Performance Evaluation

Data Mining for Business Analytics in Python

Shmueli, Bruce, Gedeck & Patel

 rev 10/17/19

Why Evaluate?

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

Outcomes of Interest

- Numerical value

- Numerical outcome variable, e.g., house price

- Membership in a class

  - Categorical outcome variable, e.g., buyer/nonbuyer

- “Propensity” - Probability of belonging to a class

- Categorical outcome variable, e.g., the propensity to default, fail, etc.

Python Functionality Needed

```
import math
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, roc_curve, auc
import matplotlib.pylab as plt

from dmba import regressionSummary, classificationSummary
from dmba import liftChart, gainsChart
```

Measuring Predictive Error
Numerical Value

- Not the same as "goodness-of-fit" (Goodness-of-fit is a statistical measure that evaluates how well a model or statistical distribution fits observed data. It assesses the degree of agreement between the predicted values from a model and the actual observed values. In other words, it helps determine whether the model adequately describes the data.) Video

- 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," "loss")

- Predictive error metrics are called validation methods and evaluate how well a model generalizes to new data

Error metrics

Error = Actual value – Predicted value

ME (Mean Error) = The average of the errors (differences) between the predicted and actual values. Indicates bias in predictions; values close to zero suggest no consistent over- or underestimation.

RMSE (Root Mean Squared Error) = It is sensitive to outliers. The square root of the average of the squared differences between predicted and actual values. Measures the magnitude of prediction errors; lower values indicate better model accuracy.

MAE (Mean Absolute Error) = It is robust to outliers. The average of the absolute differences between predicted and actual values. Provides a straightforward measure of prediction accuracy; lower values indicate better performance.

MPE (Mean Percentage Error) = It shows the tendency to overestimate (positive values) or underestimate (negative values). The average of the percentage errors (differences) between predicted and actual values. Shows average bias in percentage terms; values close to zero indicate minimal bias.

MAPE (Mean Absolute Percentage Error) = It shows the absolute tendency to overestimate (positive values) or underestimate (negative values). The average of the absolute percentage errors between predicted and actual values. Measures prediction accuracy in percentage terms; lower values indicate better model performance.

Comparing training and validation performance

- Errors that are based on the training set tell us about model fit, whereas those that are based on the validation set (called "prediction errors") measure the model's ability to predict new data (predictive performance).
- We expect training errors to be smaller than validation errors (because the model was fitted using the training set).
- The more complex the model, the greater the likelihood that it will overfit the training data (indicated by a greater difference between the training and validation errors). In an extreme case of overfitting, the training errors would be zero (perfect fit of the model to the training data), and the validation errors would be non-zero and non-negligible. For this reason, it is important to compare the error plots and metrics (RMSE, MAE, etc.) of the training and validation sets.

Cumulative Gains and Lift Charts

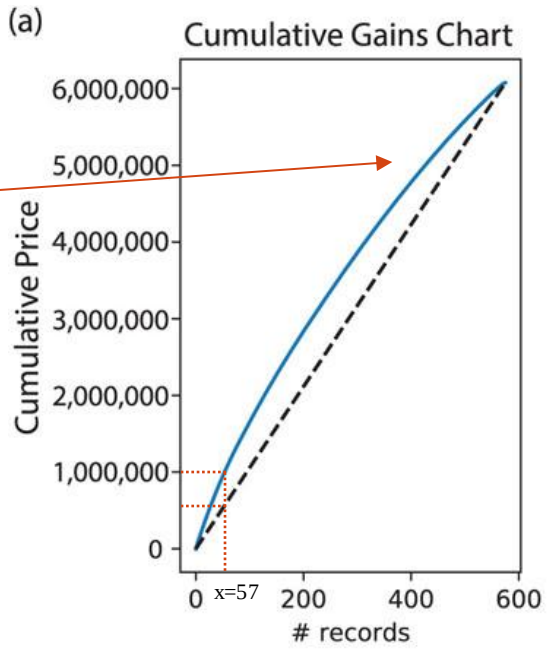
- In some applications, the goal is to search, among a set of new records, for a subset of records that gives the highest cumulative predicted values. In such cases, a graphical way to assess predictive performance is through the *cumulative gains chart* and *lift chart*. This compares the model's predictive performance to a baseline model that has no predictors. The terms *lift* and *gains* are occasionally used interchangeably. *Cumulative gains* refers to the sum of outcome values for the top cases identified by a model and *lift* to the ratio of that sum to the sum obtained through random selection.
- Cumulative gains and lift charts for a continuous response are relevant only when we are searching for a *set of records* that gives the highest cumulative predicted values. Such charts are not relevant if we are interested in predicting the outcome value for *each new record*.
- The cumulative gains and lift charts are based on ordering the set of records of interest (typically validation data) by their predicted value, from high to low, called *ranking*. Then, we accumulate the actual values and plot their cumulative value (=gains) on the y-axis as a function of the number of records accumulated (the x-axis value). This is the *cumulative gains* curve. This curve is compared to assigning a naive prediction ($\bar{y}$) to each record and accumulating these average values, which results in a diagonal line. The further away the cumulative gains curve from the diagonal benchmark line, the better the model is doing in separating records with high value outcomes from those with low value outcomes.
- The same information can be presented in a decile lift chart, where the ordered records are grouped into 10 deciles, and for each decile, the chart presents the ratio of model gains to naive benchmark gains, which is called *lift*.

