O'REILLY®

Security

BUILD BETTER DEFENSES

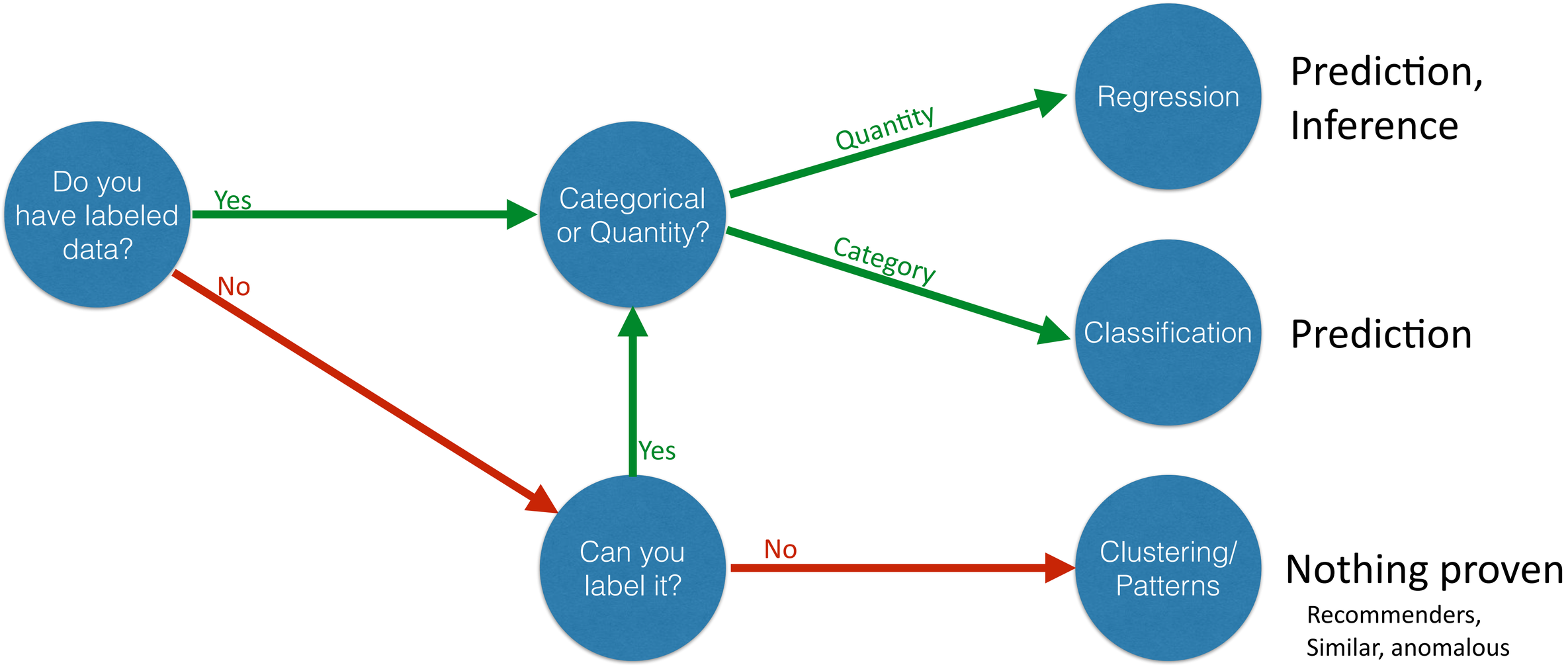
oreillysecuritycon.com
#oreillysecurity

Foundations of Security Data Science

Machine Learning: Supervised

Jay Jacobs
Charles Givre

What Data Science kind of is…



Data Science Topics

Core Statistics
Descriptive (range, median, mean)
Confidence Intervals
Correlation
Tests (t.test, ks.test, fitting)

Regression/Classification
Labeled Data
Can falsify claims
Prediction, maybe inference
No free lunch

Clustering/Unsupervised
Unlabeled data
Cannot falsify claims
Clustering
Anomoly Detection

Visualization
Visual cues, coord. system, scale, context
Pre-attentive processing, saccadic
Working and Long-term memory
Accuracy in Decoding, mumbling

Feature Engineering

1. Define the Object of Measurement
   These will become the rows in your data (one per object)

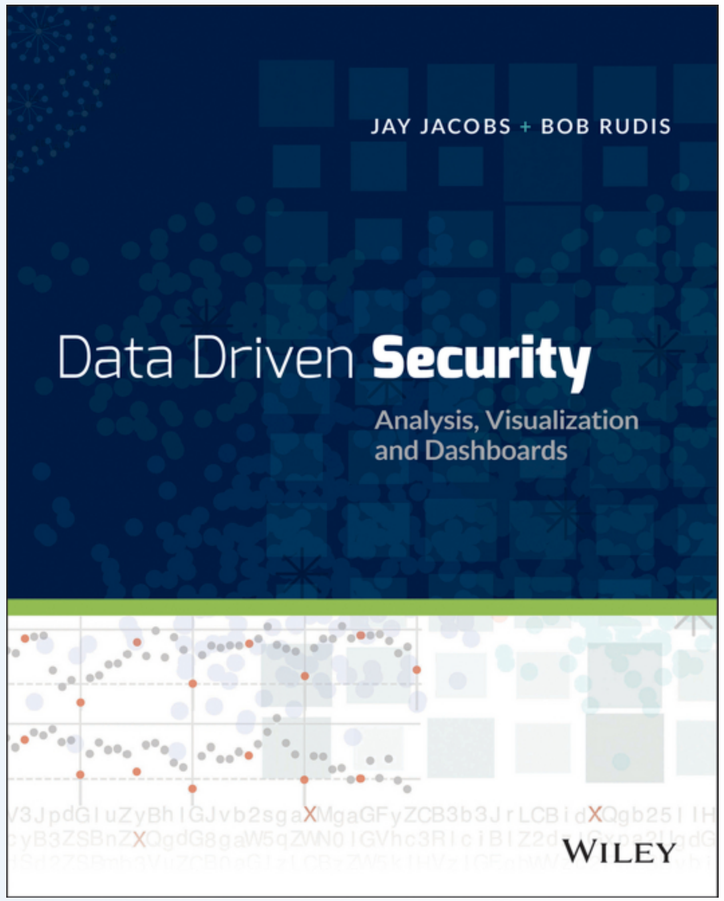
2. Define one or more “features” that describe each object
   These will become the columns in your data

3. Run Algorithm (more on this later)

4. Optimally, measure feature contribution and model performance

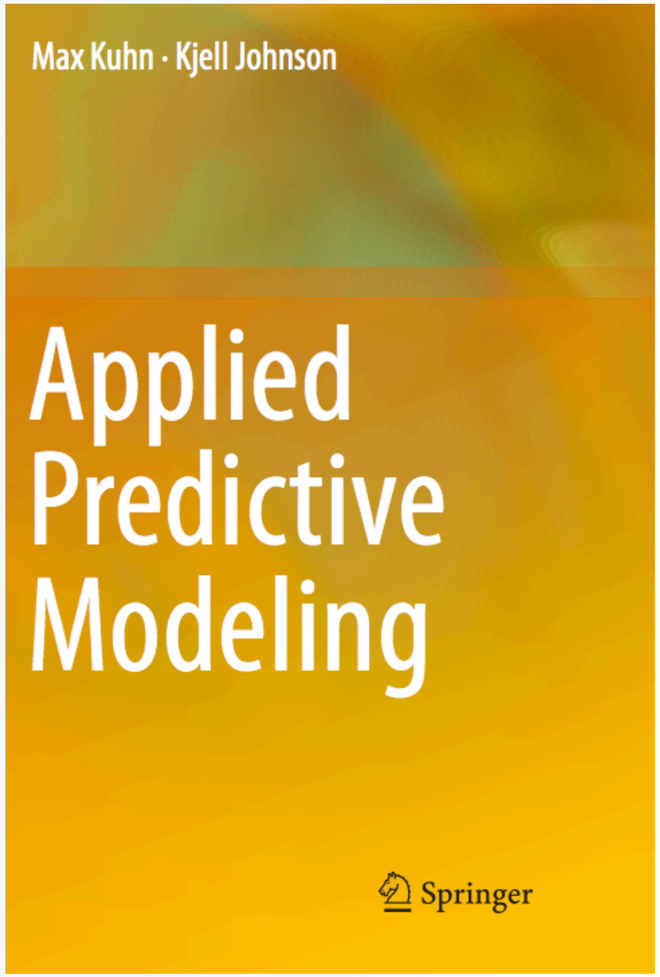
Intuition vs. Expertise

"…It's your experience with information security that will guide the direction of the analysis, provide context to the data, and help apply meaning to the results. In other words, domain expertise is beneficial in the beginning, middle, and end of all your data analysis efforts…."



Prediction vs. Interpretation

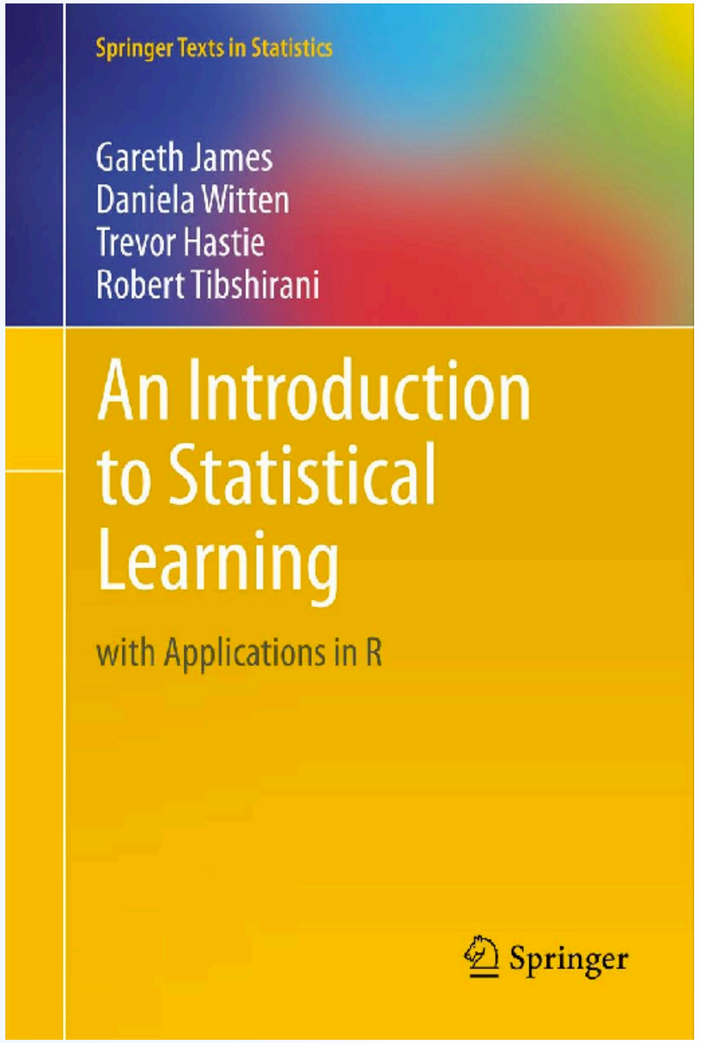
"While the primary interest of predictive modeling is to generate accurate predictions, a secondary interest may be to interpret the model and understand why it works. The unfortunate reality is that as we push towards higher accuracy, models become more complex and their interpretability becomes more difficult."



https://topepo.github.io/caret/available-models.html

No Free Lunch / Measuring Quality

Why is it necessary to introduce so many different statistical learning approaches, rather than just a single *best* method? ***There is no free lunch in statistics***: no one method dominates all others over all possible data sets. Selecting the best approach can be one of the most challenging parts of performing statistical learning in practice."



