Statistical Learning with *caret* in R

Wendy Martinez and Peter Meyer
Bureau of Labor Statistics

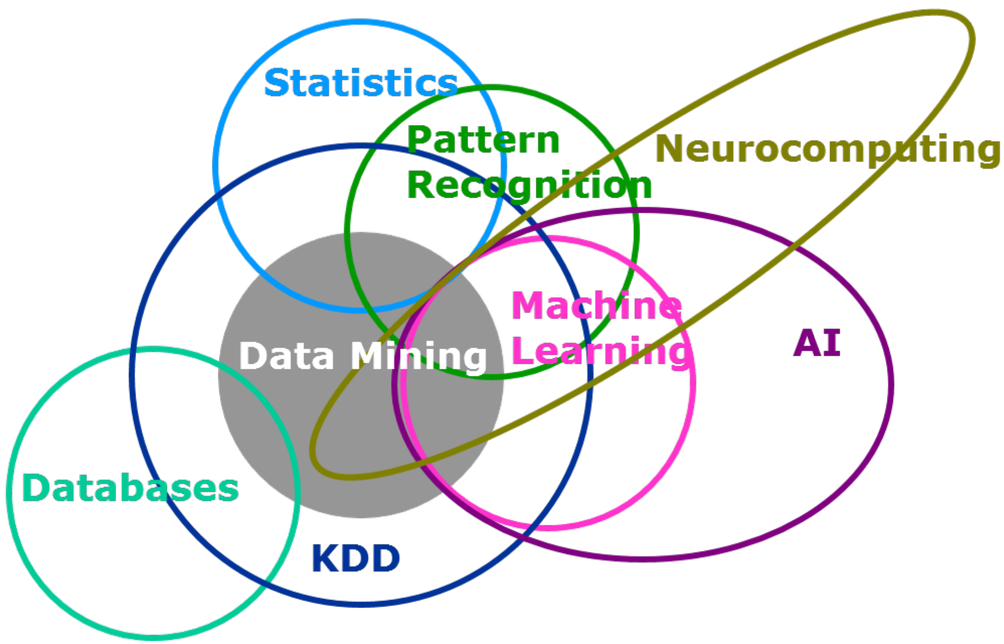
http://github.com/wendylmartinez/R-caret-May2019

For statistical agency staff

May 9, 2019

What is Statistical Learning?

- Learning from data
- Understand underlying phenomena generating the data



https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/

Statistical Learning

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering

https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/

Statistical Learning

- Two main types of learning from data
- Supervised learning
  - Observations are labeled with truth
  - Learn the relationship between them
  - Predictors and response variables
  - Examples: regression, classification

Statistical Learning

- Unsupervised learning – not discussed today
  - No labels or ground truth or “right answers”
  - Usually exploratory
  - Given data, look for interesting structure
- Example: finding clusters
  - Identify groups with similar data
  - Which are notably different from other groups

Statistical Learning in R

- Many, many packages in R for statistical learning – both but just supervised today
- Their syntax for training and prediction varies
- Max Kuhn developed the **caret** package ~2008 to unify and streamline the process
- **caret** = **C**lassification **a**nd **Re**gression **T**raining

Caret Package

- It’s on CRAN:
  - https://cran.r-project.org/web/packages/caret/index.html
- There are some references & links.

caret: Classification and Regression Training

Misc functions for training and plotting classification and regression models.

Version:	6.0-81
Depends:	R (≥ 2.10), lattice (≥ 0.20), ggplot2
Imports:	foreach, methods, plyr, ModelMetrics (≥ 1.1.0), nlme, reshape2, stats, withr (≥ 2.0.0)
Suggests:	BradleyTerry2, e1071, earth (≥ 2.2-3), fastICA, gam (≥ 1.15), ipred, k mlbench, MLmetrics, nnet, party (≥ 0.9-99992), pls, pROC, proxy, ra superpc, Cubist, testthat (≥ 0.9.1), rpart, dplyr
Published:	2018-11-20
Author:	Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Willi Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael I Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt.
Maintainer:	Max Kuhn <mxkuhn at gmail.com>
BugReports:	https://github.com/topepo/caret/issues
License:	GPL-2 | GPL-3 [expanded from: GPL (≥ 2)]
URL:	https://github.com/topepo/caret/

Caret Package

- Provides a unifying framework to explore models
- Has many useful tools
  - Data splitting into testing and training sets
  - Pre-processing
  - Feature selection, Model tuning
  - Estimating variable importance
  - Testing prediction models
  - More …

Caret

- We will illustrate just a few of the features – enough to get you started.
- Resources at the end to learn more about **caret**.