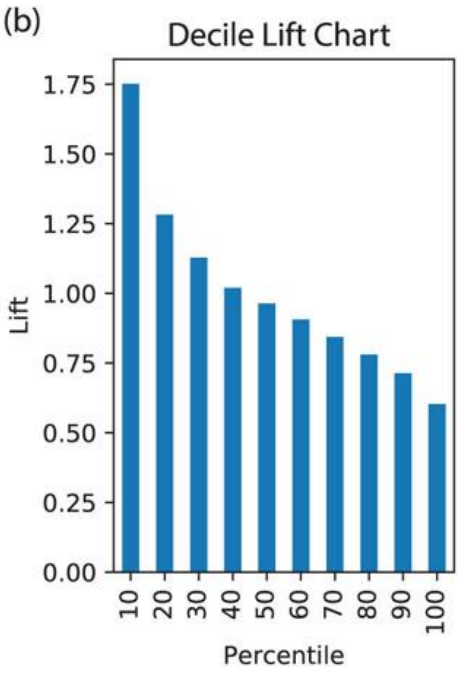
Lift Chart for Predictive Error

- 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 (ignoring all predictor information), i.e. not using model

Cumulative Gain Chart vs. Decile Lift Chart

The model`s predictive performance in terms of gains is better that the baseline model, since its cumulative gains curve is higher than that of the baseline model.





- The above figure shows a cumulative gains chart and decile lift chart based on fitting a linear regression model to the Toyota data. The charts are based on the validation data of 575 cars.
- The above figure is useful in the following scenario: Choosing the top 10% of the cars that gave the highest predicted sales, for example, we would gain 1.75 times the amount of revenue, compared to choosing 10% of the cars at random.
- This lift number can also be computed from the cumulative gains chart by comparing the sales for 57 random cars (the value of the baseline curve at x=57), which is $613,292 (=the sum of the actual sales for the 575 validation set cars divided by 10) with the actual sales of the 57 cars that have the highest predicted values (the value of the cumulative gains curve at x=57), $1,089,905. The ratio between these numbers is 1.78.
- In ”most probable” (top) decile, model is 1.75 times more likely to identify the important class (compared to random sampling).
- Lift measures how much better the model performs compared to a random selection. To calculate lift for each decile:
  Lift = (Response Rate for the Decile) / (Overall Response Rate)
- The overall response rate is calculated as the total number of positive responses divided by the total number of observations.

Accuracy Measures (Classification)

Misclassification Error

- A natural criterion for judging the performance of a classifier is the probability of making a 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

Benchmark: 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

(3000 cases)

function: "`classificationSummary`" found in the appendix of the book

- In practice, most accuracy measures are derived from the confusion matrix, also called classification matrix.
- This matrix summarizes the correct and incorrect classifications that classifier produced for a certain dataset.
- Confusion matrix is computed from the validation data.

		Actual Class	
		0	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

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

If there are multiple classes, the error rate is:
(sum of misclassified records)/(total records)

Overall error rate **(estimated misclassification rate)** = (25+85)/3000 = 3.67%

Accuracy = 1 – err = (201+2689)/3000 = 96.33%

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

Predicted Probability of Being an Owner

ACTUAL	Pred. Prob.	ACTUAL	Pred. Prob.
owner	0.9959	owner	0.5055
owner	0.9875	nonowner	0.4713
owner	0.9844	nonowner	0.3371
owner	0.9804	owner	0.2179
owner	0.9481	nonowner	0.1992
owner	0.8892	nonowner	0.1494
owner	0.8476	nonowner	0.0479
nonowner	0.7628	nonowner	0.0383
owner	0.7069	nonowner	0.0248
owner	0.6807	nonowner	0.0218
owner	0.6563	nonowner	0.0161
nonowner	0.6224	nonowner	0.0031

If cutoff is 0.5, then 11 records classified as "owner"

If 0.5 is adapted as the standard cutoff, the misclassification rate is 3/24.

Cutoff Table

Predicted Probability of Being an Owner

ACTUAL	Pred. Prob.	ACTUAL	Pred. Prob.
owner	0.9959	owner	0.5055
owner	0.9875	nonowner	0.4713
owner	0.9844	nonowner	0.3371
owner	0.9804	owner	0.2179
owner	0.9481	nonowner	0.1992
owner	0.8892	nonowner	0.1494
owner	0.8476	nonowner	0.0479
nonowner	0.7628	nonowner	0.0383
owner	0.7069	nonowner	0.0248
owner	0.6807	nonowner	0.0218
owner	0.6563	nonowner	0.0161
nonowner	0.6224	nonowner	0.0031

if cutoff is 0.8, then 7 records classified as "owner"

Confusion Matrix for Different Cutoffs

```
predicted = ['owner' if p > 0.5 else 'nonowner' for p in
     owner_df.Probability]
class_names=['nonowner', 'owner']
classificationSummary(owner_df.Class, predicted, class_names)

Confusion Matrix (Accuracy 0.8750)

              Prediction
Actual    nonowner owner
nonowner    10       2
owner        1      11
```

Cutoff = 0.25 (Accuracy 0.7917)

Actual	Prediction: nonowner	Prediction: owner
nonowner	8	4
owner	1	11

Cutoff 0.75 (Accuracy 0.7500)

Actual	Prediction: nonowner	Prediction: owner
nonowner	11	1
owner	5	7

When One Class is More Important

In some 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.