O'REILLY®

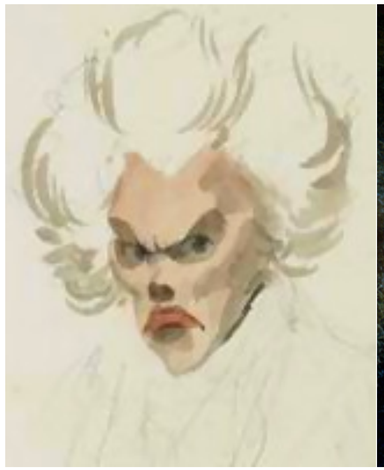
Security

BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

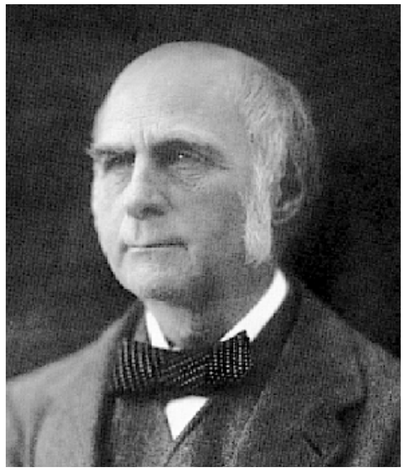
Brief Overview of Regression Analysis

Regression: Standing on the Shoulders of Giants



Adrien-Marie Legendre (1805)

Carl Frederic Gauss (1809)

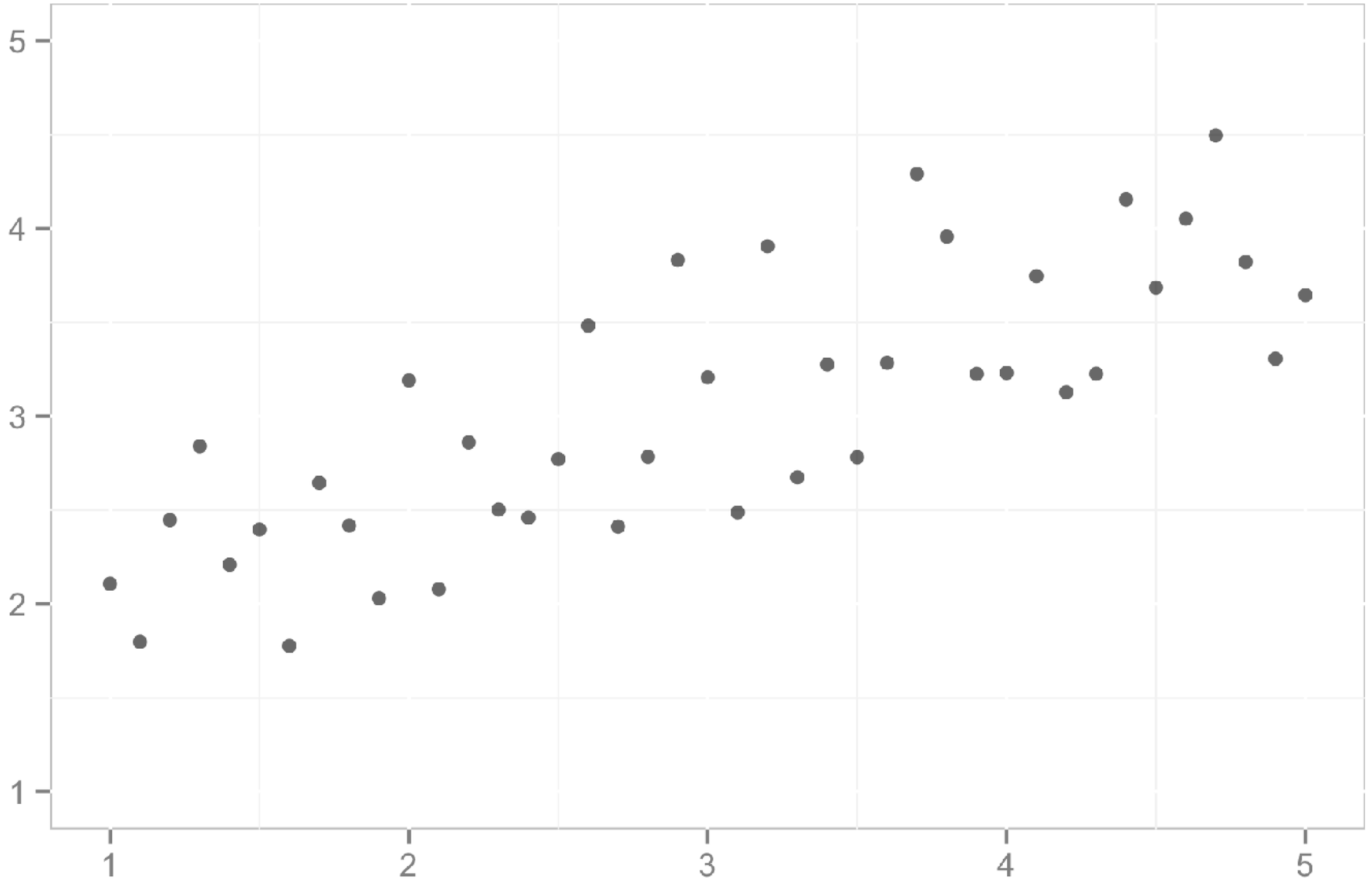


Francis Galton (1869)

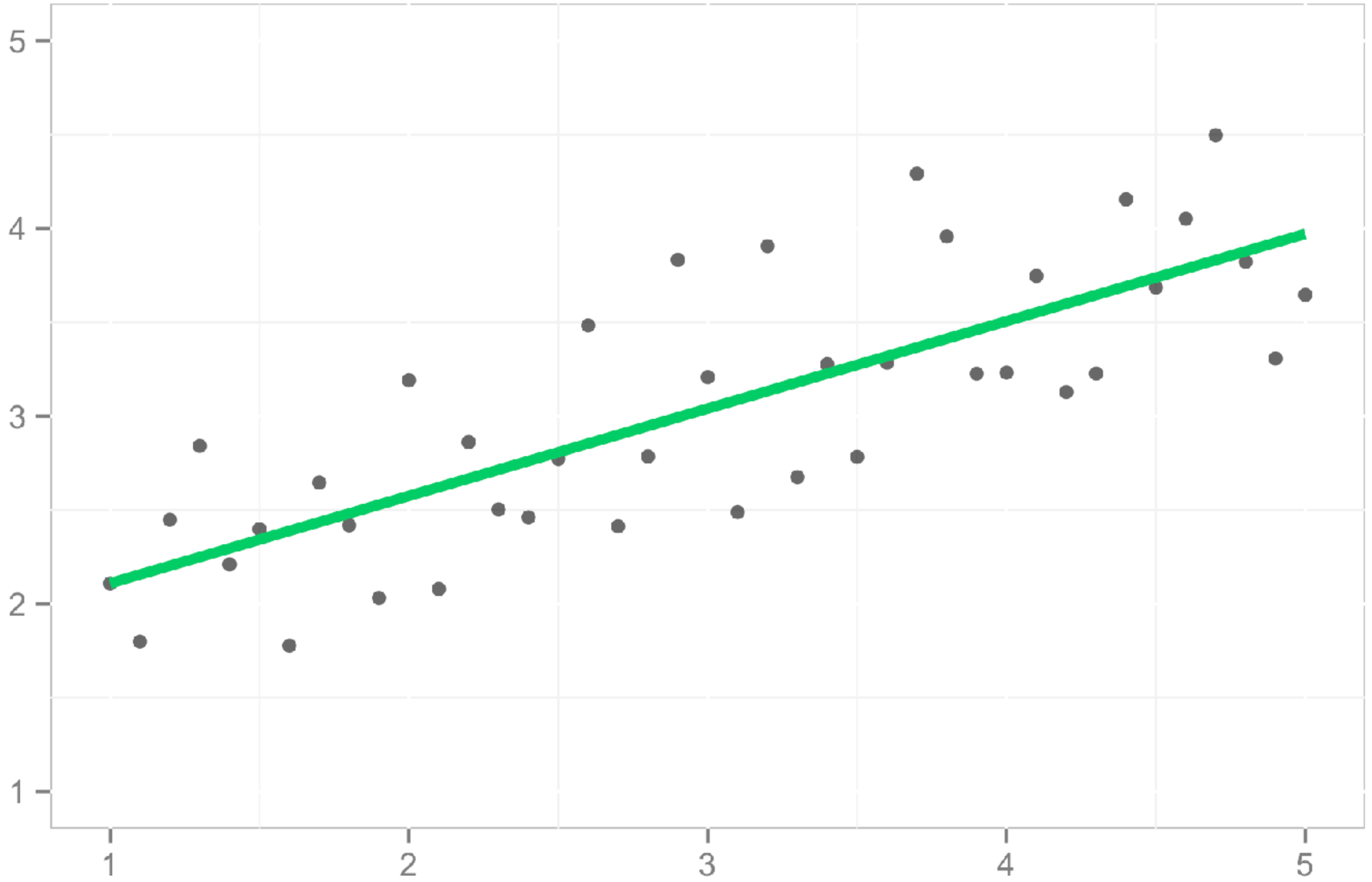
Karl Pearson (1903)

Ronald A. Fisher (1922)