A Short Introduction to the caret Package

The **caret** package (short for Classification And REgression Training) contains functions to streamline the model training process for complex regression and classification problems. The package utilizes a number of R packages but tries not to load them all at package start-up (by removing formal package dependencies, the package startup time can be greatly decreased). The package "suggests" field includes 30 packages. **caret** loads packages as needed and assumes that they are installed. If a modeling package is missing, there is a prompt to install it.

Install **caret** using

```
install.packages("caret", dependencies = c("Depends", "Suggests"))
```

to ensure that all the needed packages are installed.

The **main help pages** for the package are at https://topepo.github.io/caret/ Here, there are extended examples and a large amount of information that previously found in the package vignettes.

caret has several functions that attempt to streamline the model building and evaluation process, as well as feature selection and other techniques.

What We’ll Cover

- Visualizing data
- Pre-processing data
- Cross-validation
- Model building or training
- Variable importance
- Measuring performance

Visualizations

- We follow the online book by Kuhn and use two simple data sets for illustration.
- Regression modeling – Boston housing data
- Classification – Fisher's iris data
- Let's look at these two data sets…

Visualizations

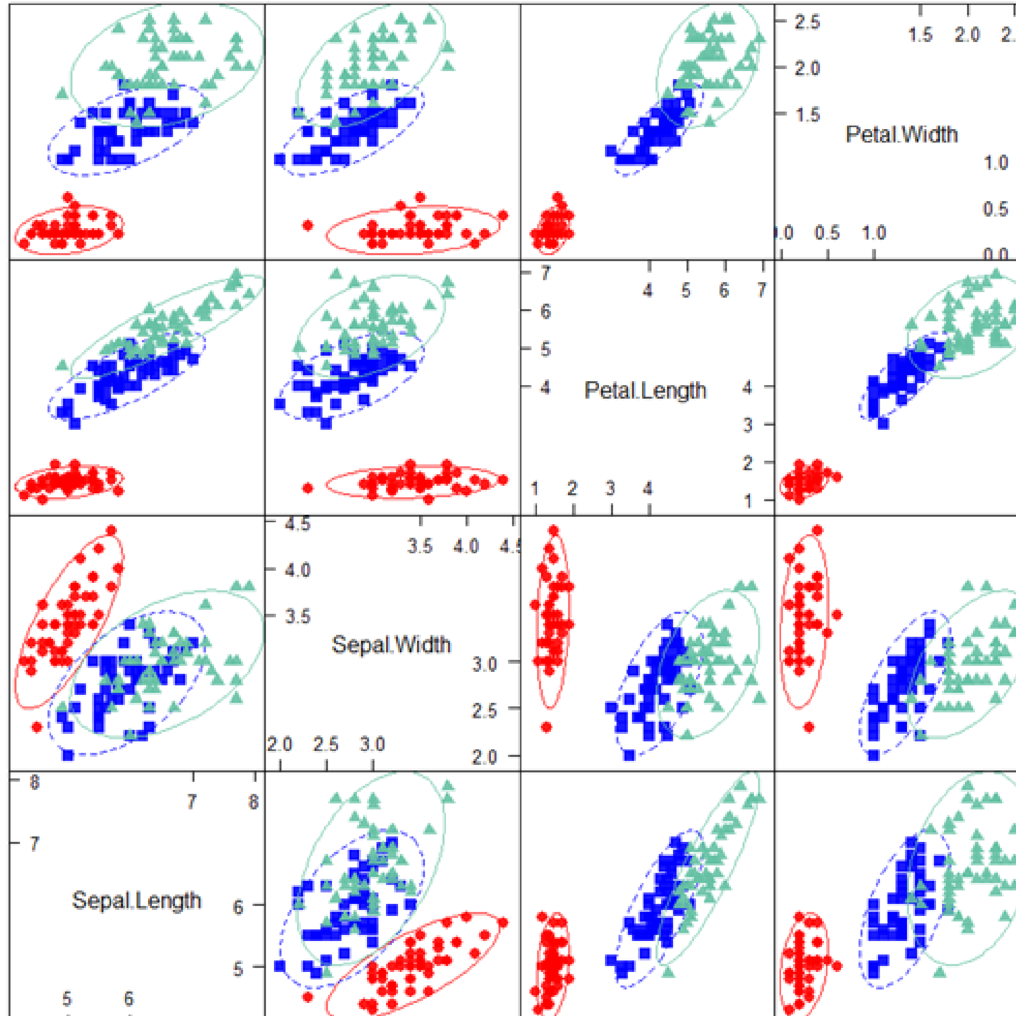
- It is always a good idea to view your data before building models.
- The **caret** package has a function called **featureplot**.
- It is a wrapper for **lattice** plots of predictors.
  - Classification: box, strip, density, pairs, ellipse
  - Regression: pairs or scatter
- Let's look at some examples ….