When One Class is More Important

If "C_1" is the important class,

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

Specificity = % of "C_2" class correctly classified

Precision = % of predicted "C_1's" that are actually"C_1's". It is often replaced by False Positive Rate.

Metric	Focus	Equation	Key Question	Use Case Example
Sensitivity (Recall)	Finding actual positives	TP / (TP + FN)	"How many of the actual positives were correctly identified?"	Medical tests to avoid missing positive diagnoses.
Specificity	Avoiding false positives	TN / (TN + FP)	"How many of the actual negatives were correctly identified?"	Spam filters to prevent blocking valid emails.
Precision	Accuracy of positive predictions	TP / (TP + FP)	"How many of the predicted positives were correct?"	Fraud detection to minimize false alarms.

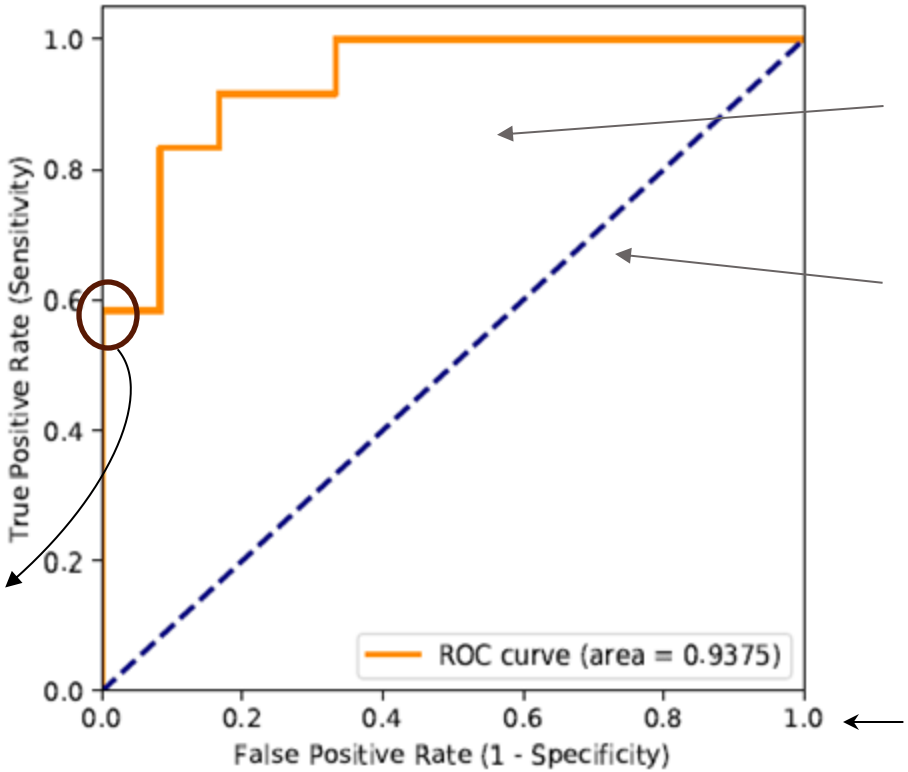
Receiver Operating Characteristic (ROC) Curve
(library `roc_curve`)

- For different thresholds, it is not doable to create many confusion matrices. Rather than being overwhelmed with confusion matrices, ROC graphs provide a simple way to summarize all of the information.

Better performance is reflected by curves that are closer to the top-left corner. The comparison curve is the diagonal, which reflects the average performance of a guessing classifier that has no information about the predictors or outcome variable. This guessing classifier guesses that a proportion α of the records is 1's and therefore assigns each record an equal probability P(Y = 1) = α.

The confusion matrix threshold represented by the new point (0.1, 0.6) correctly classified 60% of 1`s and 100% of 0`s.

Video



Area under the ROC curve ("AUC," function `auc`) is a useful metric.

Diagonal reflects random classification on a sliding probability, from 0 to 1, of labeling a record as a "1".
It shows where:

True Positive Rate

=

False Positive Rate

Any point on this line means that the proportion of correctly classified 1`s is the same as the proportion of incorrectly classified samples that are not 1.

Other softwares sometimes plot x-axis as "specificity" with 1 on left, 0 on right.

- Starting from the lower left, the ROC curve plots the pairs {sensitivity, specificity} as the cutoff value descends from 1 to 0.
- ROC term originated in WWII, applied to radar, when US sought to measure effectiveness in identifying enemy aircraft, i.e. distinguishing signal from noise.

ROC's are one way to measure a model's effectiveness in separating the "wheat from the chaff"

"**Lift**" ("**gains**") is a similar metric, but measures "how much does the model improve on random chance in finding the class of interest"

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.