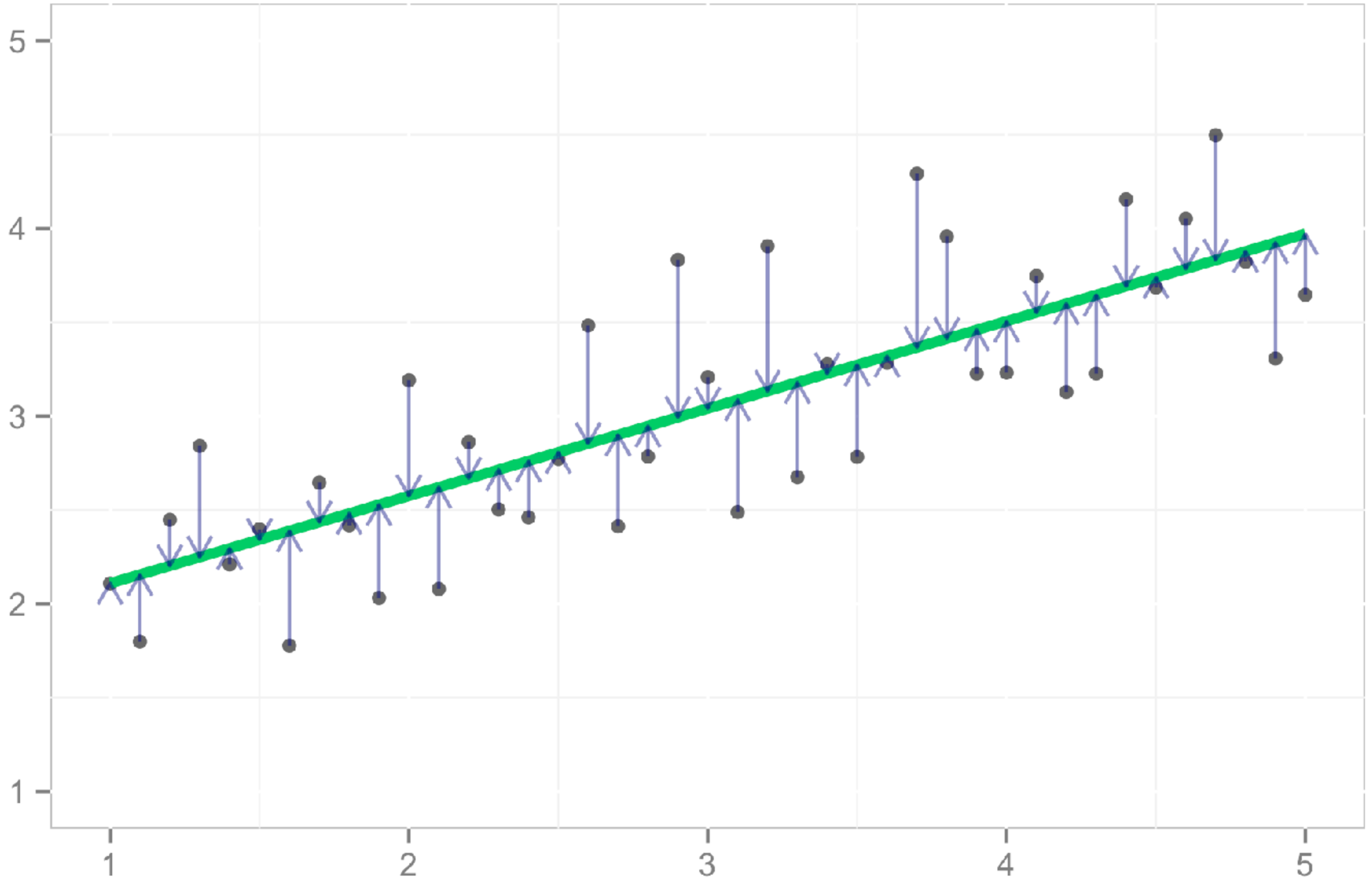
Most Likely Y, given X?



Least Squares



Least Squares



Simple Linear Regression

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

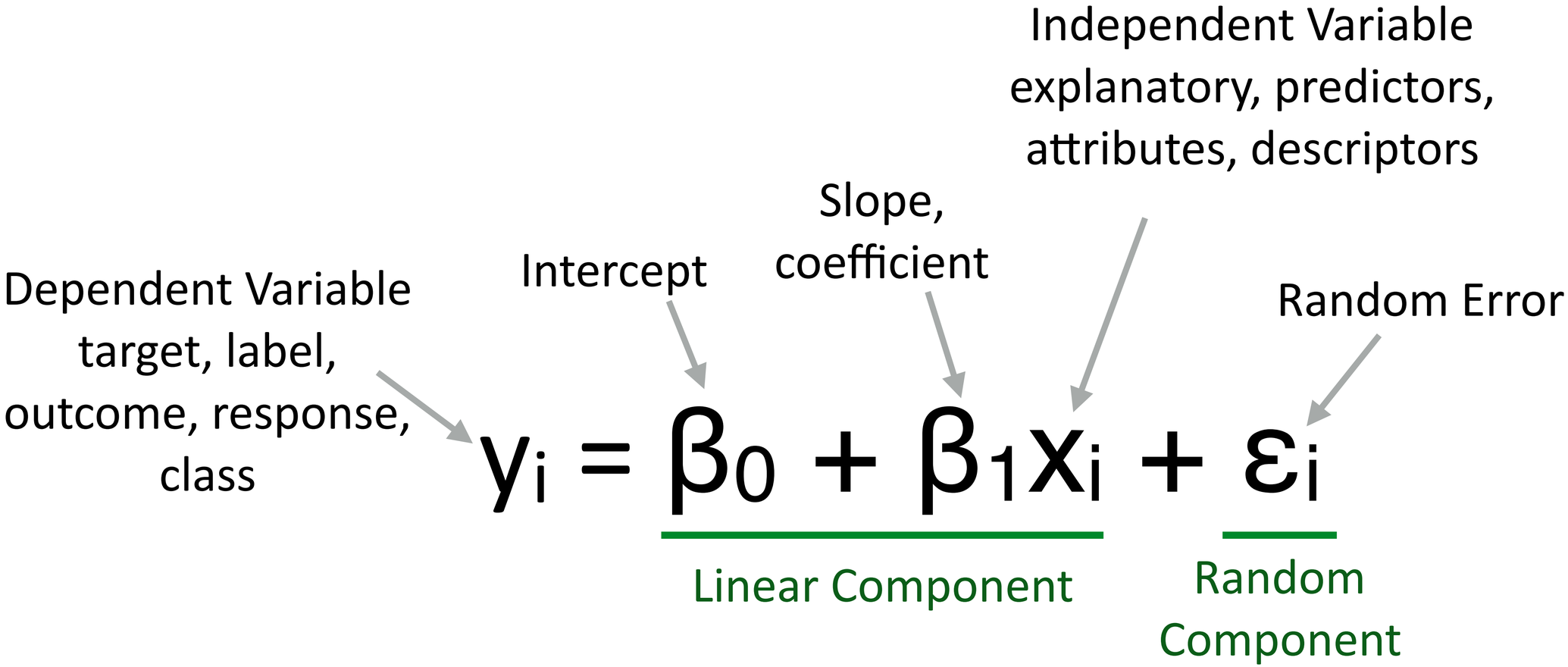
…Define the model beforehand

Simple Linear Regression

$$y_i = \underbrace{\beta_0 + \beta_1 x_i}_{\text{Linear Component}} + \underbrace{\varepsilon_i}_{\text{Random Component}}$$