Scatter plot Matrix

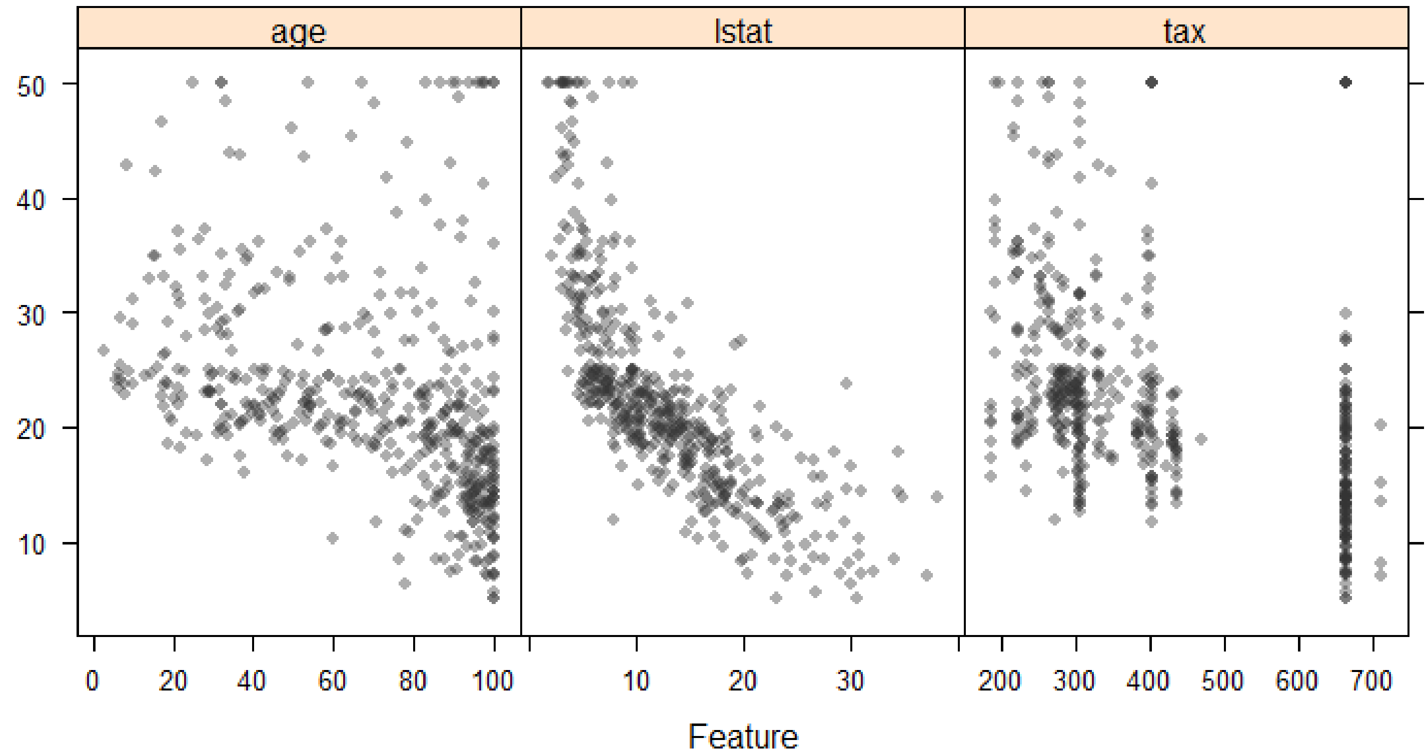
setosa

versicolor

virginica



Scatter plots of Predictors with Response Variable



Pre-Processing Data

- Has options to pre-process predictor data.
- Assumes all are numeric and any factors have been converted to dummy variables, e.g., using **dummyVars** (part of **caret**).
- There is a **preProcess** function can be used to implement several options.
- Let's look at help: **?preProcess**

Parameterize Models

- The **caret** package has a function **train** to build parameterize models based on training set
- Features
  - Use different parameters to build model
  - Evaluate models using resampling
  - Choose ‘best’ model
  - Estimate model performance

Build Parameterized Models

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

https://cran.r-project.org/web/packages/caret/vignettes/caret.html

Split the Data: train and test sets

- We need two data sets: training and testing.
- We can use the **createDataPartition** function to create these two data sets.
- If the response is a factor – a classification problem, then the random sampling is done so the class distribution is maintained in each.
- Returns a vector of index numbers for observations in the training set.
- Let’s look at some examples …..

Choose Models

- The caret package makes it easy to try out different models and compare results.
- It has 239 models as of April 2019.
  - Listed here: https://topepo.github.io/caret/available-models.html
- The function **getModelInfo** provides information.
- Look at **?getModelInfo**
- https://github.com/topepo/caret/tree/master/models/files

Classification Example

- Let’s look at a classification example using the iris data.
- We keep this simple and look at two options – linear and quadratic classifiers
- Go to R script….

Regression Example

- Now, we look at a regression example using the Boston Housing data.
- Again, we are keeping this simple and will look at a linear (least squares) model.
- Go to R script …

