OPEN DATA SCIENCE CONFERENCE

Burlingame I November 2nd 2017

@ODSC

@WinVectorLLC

@RStudio

Nov 02
2:00 PM
Room T2

Modeling big data with R, sparklyr, and Apache Spark

BIG DATARINTERMEDIATE

Dr. John Mount
Consulting Algorithmist/Researcher/Principal at Win-Vector LLC and
Co-author of Practical Data Science with R

Plan

- Define machine learning terms we are going to use in next set of exercises.

We will try a few supervised classification algorithms

- Logistic regression
- Decision tree
- Random forest
- Gradient boosted trees
- Neural network
- Naive Bayes

Classification

- Training data consists of rows with a vector x (the independent variables) and a scalar y (the dependent variable).
- y is "categorical" variable taking values from a fixed set of possible values.
    - If the set is exactly 2 possible values (such as TRUE/FALSE, 0/1, or Y/N) this is called binomial classification.
    - If the set is larger than 2 possible values this is called multinomial classification.

Evaluating classification algorithms 1/3

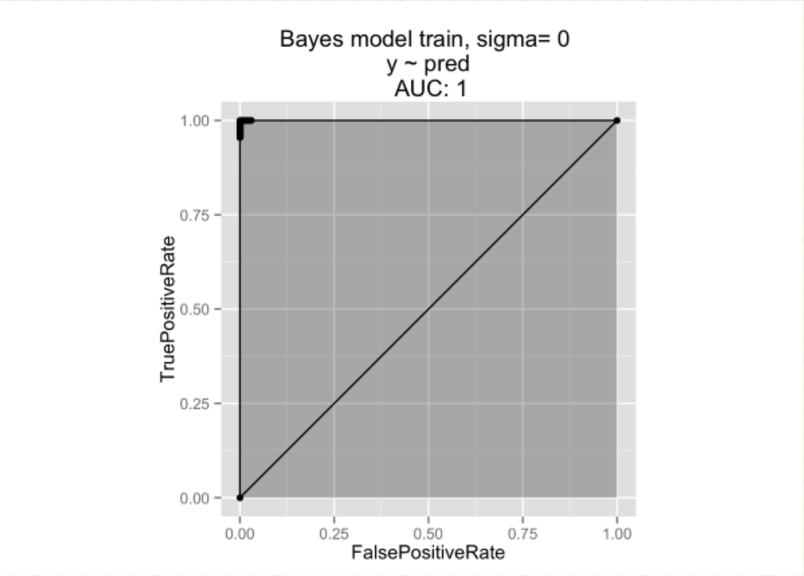
- *Always* insist on either:
  - Out of sample evaluation (test or hold-out set)
  - Cross-validated results (simulated hold-out).
- http://www.win-vector.com/blog/2015/09/willyourmodelworkpart1/

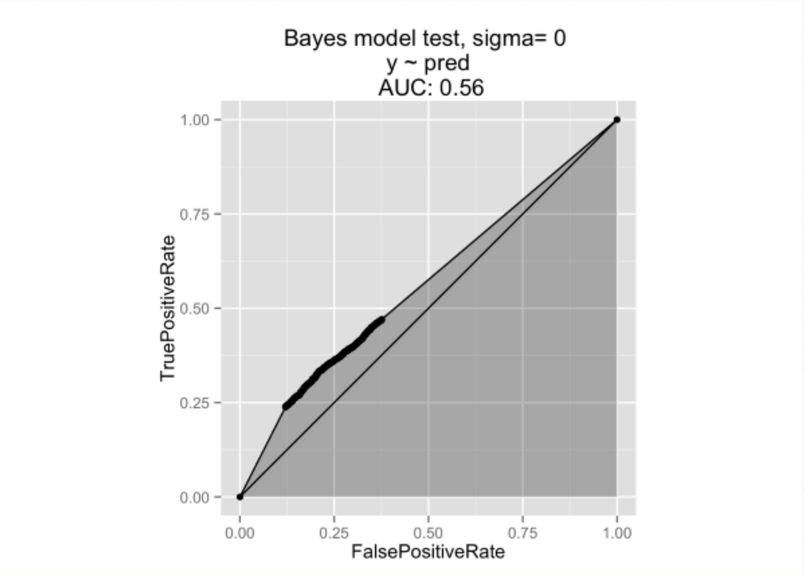
Why holdout, discussion

- With enough diverse variables is embarrassingly easy to produce models that look great on training data and are utterly worthless in practice.