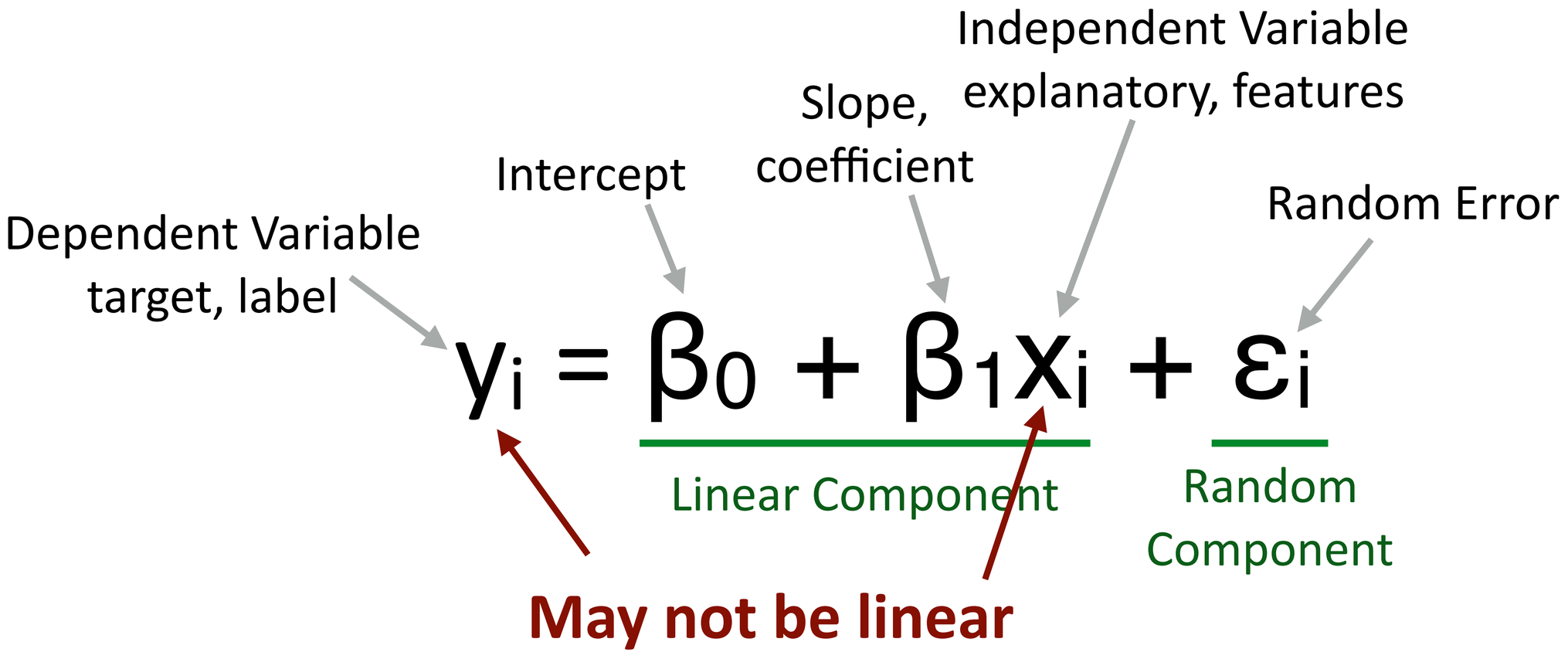
…Define the model beforehand

Simple Linear Regression



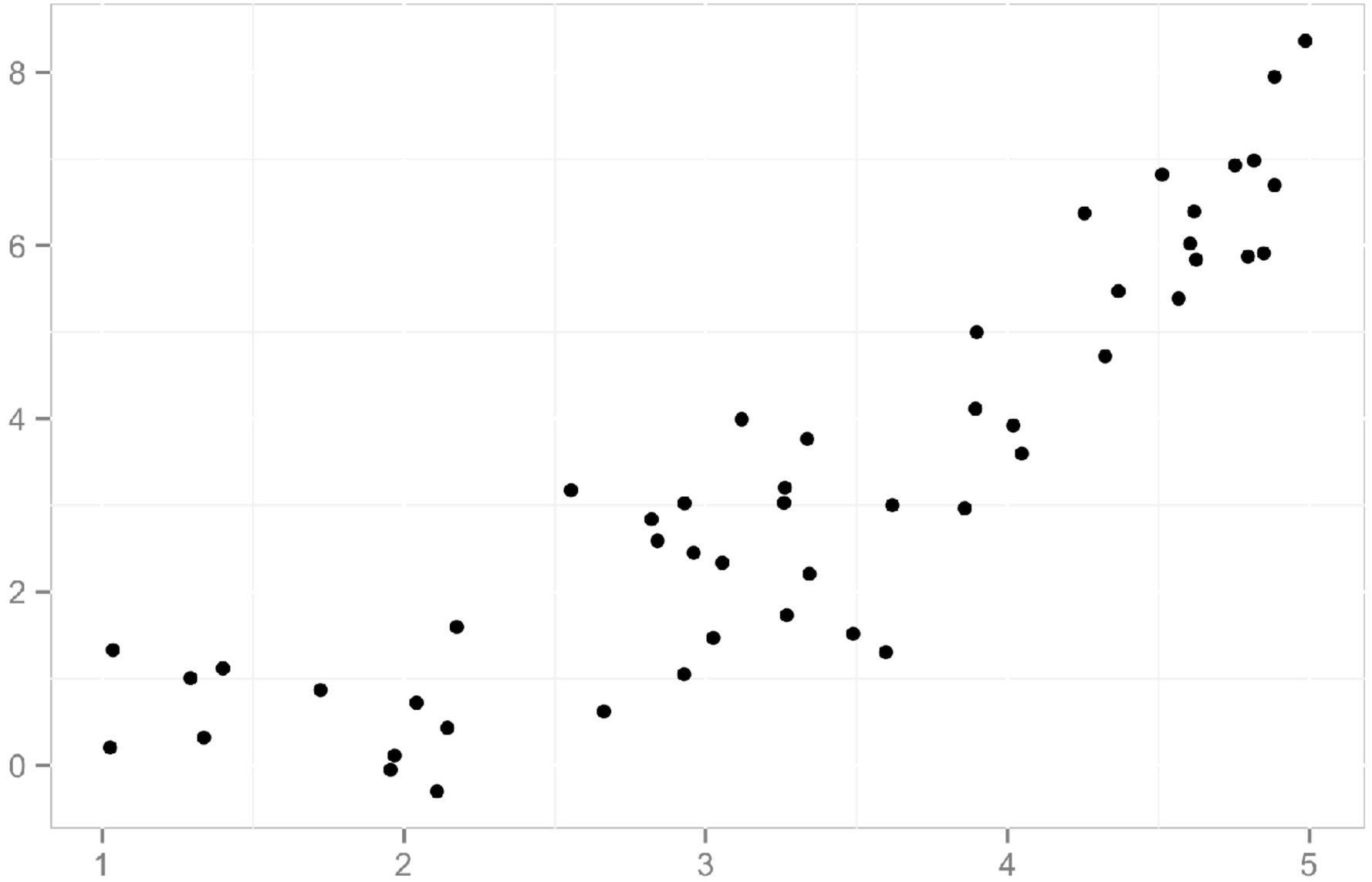
…Define the model beforehand

Simple Linear Regression

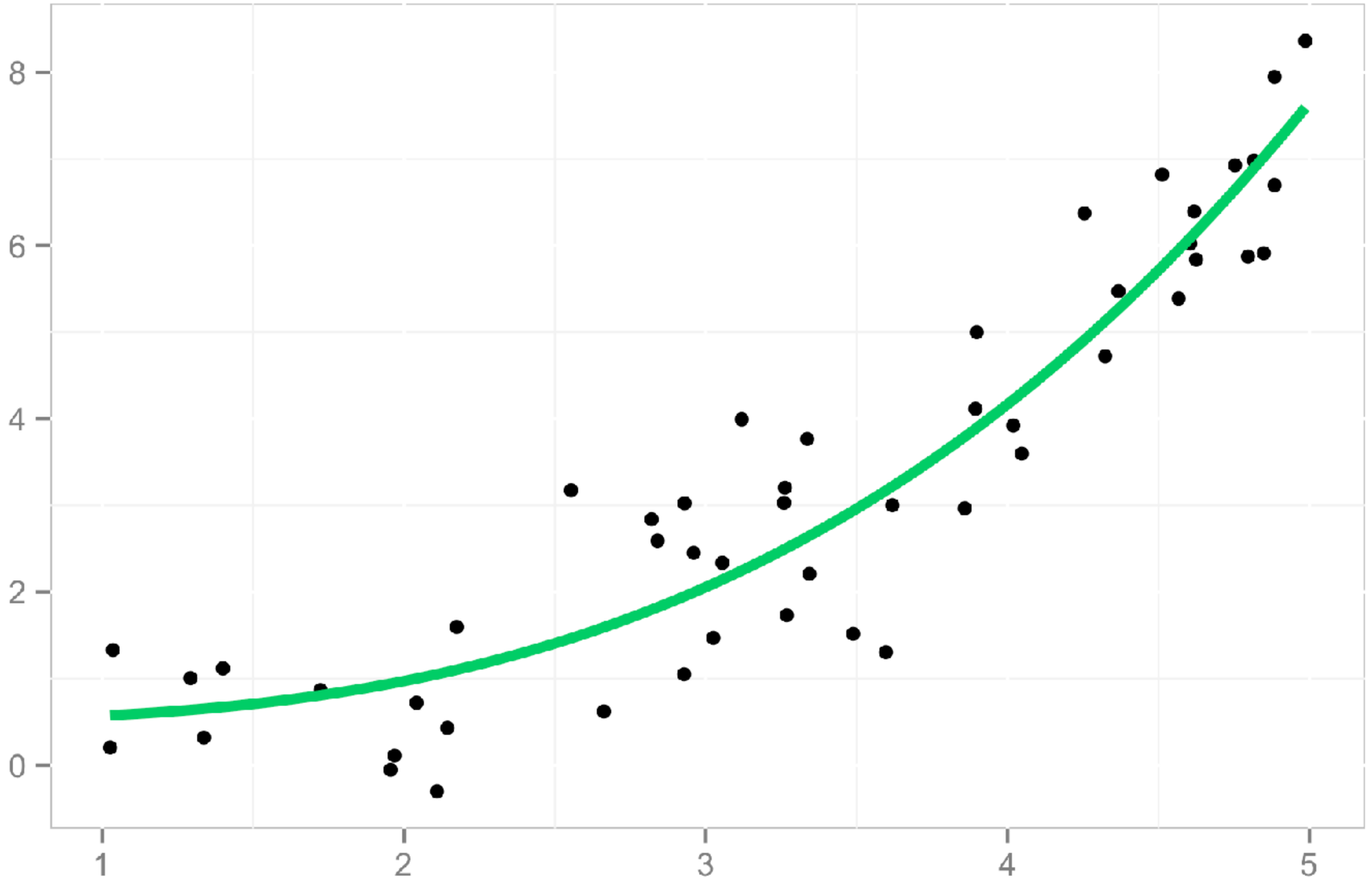


…Define the model beforehand

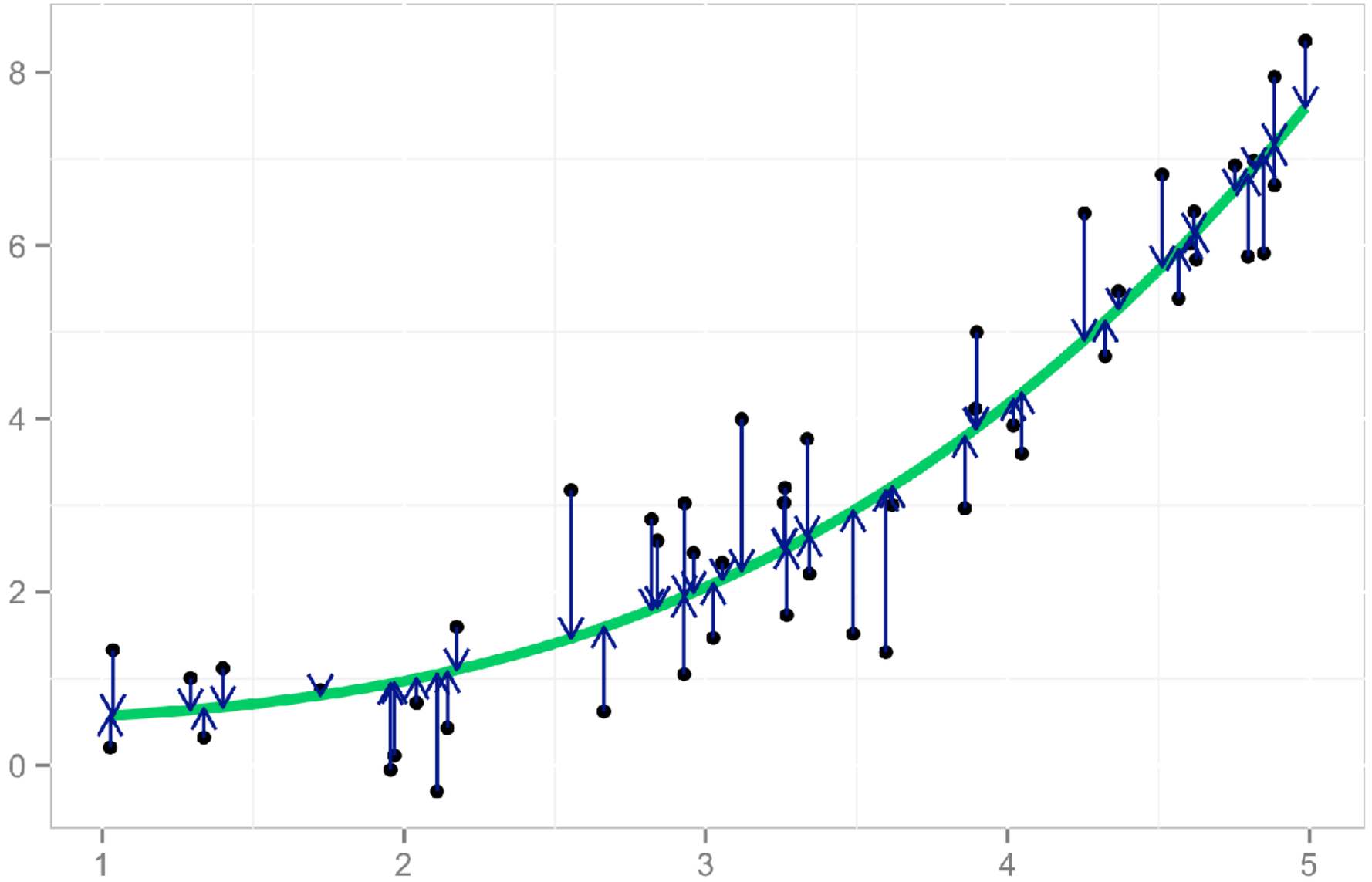
Linear Regression



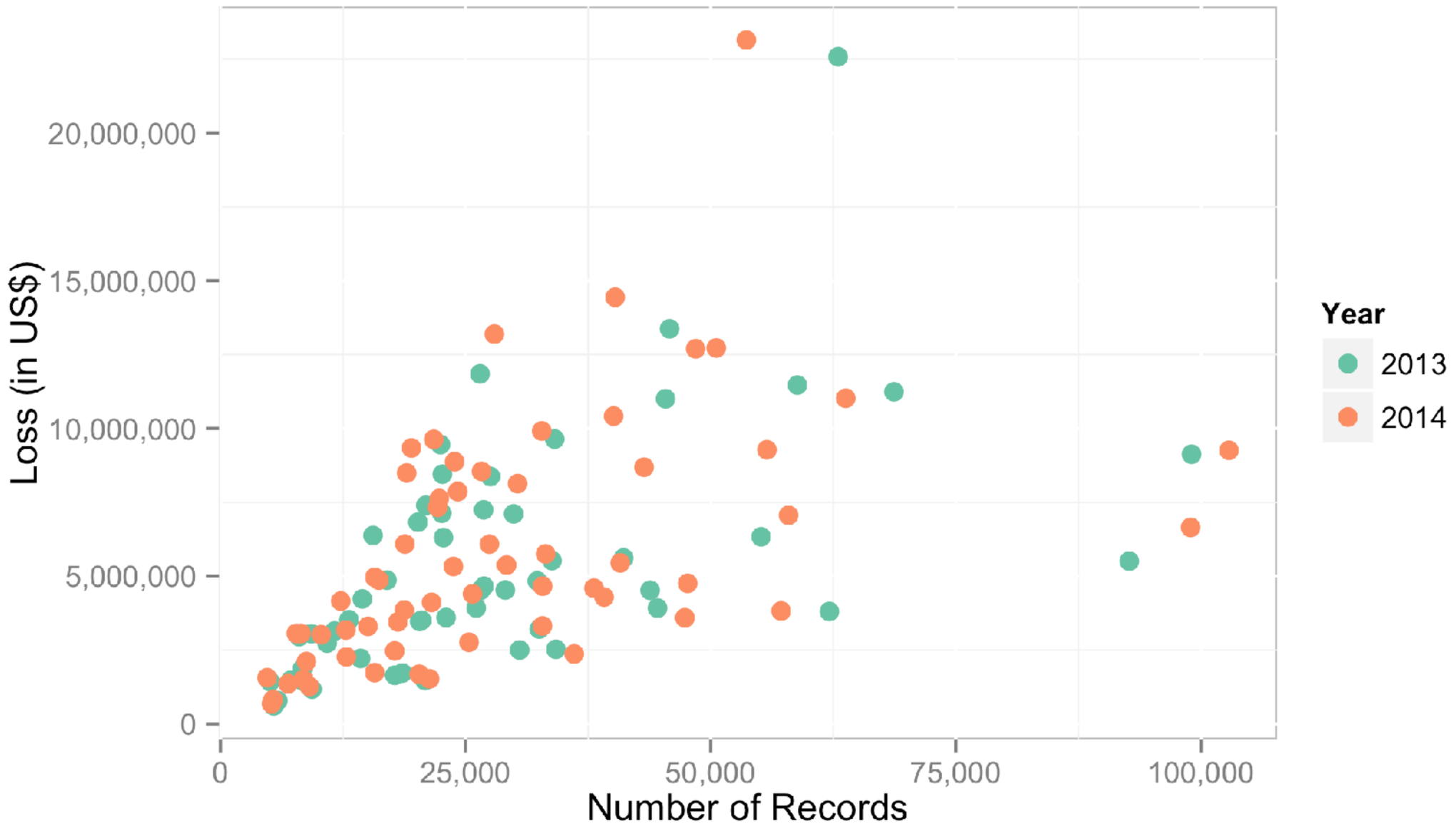
Linear Regression



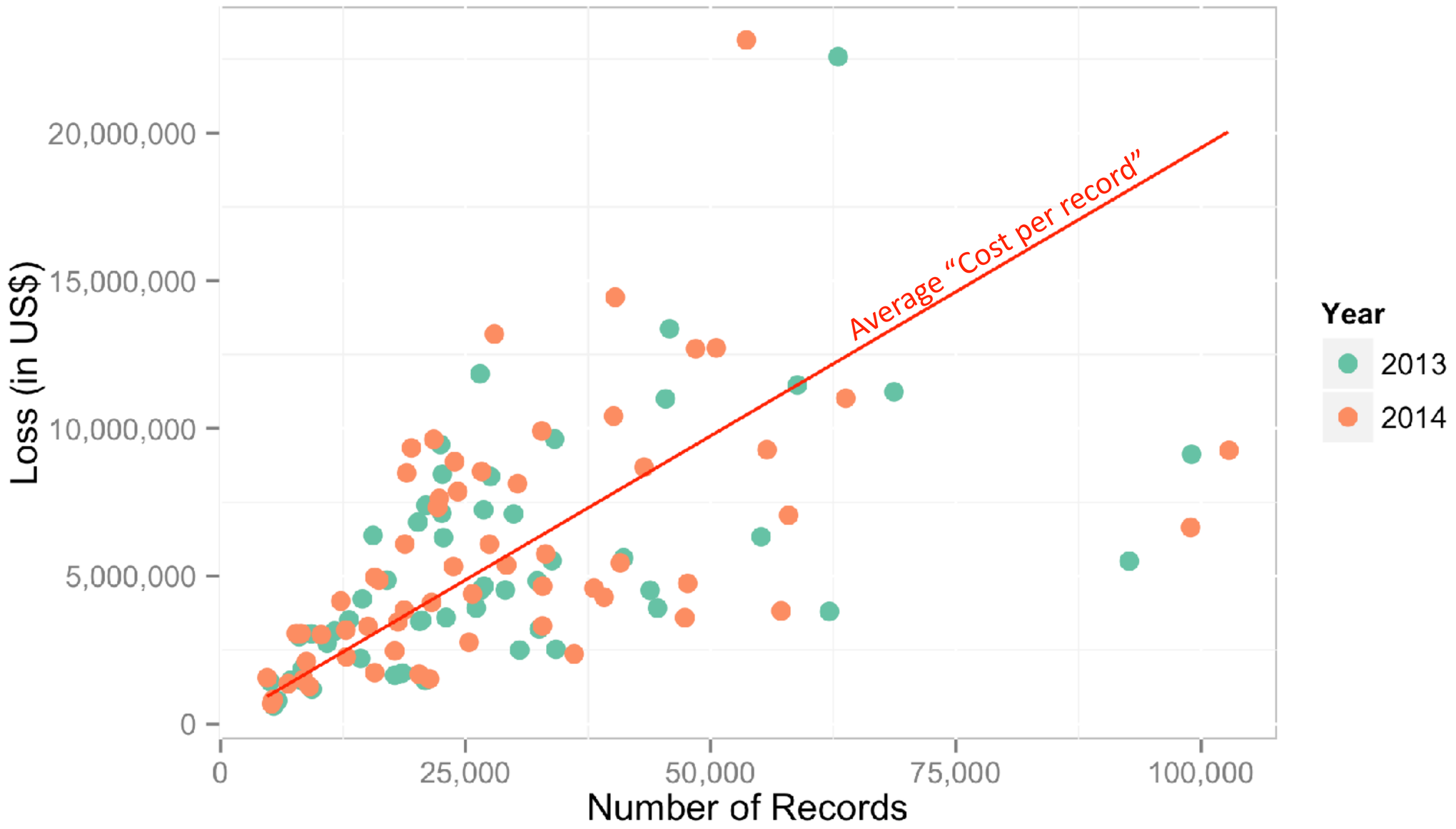
Linear Regression



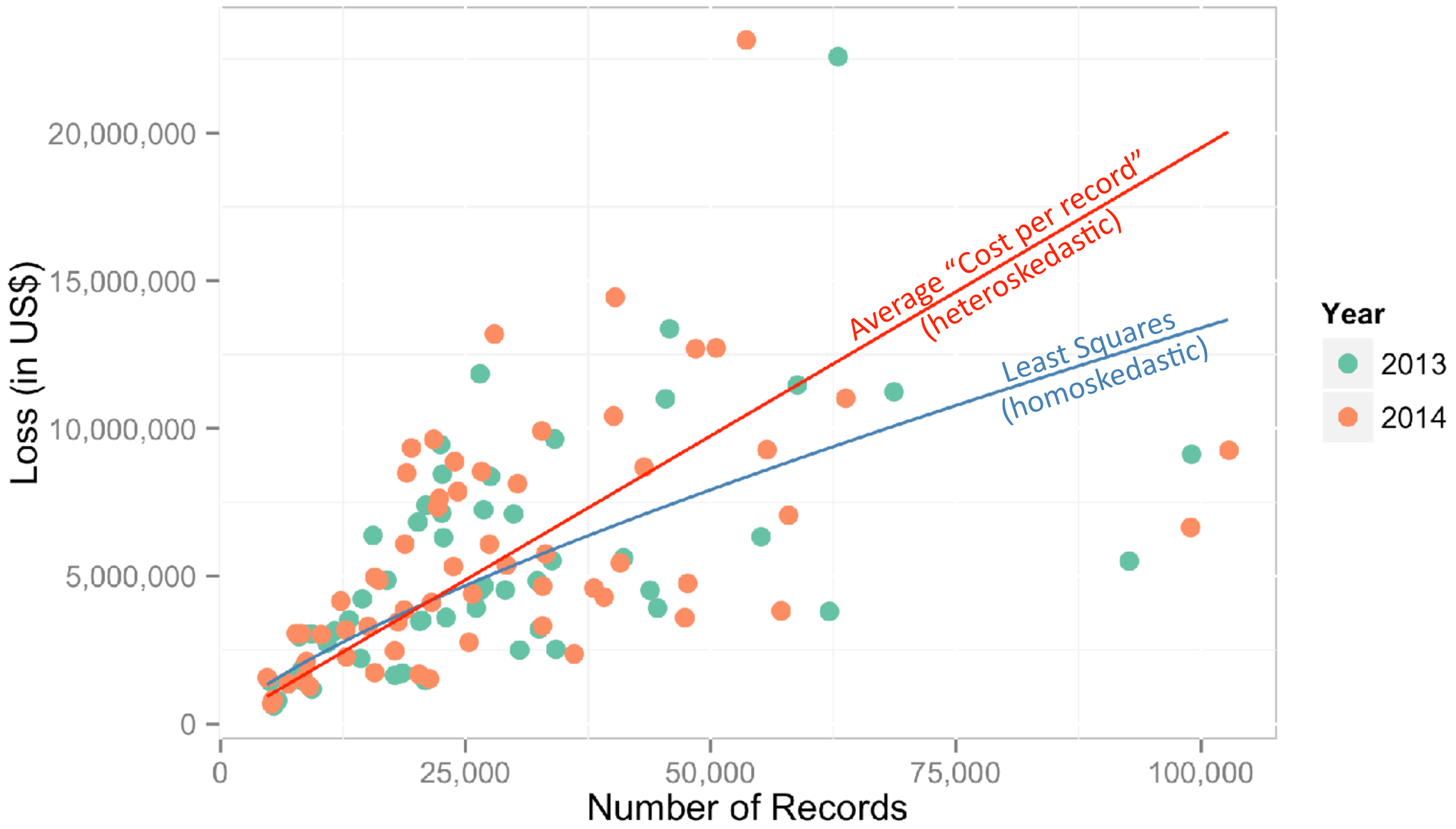
Cost per Record



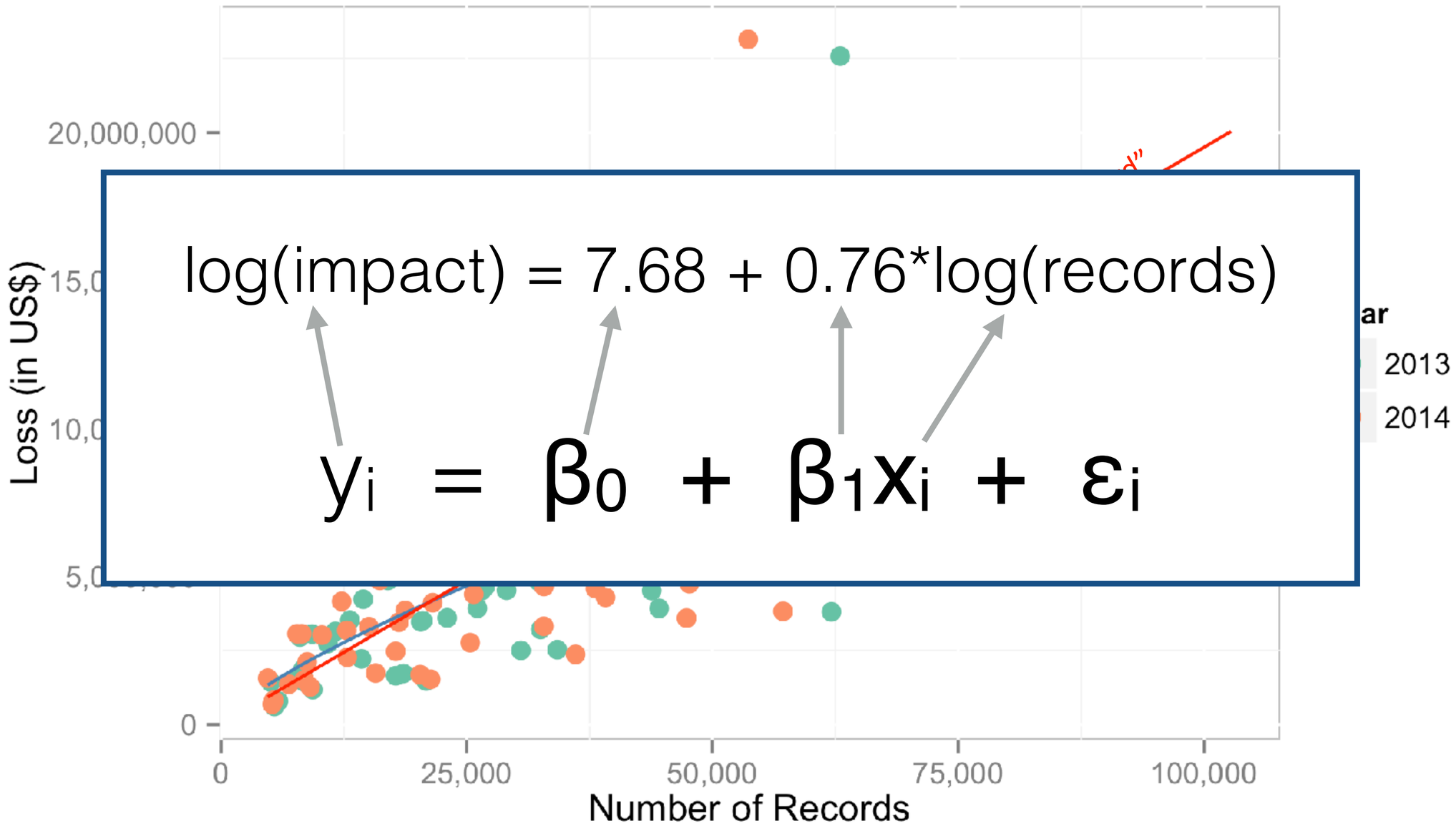
Cost per Record



Cost per Record



Cost per Record



$$\log(\text{impact}) = 7.68 + 0.76 * \log(\text{records})$$

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

linear regression in R

```
> lm(log(total) ~ log(records), data=ponemon)