Resources

- https://cran.r-project.org/web/packages/caret/index.html
- https://cran.r-project.org/web/packages/caret/vignettes/caret.html
- https://topepo.github.io/caret/ # Main help pages
- http://appliedpredictivemodeling.com/ # Kuhn's book
- https://www.jstatsoft.org/article/view/v028i05 # Journal article
- https://github.com/topepo/caret # GitHub project
- https://topepo.github.io/caret/available-models.html # Models in **caret**
- https://www.analyticsvidhya.com/blog/2014/12/caret-package-stop-solution-building-predictive-models/ # Ref for regression example
- https://topepo.github.io/caret/index.html # Main source for these note
- https://www.analyticsvidhya.com/blog/2016/12/practical-guide-to-implement-machine-learning-with-caret-package-in-r-with-practice-problem/
- https://www.machinelearningplus.com/machine-learning/caret-package/
- https://datascienceplus.com/machine-learning-with-r-caret-part-1/

Caret use example

Issue: We want to estimate industry output growth for each year using other variables before the full output data is available
Desired output data are available 11+ months after reference year.

Goal here: Evaluate different statistical methods for prediction

Domain: 21 manufacturing industries
They have NAICS codes in the range 311-339
Data from years 2007-2014

Output definition, predictors, timing

- Dependent variable is growth in industry's value-of-production from previous year (called "output" casually, and "prod" in the data)
- For each industry-year we have as predictors the growth rates of 7 other measures of industry activity correlated to output
- They are highly correlated to output:

Predictors: growth rates in each industry-year	**Correlation to dependent variable prod**
Industrial Production Indexes (FRB IPI)	0.667
Producer Price Indexes (PPI)	0.526
M3 shipments survey (Census)	0.768
Imports to U.S. (Census, USITC)	0.758
Exports from U.S. (Census, USITC)	0.668
Wages paid (quarters 1-3) (QCEW)	0.631
Employment (quarters 1-3) (QCEW)	0.731