- Test data is one way to *try to* simulate new data.

"Looks good on training" is the data scientist's version of developer's "works on my machine."





Helmuth von Moltke the Elder
"No plan of operations extends with any certainty beyond the first contact with the main hostile force."

A model scored only "at home" (on training data) is only a plan, not a result.

About holdout (continued)

- Models such as `lm()` are traditionally scored "in sample".
    - If there are not too many variables or categorical variables with too many levels this can be reasonable and statistically efficient.
    - In the era of "big data" this is no reason to ever do this.
        - Reserve some data for hold out
            - If you get the same result as in-sample you have lost very little
            - If you get a significantly different result, you have detected a problem
- Machine learning techniques (trees, ensembles, neural nets, …) tend to have very high model complexity and are well worth confirming on held-out data.
- Held-out data tends to be more exchangeable with future application data than training data, so you should be more interested in it than in training data (once you have a model).

Evaluating classification algorithms 2/3

- *Always* insist on seeing the full confusion matrix
  - Most summaries can be derived from it
    - Precision / Recall (sensitive to class prevalence)
    - Sensitivity / Specificity (insensitive to class prevalence)
  - Managers often only ask for accuracy, as it is the only measure they know of
- When to and when to not use accuracy as your evaluation metric:
  - http://www.win-vector.com/blog/2009/11/i-dont-think-that-means-what-you-think-it-means-statistics-to-english-translation-part-1-accuracy-measures/

```
caret::confusionMatrix(tab,
            positive= 'TRUE')
## Confusion Matrix and Statistics
##
##        lbsurvived
## pred    FALSE TRUE
##   FALSE   125   23
##   TRUE     14   57
##
##                Accuracy : 0.8311
##                  95% CI : (0.7747, 0.8782)
##     No Information Rate : 0.6347
##     P-Value [Acc > NIR] : 1.339e-10
##
##                   Kappa : 0.6267
##  Mcnemar's Test P-Value : 0.1884
##
##             Sensitivity : 0.7125
##             Specificity : 0.8993
##          Pos Pred Value : 0.8028
##          Neg Pred Value : 0.8446
##              Prevalence : 0.3653
##          Detection Rate : 0.2603
##    Detection Prevalence : 0.3242
##       Balanced Accuracy : 0.8059
##
##        'Positive' Class : TRUE
```

(note: data is a sample)