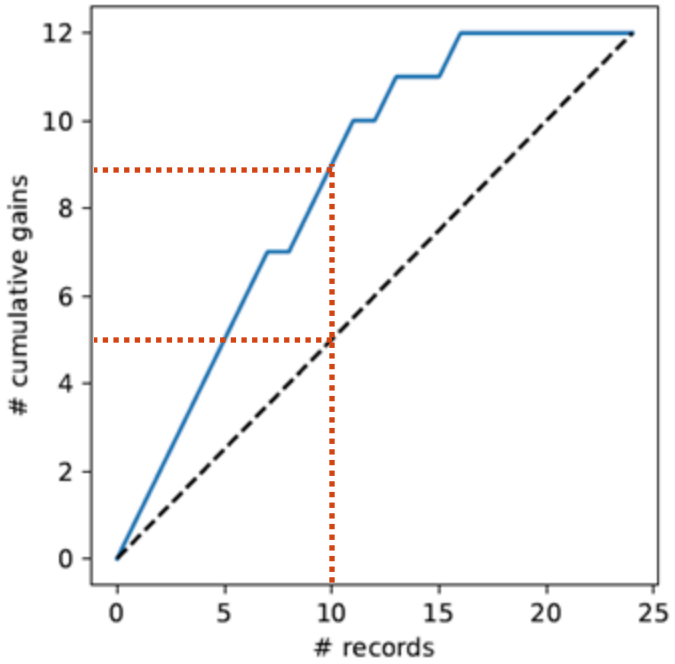
- Gain and lift are measures of the effectiveness of a classification model calculated as the ratio between the results obtained *with* and *without* the model
- Compares performance of DM model to “no model, pick randomly”
- Measures ability of DM model to identify the important class, relative to the 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

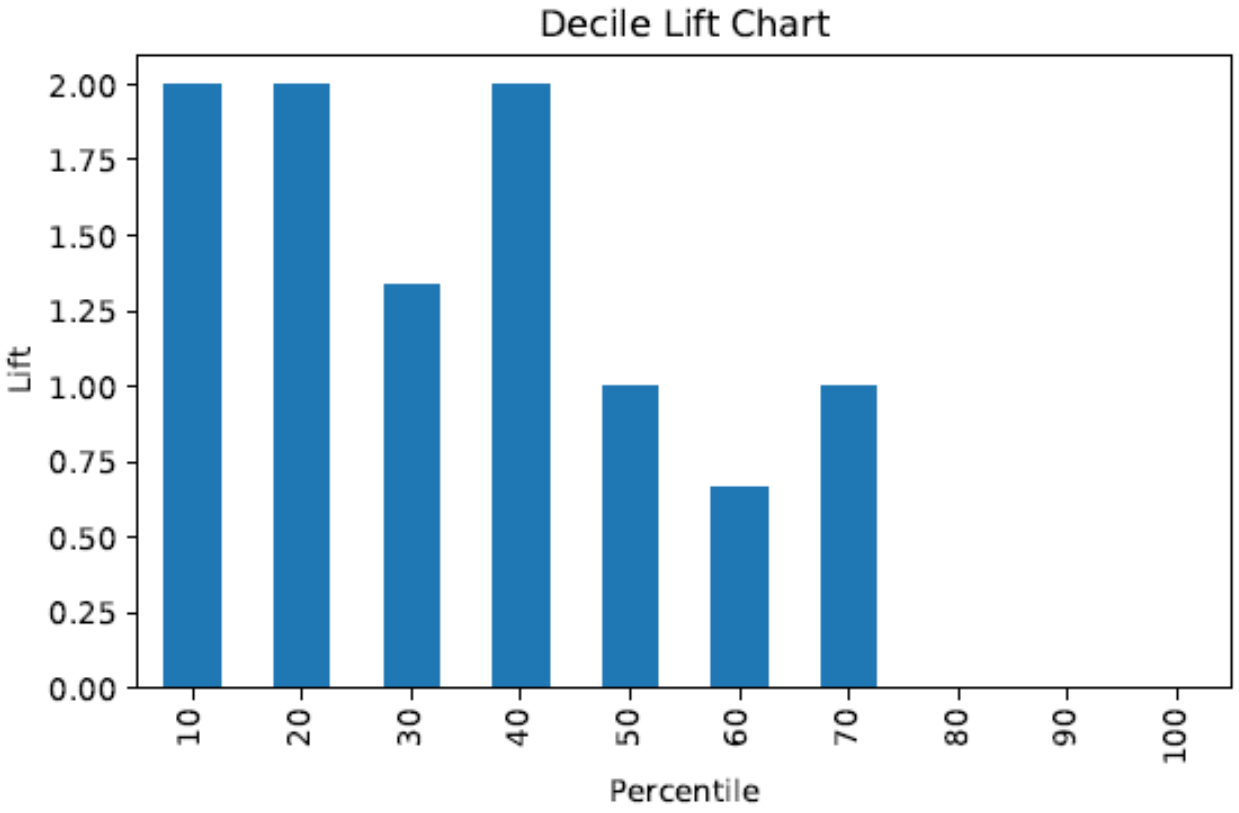
Lift (Gains) Chart – cumulative performance



To choose the 10 top records, after examining (e.g.) 24 cases (x-axis), about 9 owners (y-axis) have been correctly identified using the model vs. 5, using random selection (10* 12/24).

```
df = pd.read_csv('liftExample.csv')
df = df.sort_values(by=['prob'], ascending=False)
gainsChart(df.actual, figsize=(4, 4))
```

Decile Chart



- In “most probable” (top) decile, the model is twice as likely to identify the important class compared to just picking at random.
- In this example, the decile lift chart indicates that we can even use the model to select the top 40% records with the highest propensities and still perform almost twice as well as rendom.