##
## Call:
## lm(formula = log(total) ~ log(records), data = ponemon)
##
## Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0243 -0.3792  0.0204  0.4197  1.0188
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.6800     0.7013    10.9   <2e-16 ***
## log(records)   0.7584     0.0697    10.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.523 on 113 degrees of freedom
## Multiple R-squared:  0.512,  Adjusted R-squared:  0.508
## F-statistic:  119 on 1 and 113 DF,  p-value: <2e-16
```

Linear Regression in R

```
> lm(log(total) ~ log(records), data=ponemon)
##
## Call:
## lm(formula = log(total) ~ log(records), data = ponemon)
##
## Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0243 -0.3792  0.0204  0.4197  1.0188
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.6800     0.7013    10.9   <2e-16 ***
## log(records)   0.7584     0.0697    10.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.523 on 113 degrees of freedom
## Multiple R-squared:  0.512,  Adjusted R-squared:  0.508
## F-statistic:  119 on 1 and 113 DF,  p-value: <2e-16
```

Look for uncorrelated, constant variance, mean=0,

Coefficients

P-Values of independent variables

R^2 value

P-Value of model

Linear Regression in Python

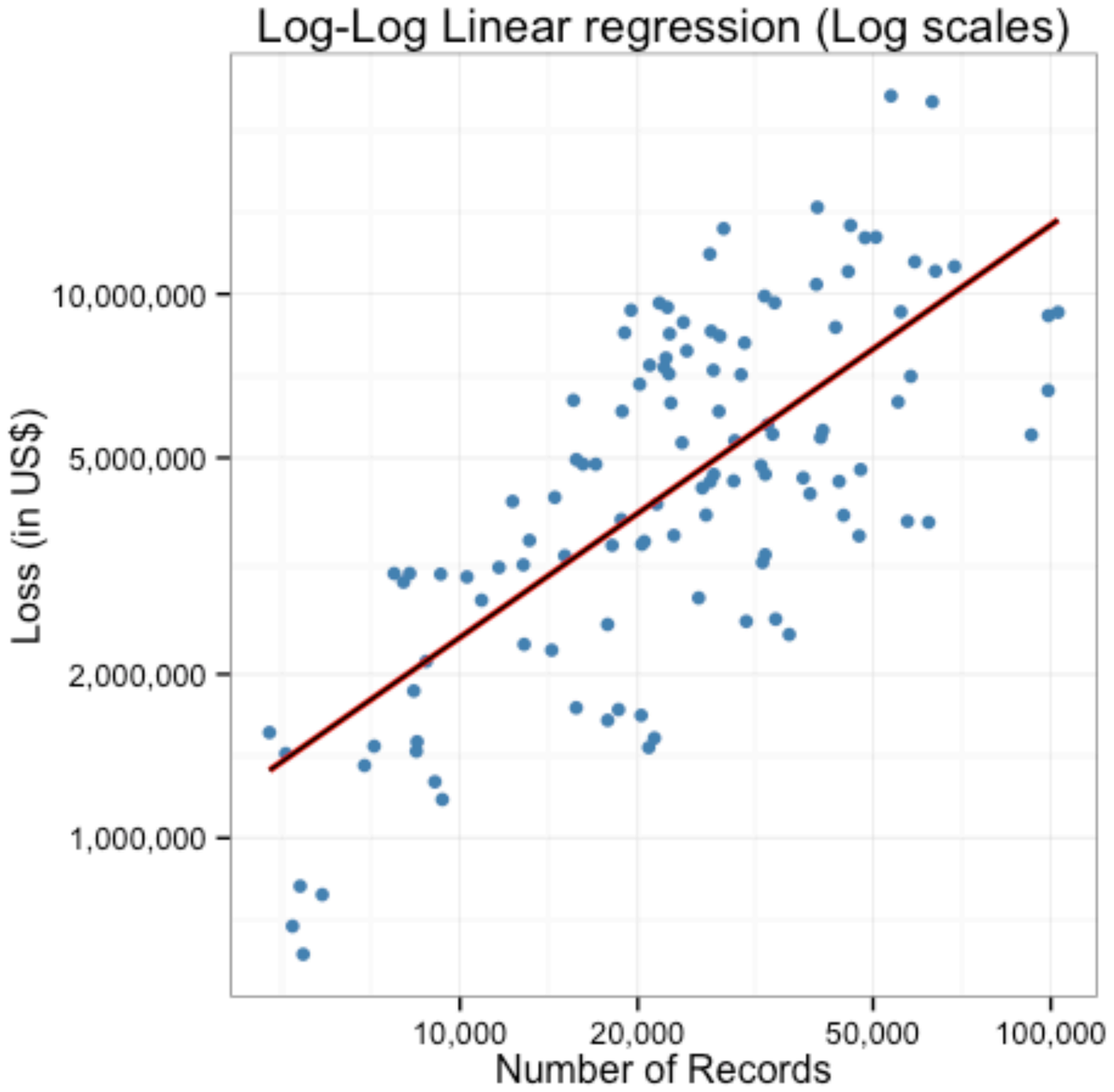
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(training_feature_matrix, training_target_vector)

# Make predictions using the testing set
predictions = regr.predict(testing_feature_matrix)

# The coefficients
print('Coefficients: \n', regr.coef_)

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(testing_target_vector, predictions))

# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(testing_target_vector, predictions))
```

http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

Cost per Record

log(impact) =
7.68 + 0.76*log(records)

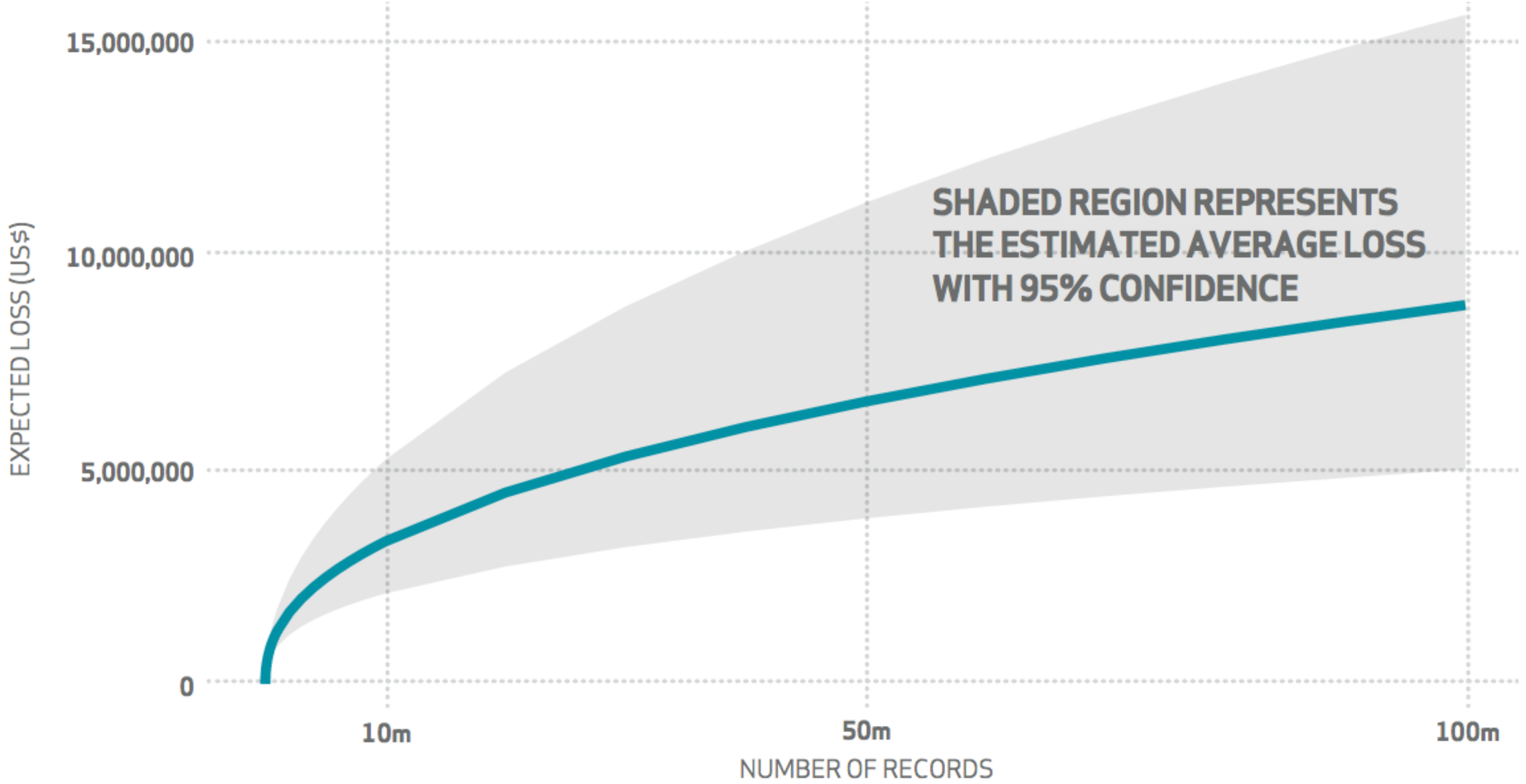
For every 1% increase in records, there is a 0.76% increase in loss.



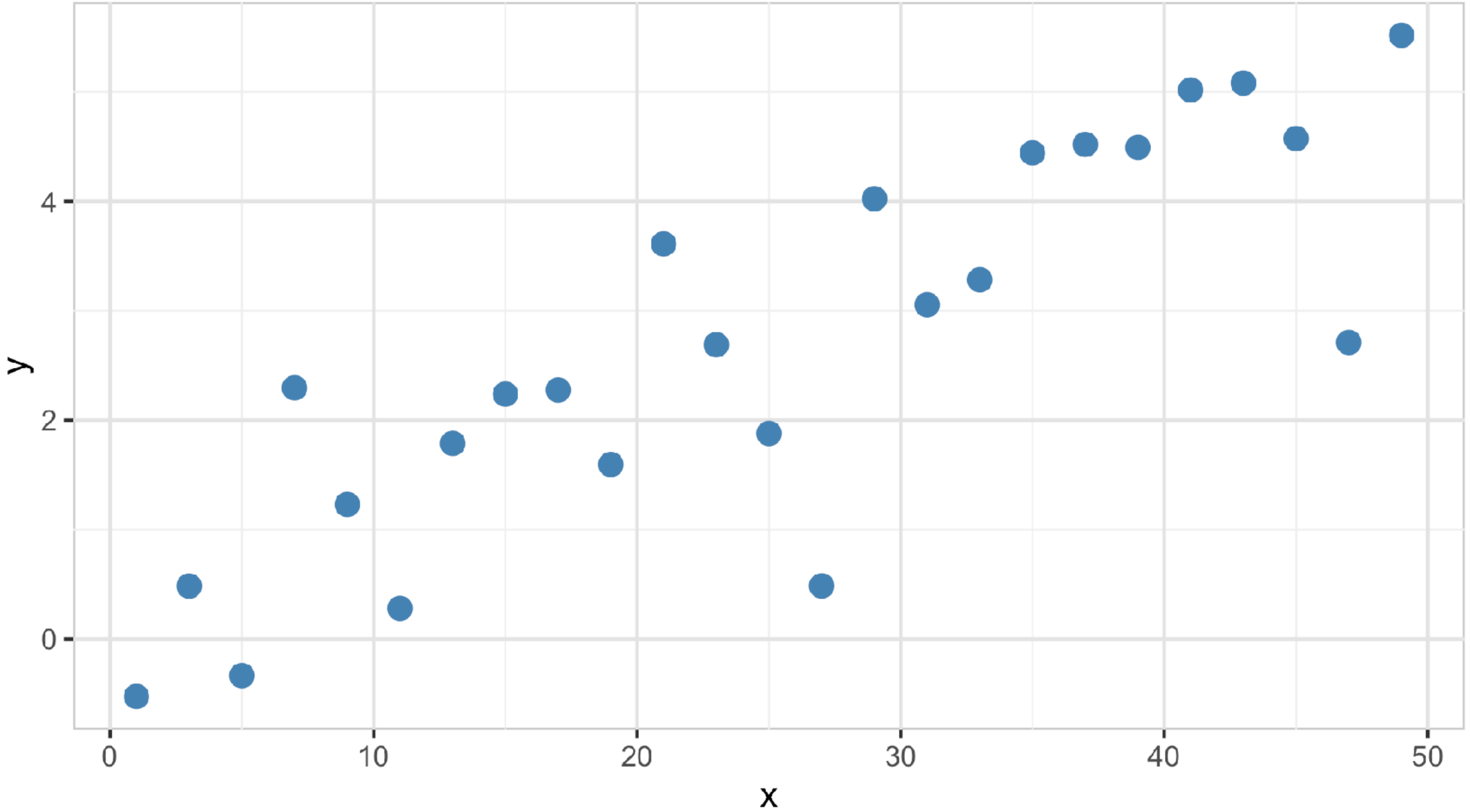
http://datadrivensecurity.info/blog/posts/2014/Dec/ponemon/