Tons of summaries

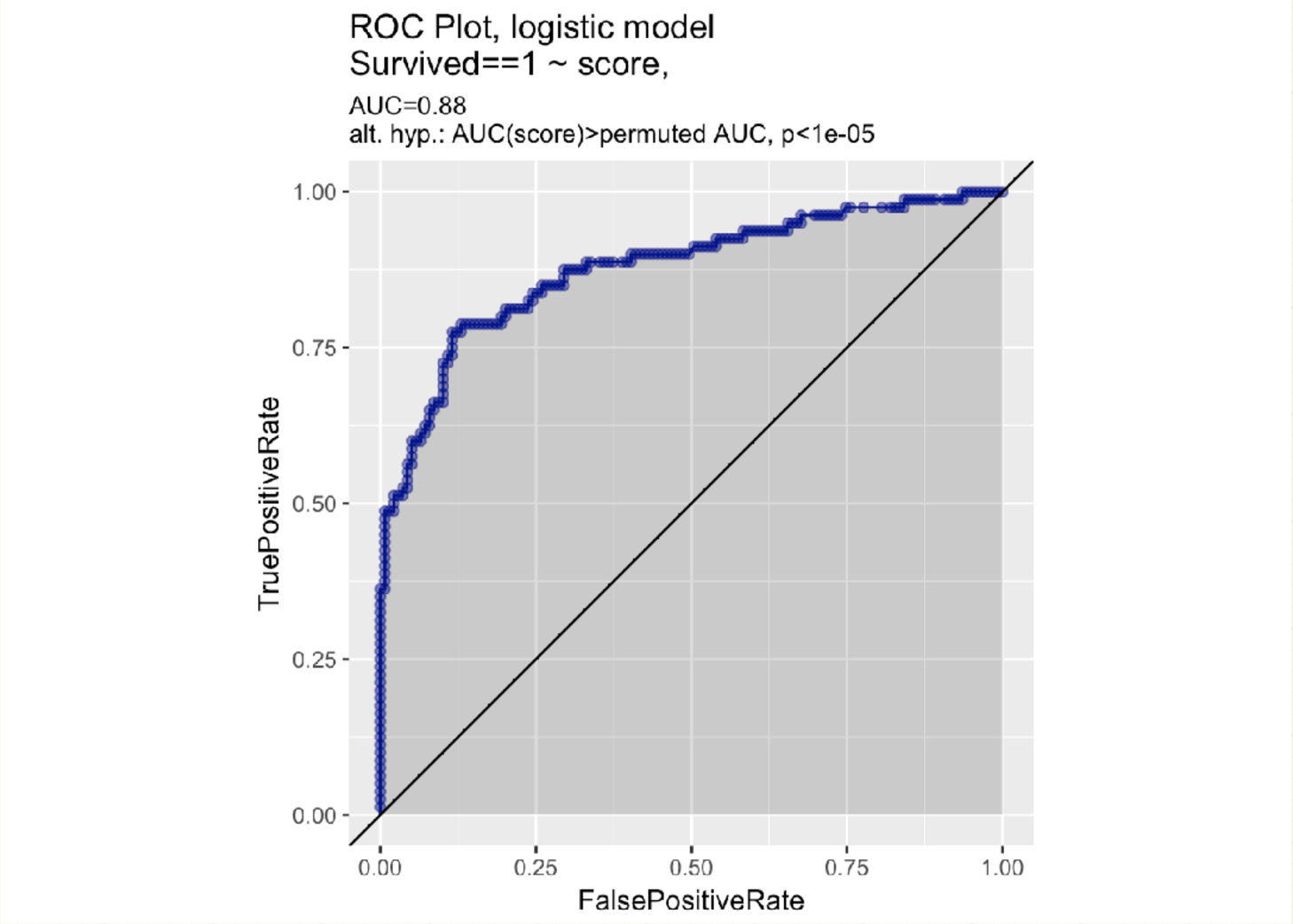
- Many of them are algebraically or monotone related
  - http://www.win-vector.com/blog/2016/07/a-budget-of-classifier-evaluation-measures/

Evaluating classification algorithms 3/3

- Some models return an estimated probability of being in a given class
- This allows finer evaluation of score quality
  - ROC plot / AUC score
  - Gain plot / lift curve / relative Gini index
  - deviance (expected log surprise).