Data in Mfg3.csv

All growth rates are in form $ln(x_t/x_{t-1})$, where *t* is a year

21 industries; 8 years
N=168 industry-years

Dependent variable: prod

Predictors (growth rates in each industry)
Industrial Production Indexes (FRB IPI)
Producer Price Indexes (PPI)
M3 shipments survey (Census)
Imports to U.S. (Census, USITC)
Exports from U.S. (Census, USITC)
Wages paid (quarters 1-3) (QCEW)
Employment (quarters 1-3) (QCEW)

year	ind	prod	ipi	ppi	imports	exports	wages	emp	M3	lagvpfrac	lnlagvp	ind2
2007	311	0.075562	0.038259	0.077644	0.087799	0.18503	0.029893	0.002428	0.061766		13.06275	31
2008	311	0.084582	0.015184	0.091229	0.141107	0.224411	0.019372	0.001756	0.035133	0.109498	13.11222	31
2009	311	-0.03228	-0.00995	-0.02464	-0.10091	-0.0994	-0.01614	-0.0182	-0.08137	0.11487	13.21961	31
2010	311	0.046659	0.041526	0.034598	0.127003	0.148456	0.025173	-0.00594	0.061022	0.133898	13.13753	31
2011	311	0.077819	0.023033	0.086368	0.199934	0.141853	0.036864	0.009171	0.123967	0.126226	13.20682	31
2012	311	0.045978	0.043443	0.037072	0.077735	0.078512	0.027606	0.003998	0.026183	0.124123	13.31296	31
2013	311	0.017766	0.015578	0.014193	-0.02476	0.058648	0.018734	0.004241	0.026325	0.125732	13.33315	31
2014	311	0.031749	0.019116	0.038476	0.077429	0.027092	0.031389	0.012605	0.05847	0.123709	13.35283	31
2007	312	0.009757	-0.00813	0.030566	0.087573	0.085578	0.0296	0.007707	0.051888		11.68511	31
2008	312	-0.03712	-0.06667	0.042386	-0.00455	0.133308	0.017912	0.002822	0.041283	0.025867	11.69813	31
2009	312	0.022074	-0.0836	0.04622	-0.09371	-0.09113	-0.0896	-0.04719	-0.02674	0.024428	11.67156	31
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

	IPI	PPI	Imports	Exports	Wages	Employment	Value of Production
IPI	1						0.667
PPI	0.207	1					0.526
Imports	0.780	0.600	1				0.768
Exports	0.599	0.648	0.824	1			0.758
Wages	0.853	0.283	0.728	0.638	1		0.668
Employment	0.793	0.198	0.614	0.545	0.946	1	0.631
M3 shipments	0.731	0.647	0.865	0.776	0.736	0.647	0.731

Problem: predictors are highly colinear

- Pair-wise correlations across the variables are generally high
- The **Variance inflation factor** (VIF) measures how much colinearity raises the variance of OLS-estimated coefficients
  - The VIF is constructed by regressing each predictor on the others and using the R^2:
    $$VIF_i = \frac{1}{1 - R_i^2}$$
  - The VIFs of these variables are all over 3, and some over 10. Colinearity is high, not just pairwise.

A core problem here: Colinearity → Overfitting

Overfitting → Predictions overly sensitive to random variation

This hurts OLS predictions a lot.

Prediction accuracy measure

- Predict output growth for the test year by each model
  - Each year's data will be **test** data in one 'fold'
  - That gives a "**Year-wise cross-validation**" measure of quality
  - Year-wise measures of error match our production objective, which is to predict all industries in a new year at once

- Compare predictions based on root mean squared error (RMSE)
  - This is a common choice when continuous variables
  - Alternatives: R-squared, or weighting worst predictions more, or an asymmetric loss function

Data and models overview

- Full data: all industries, 2007-2014, all growth variables
  - Pruning variables is feasible. But can we benefit from all the variables?
- Estimate coefficients on **training** data
  - All years but one
    - Predictors here: 7 growth predictors, 3 industry indicators
- Predict output growth for the test year by each model
    - Tricky choice here: Using later years to predict earlier years.
    - Yes, we’re going to try that.

OLS regression

Ordinary least squares minimizes squared-errors, and is unbiased.

$$\hat{\beta}_{OLS} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{N} (y - X\beta)^2 \right\}$$

Residuals from OLS appear normally distributed and autocorrelated.