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 (gains) Curve vs. Decile Charts

- Embody the 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

Lift (gains) vs. ROC curve

- Gains and ROC curves have similar appearance
- ROC curve and AUC provide a widely used single metric and visual to assess a model's ability to separate classes **overall**
- Lift (gains) yields intuitive measure of model **performance at different cutoffs**, but no overall metric

Asymmetric Costs

Misclassification Costs May Differ

- The cost of making a misclassification error may be higher for one class than the other(s)
- Looked at another way, the benefit of making a correct classification may be higher for one class than the other(s)

Example – Response to Promotional Offer

Suppose we send an offer to 1000 people, with an average response rate of 1% ("1" = response, "0" = nonresponse)

- "Naïve rule" (classify everyone as "0") has error rate of 1% (seems good)

- Using DM we can correctly classify eight 1's as 1's

  It comes at the cost of misclassifying twenty 0's as 1's and two 1's as 0's.

Example (cont.)

The Confusion Matrix*

	Actual 0	Actual 1
Predicted 0	970	2
Predicted 1	20	8

Error rate = (2+20) = 2.2% (higher than naïve rate)

*confusion matrix is often shown with predictions as rows, actuals as columns, the reverse of what Python produces

Introducing Costs & Benefits

Suppose:

- Profit from a “1” is $10
- Cost of sending offer is $1

Then:

- Under naïve rule, all are classified as “0”, so no offers are sent: no cost, no profit
- Under DM predictions, 28 offers are sent.
  - 8 respond with profit of $10 each
  - 20 fail to respond, cost $1 each
  - 972 receive nothing (no cost, no profit)
- Net profit = $60

Profit Matrix

	Actual 0	Actual 1
Predicted 0	$0	$0
Predicted 1	($20)	$80

Lift (again)

Adding costs to the mix, as above, does not change the actual classifications

Better: Use the lift curve and change the cutoff value for “1” to maximize profit

Generalize to Cost Ratio

Sometimes actual costs and benefits are hard to estimate.

- Need to express everything in terms of costs

  (i.e., cost of misclassification per record)
- Goal is to minimize the average cost per record

A good practical substitute for individual costs is the **ratio** of misclassification costs (e.g., “misclassifying fraudulent firms is 5 times worse than misclassifying solvent firms”)

Minimizing Cost Ratio

q_1 = cost of misclassifying an actual “1”,

q_0 = cost of misclassifying an actual “0”

Minimizing the **cost ratio** q_1/q_0 is identical to minimizing the average cost per record.

Software may provide option for user to specify cost ratio.

Note: Opportunity costs

- As we see, best to convert everything to costs, as opposed to a mix of costs and benefits.
- E.g., instead of “benefit from sale” refer to “opportunity cost of lost sale”.
- Leads to same decisions, but referring only to costs allows greater applicability.

Cost Matrix
(inc. opportunity costs)

	Predict as 1	Predict as 0
Actual 1	$8	$20
Actual 0	$20	$0

Recall original confusion matrix (profit from a "1" = $10, cost of sending offer = $1):

	Predict as 1	Predict as 0
Actual 1	8	2
Actual 0	20	970

Multiple Classes

For *m* classes, confusion matrix has *m* rows and *m* columns

- Theoretically, there are *m*(*m*-1) misclassification costs, since any case could be misclassified in *m*-1 ways.
- Practically, too many to work with.
- In decision-making context, though, such complexity rarely arises – one class is usually of primary interest.

Adding Cost/Benefit to Lift Curve

- Sort records in descending probability of success
  (success = belonging to the class of interest)
- For each case, record cost/benefit of actual outcome
- Also record cumulative cost/benefit
- Plot all records
  - X-axis is index number (1 for 1st case, n for nth case)
  - Y-axis is cumulative cost/benefit
  - Reference line from origin to y_n (y_n = total net benefit)