ROC / AUC

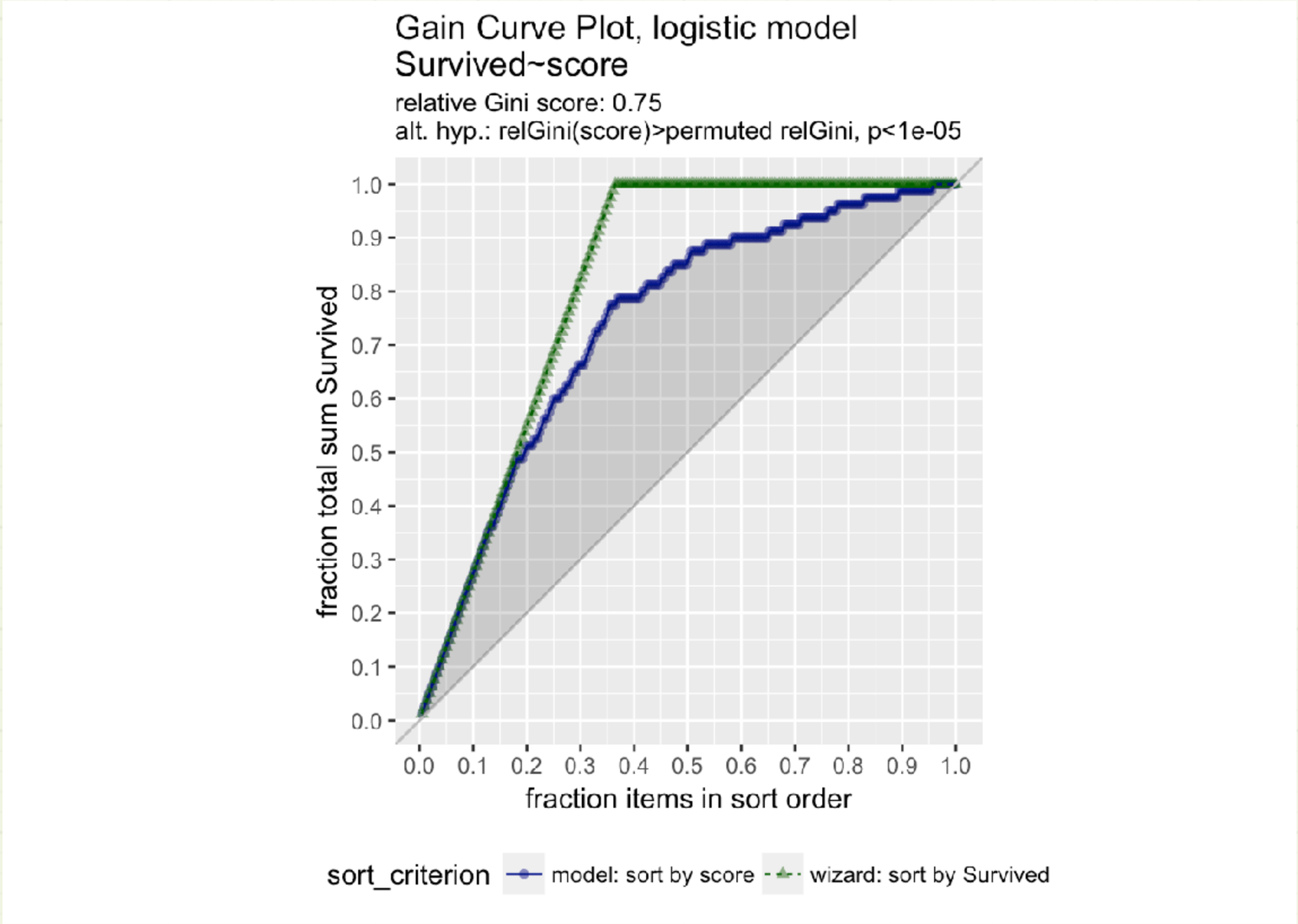
- For a scoring model each possible decision threshold induces a new classifier.
- The ROC plot shows the performance of all of these classifiers at the same time in terms of TruePositiveRate and FalsePositiveRate.
- AUC is "area under the curve" and is the shared region.
- ROC / AUC is insensitive to population prevalence.



`Exercises/solutions/04a-Spark-ML.html`

Gain / Lift

- Gain / lift curves are based on rank comparison.
- They show how similar sorting with respect to the outcome is to sorting with respect to the model score.
- Gain / lift is sensitive to population prevalence.



Feature importance

- Which features are doing the work?
- Different possible definitions:
  - Can we re-build a model if the feature had not been present?
  - Does changing the value of a feature (pulling it to its mean-value or permuting between rows) change the quality of the model in hand?
  - Is the feature dominant a lot in sub-models?

The paradox of features

- Features often have their best chance of seeming important alone
    - Despite possible masking effects of both dependencies and interactions (example: http://www.win-vector.com/blog/2016/09/variables-can-synergize-even-in-a-linear-model/)
- Features definitely look incrementally useless by the end of model construction
    - Classic linear regression: all features are orthogonal to the residuals on the training set (else you would have used them).
    - Iterative methods (gradient boosted trees): obvious combinations of features look useless (else you would add them).
    - Optimization based methods: no obvious tweaks available (else the optimizer would have continued on).

next: some machine learning at scale examples