We'll try an OLS model with 7 industry activity predictors and 3 industry dummies (or factors, or categories)

And we'll try a model with 3 selected growth predictors and the factors

Ridge and Lasso regressions

A ridge regression is a constrained LS, designed to reduce sensitivity resulting from colinearity:

$$\hat{\beta}_{ridge} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{N}(y - X\beta)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq R$$

The ridge regression constraint is denoted R above. R is a **hyperparameter**, or **tuning parameter**. A smaller R presses all the coefficients toward zero. It adds a bias but reduces the problem of overfitting.

A **lasso** regression is a constrained least-squares also:

$$\hat{\beta}_{lasso} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{N}(y - X\beta)^2 \right\} \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq L$$

Lasso tends to SELECT among regressors, that is, to push some of the betas toward zero. (Tibshirani, 1996)

Basic caret code

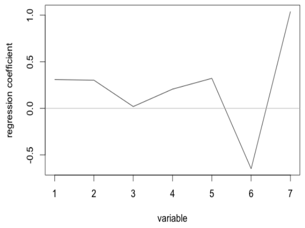
Jump to R Studio here.

For cross-validation of models built from 7 years and applied to the other one, using `caret` package

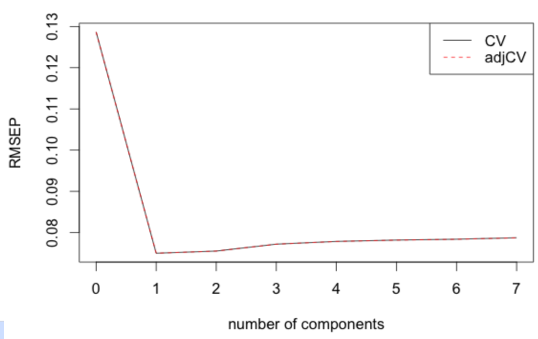
```
for (yr in 7:14) {
  TrainData <- subset(Mfg3, year != 2000+yr)  # training set, used to set coefficients
  TestData <- subset(Mfg3, year == 2000+yr)   # test set, for evaluating accuracy

  # Train the model, meaning set up coefficients to predict the year left out
  lmtrainedmodel <- caret::train(prod ~ ipi+ppi+imports+exports+wages+emp+M3,
                               data = TrainData, method = "lm")

  predictions <- predict(lmtrainedmodel, newdata = TestData)   # make prelim estimates
  RMSE <- sqrt(mean((TestData$prod - predictions)^2))  # calc root mean squared error

  # then display RMSE and compare across models
  # . . . .
}
```

Principal components regression

- The 7 growth predictors are laid out in 7-dimensional space and rotated so one dimension has as much variation as possible.
- The 1st principal component is a linear combination of the 7 growth predictors
  - Can benefits from all predictors
  - No attempt to fit to dependent variable
  - It weights mainly IPI, PPI, exports, and employment-minus-wages; chart



- The 1st component is then the one growth predictor in an OLS
  - The 2nd component doesn't help; see 2nd chart.
- RMSE for 3-digit industries: .065
  - The best of all methods tried



More tools and challenges

- Can export a file with R2 and RMSE for each fold
- Can try more tuning parameters (hyperparameters) including specific values of them
- Challenge: Errors in the underlying regressions can pop out of caret somewhat strangely; e.g., too many parameters for too little data
- However we can rank several statistical methods by their average RMSE performance
  - As we did here: Meyer and Martinez. Predicting industry output with statistical learning methods. 2017. In *JSM Proceedings*, Government Statistics Section. Alexandria, VA: American Statistical Association. 3256-3269.

Lessons about the original problem

- OLS was highly sensitive to collinearity of predictors
- Across models, errors were largest for:
  - Small industries, presumably because of smaller underlying samples
  - Years 2007-2010 - apparently measurements was noisier in recession and economically turbulent period.
- Underlying relations between predictors and production seem to be linear with noise
  - No major bends and curves
  - Spline and random forest models didn't perform better; the underlying problem doesn't call for their flexibility
  - Splitting up industries into groups did not seem to help accuracy
  - Using all variables CAN help, a bit
- N is not large here; can't try all models
  - N = 147 observations in training set for 21 3-digit industries
  - Some estimators break down because of too few observations

Contact Information

Wendy Martinez
Bureau of Labor Statistics
martinez.wendy@bls.gov
202-691-7400

Peter Meyer
Bureau of Labor Statistics
meyer.peter@bls.gov
202-691-5678