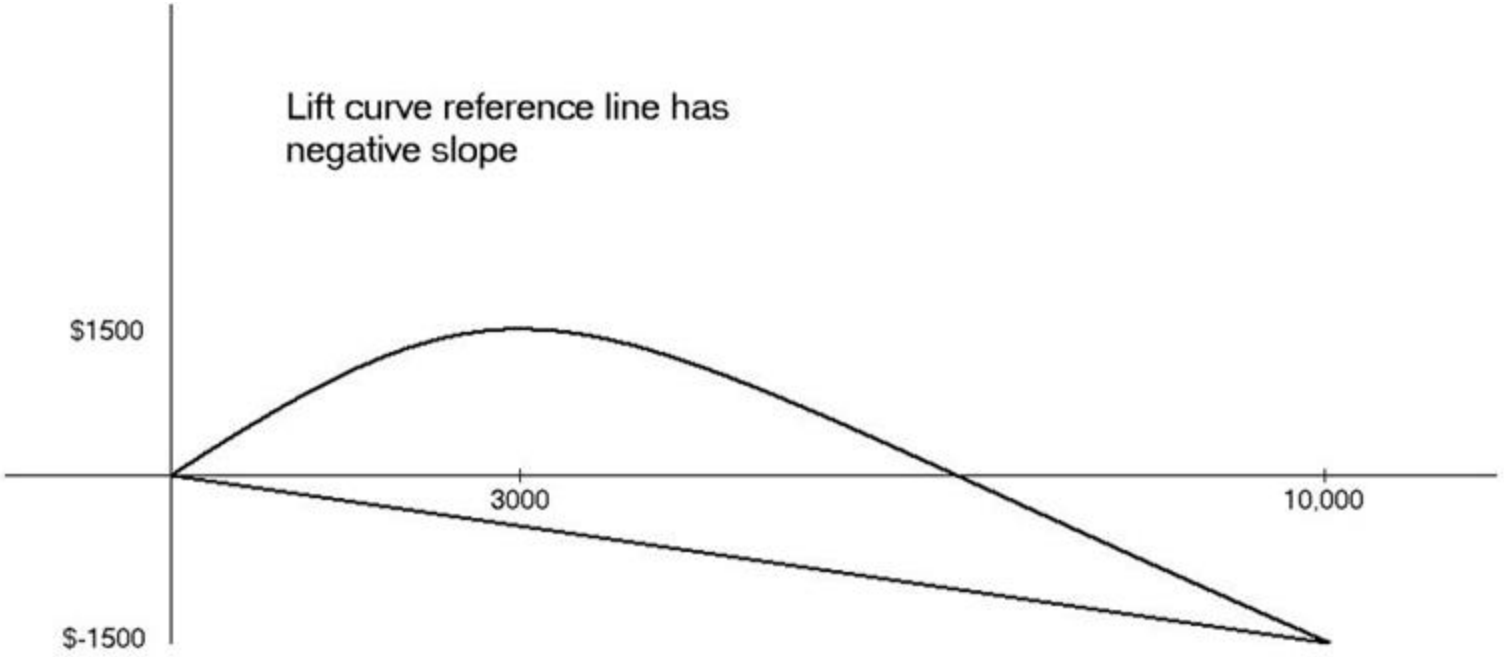
Lift Curve May Go Negative

If total net benefit from all cases is negative, reference line will have **negative slope.**

Nonetheless, goal is still to use cutoff to select the point where net benefit is at a maximum.

Negative slope to reference curve

It is entirely possible for a reference line that incorporates costs and benefits to have a negative slope if the net value for the entire dataset is negative. For example, if the cost of mailing to a person is \$0.65, the value of a responder is \$25, and the overall response rate is 2%, the expected net value of mailing to a list of 10,000 is $(0.02 \times \$25 \times 10,000) - (\$0.65 \times 10,000) = \$5000 - \$6500 = -\$1500$. Hence, the y-value at the far right of the lift curve (x = 10,000) is −1500, and the slope of the reference line from the origin will be negative. The optimal point will be where the cumulative gains curve is at a maximum (i.e., mailing to about 3000 people) as shown below.



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

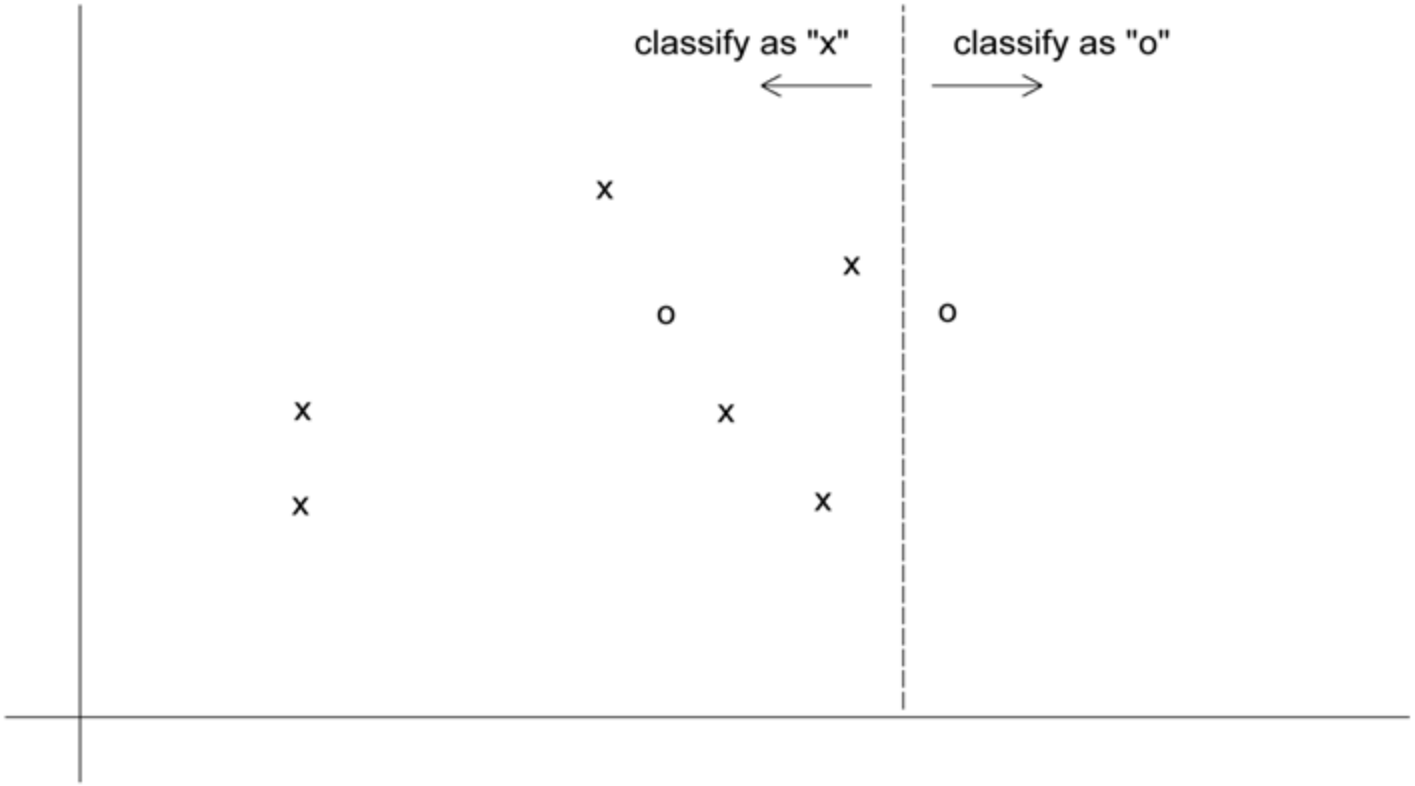
- Often we oversample rare cases to give model more information to work with.
- Typically use 50% “1” and 50% “0” for training
- Stratified sampling (weighted sampling or undersampling) is often used.

Example

Following graphs show optimal classification under three scenarios:

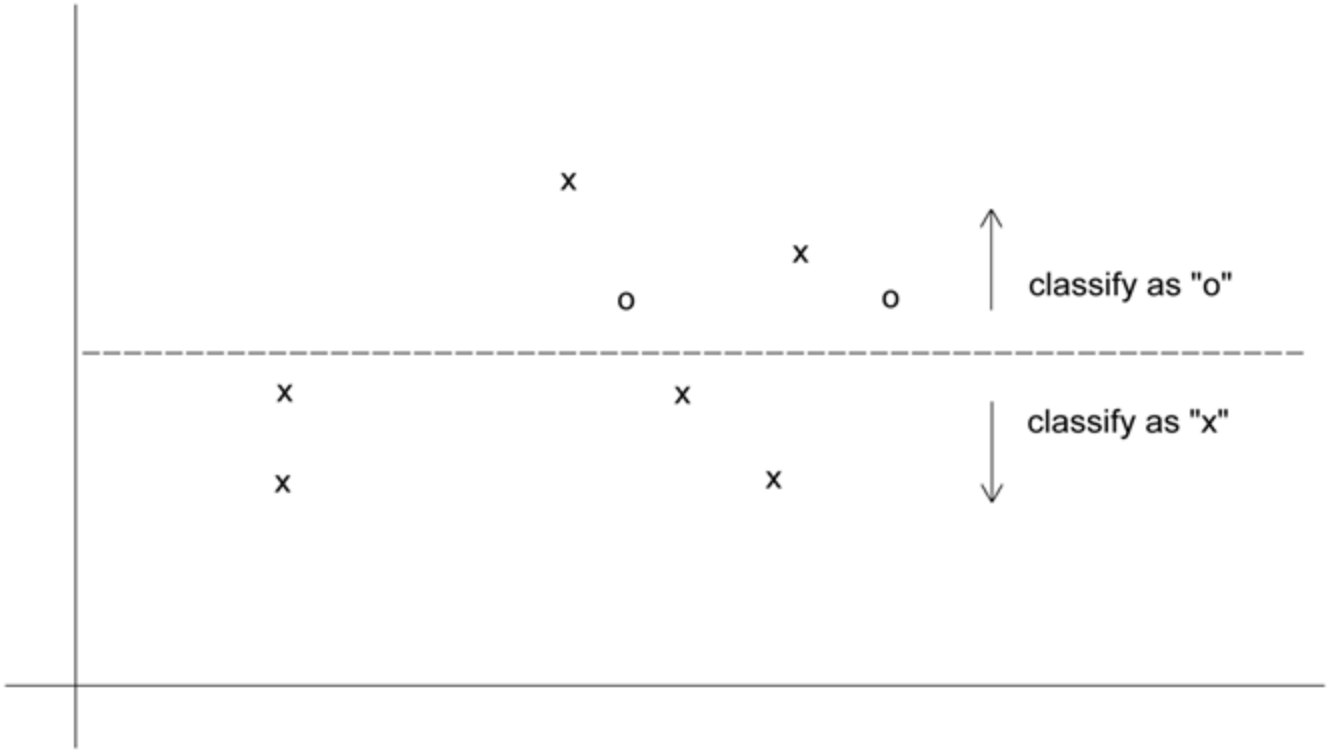
- Assuming equal costs of misclassification
- Assuming that misclassifying “o” is five times the cost of misclassifying “x”
- Oversampling scheme allowing DM methods to incorporate asymmetric costs

Classification: equal costs



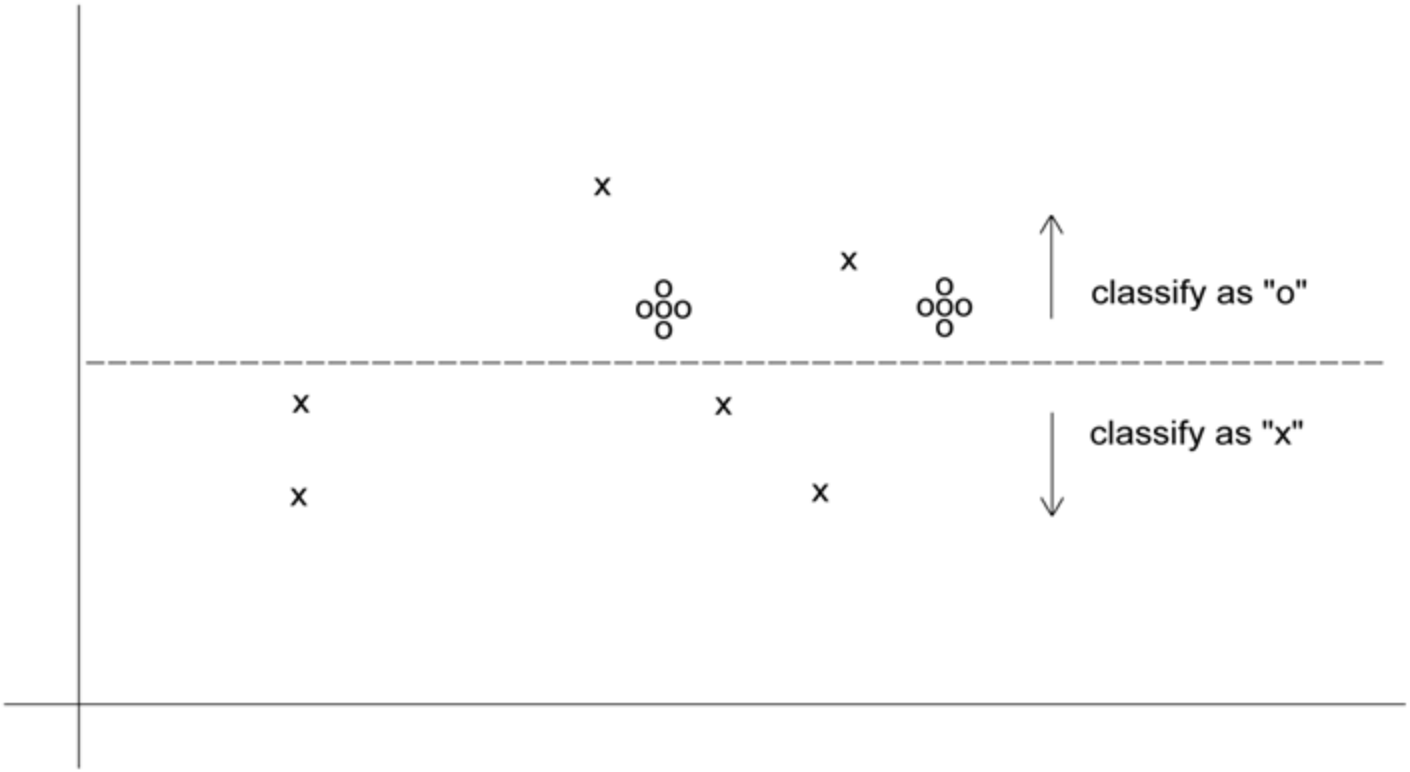
Classification: Unequal costs

Suppose misclassifying “o” is 5 times as costly as misclassifying “x”



Oversampling Scheme

Oversample “o” to appropriately weight misclassification costs



An Oversampling Procedure

1. Separate the responders (rare) from non-responders (two distinct sets or strata)
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

Oversampling Adjustment

When it comes time to assess and predict model performance, we will need to adjust for the oversampling in one of two ways:

1. Score the model to a validation set that has been selected without oversampling (i.e., via random sampling). This is more straightforward and easier to implement.
2. Score the model to an oversampled validation set, and reweight the results to remove the effects of oversampling.

Classification Using Triage

Take into account a gray area in making classification decisions

- Instead of classifying as C_1 or C_2, we classify as
  - C_1
  - C_2
  - Can't say
- The third category might receive special human review

An example is classification of documents found during legal discovery (reciprocal forced document disclosure in a legal proceeding). Under traditional human-review systems, qualified legal personnel are needed to review what might be tens of thousands of documents to determine their relevance to a case. Using a classifier and a triage outcome, documents could be sorted into clearly relevant, clearly not relevant, and the gray area documents requiring human review. This substantially reduces the costs of discovery.