Different data, Still a log-log model

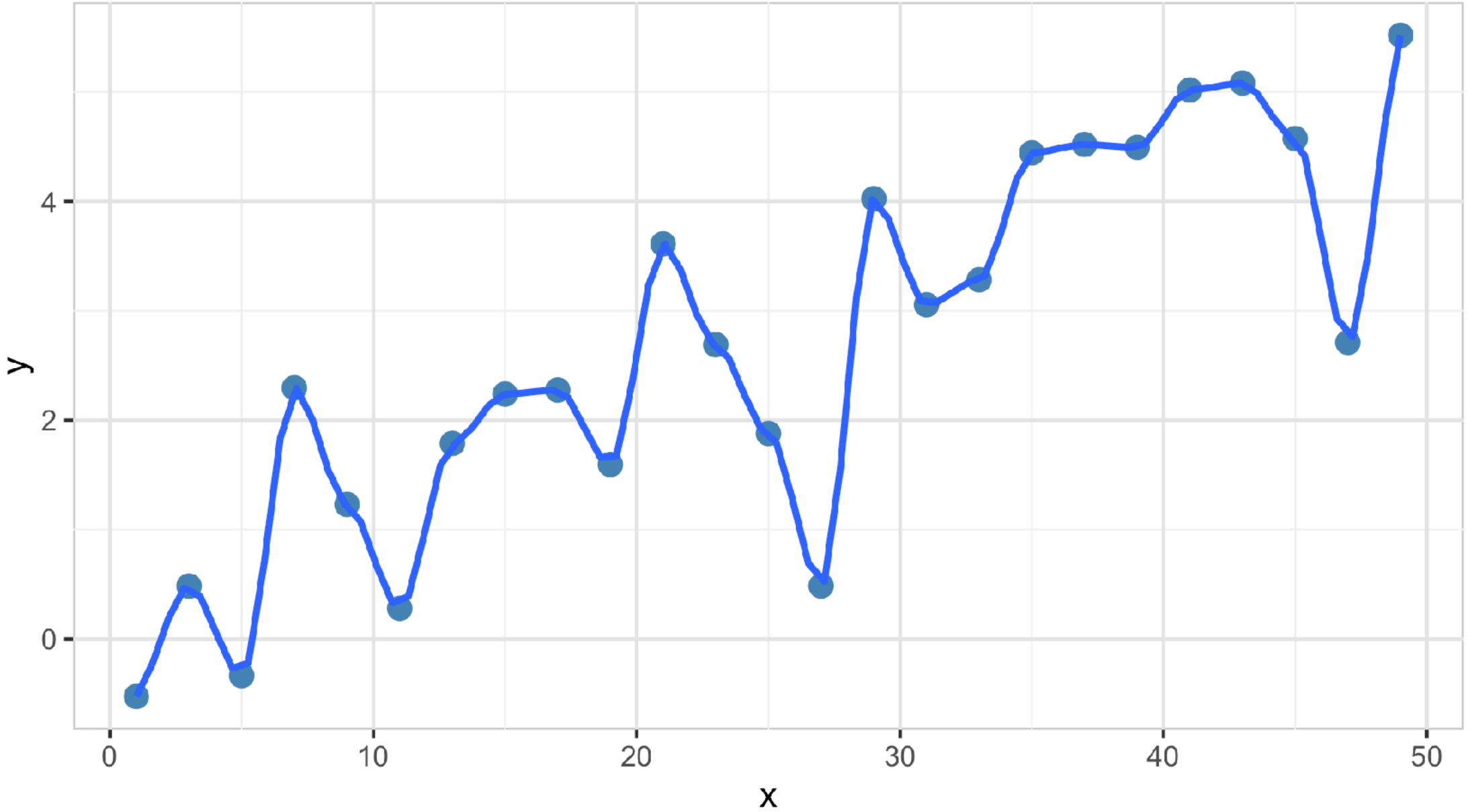
"…there's a lot of stuff contributing to the cost of breaches besides the number of records lost."



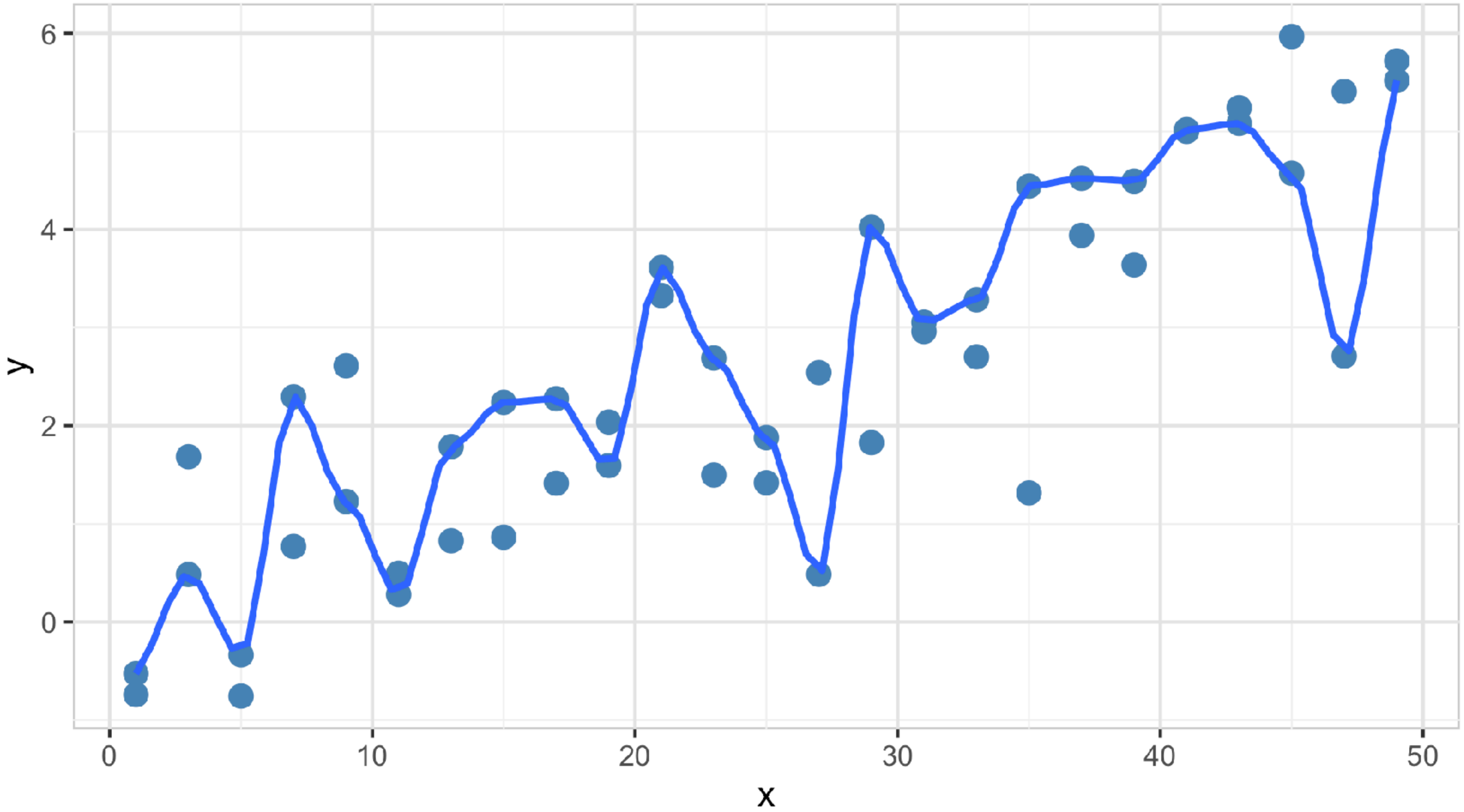
The challenge of over fitting



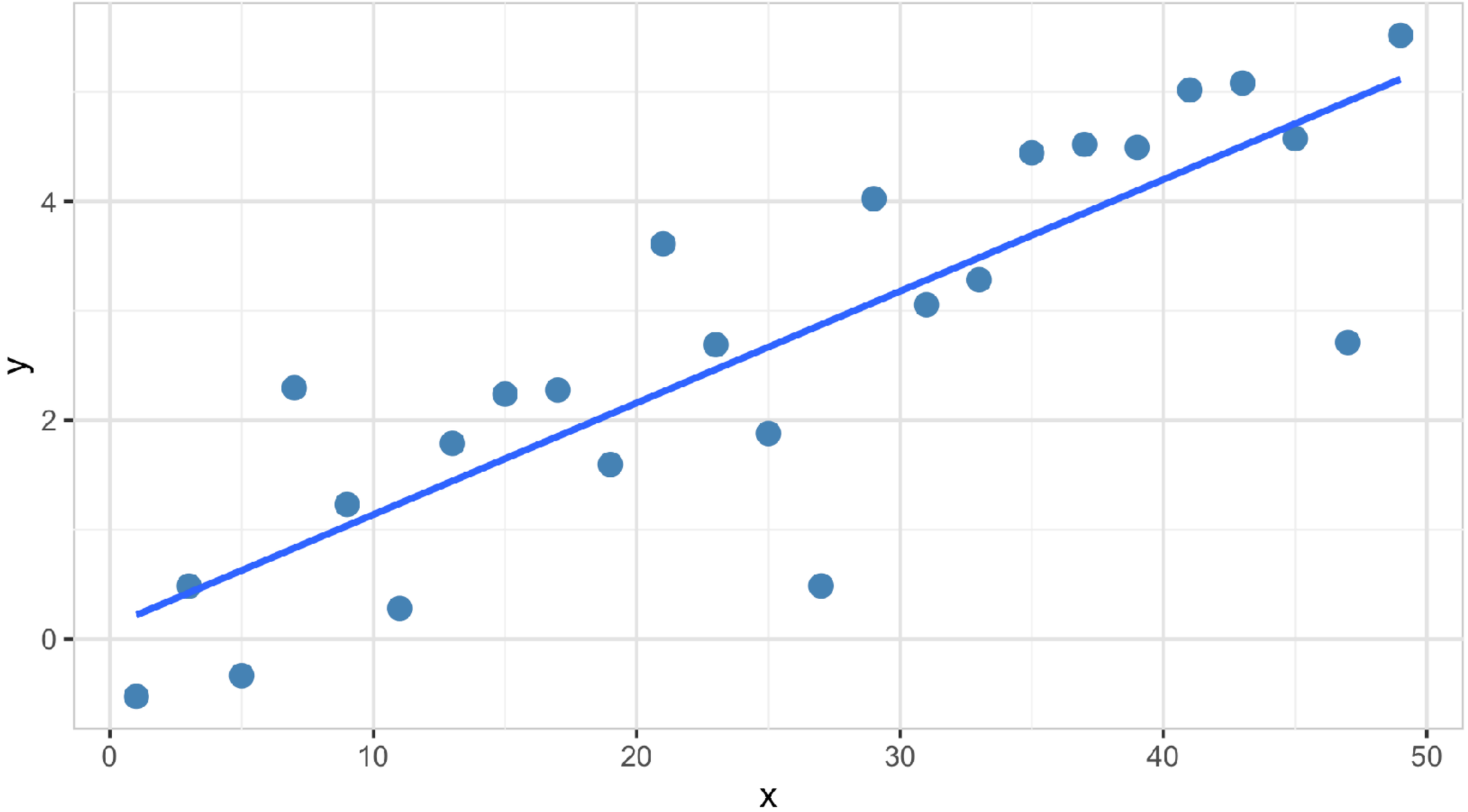
The challenge of over fitting



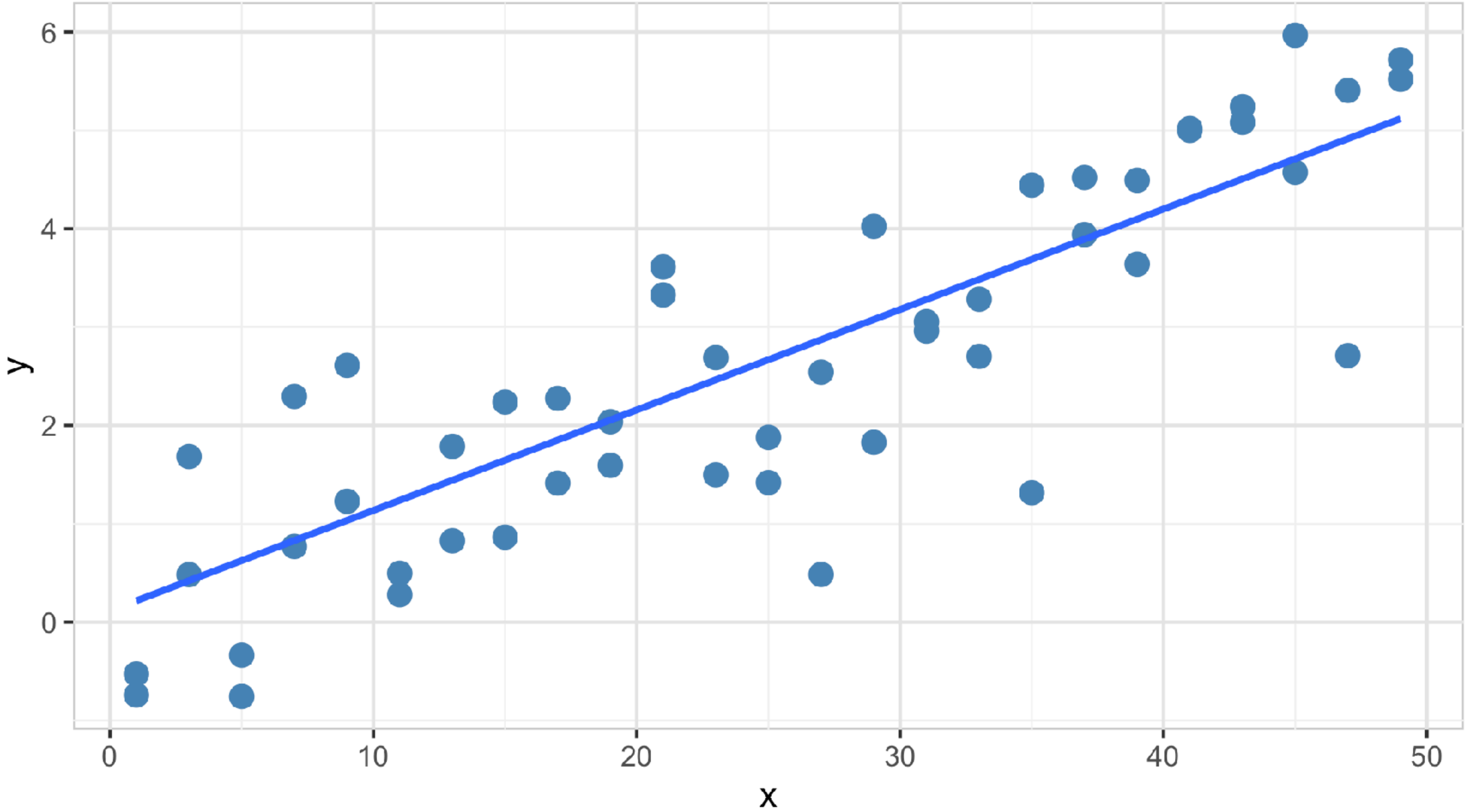
The challenge of over fitting



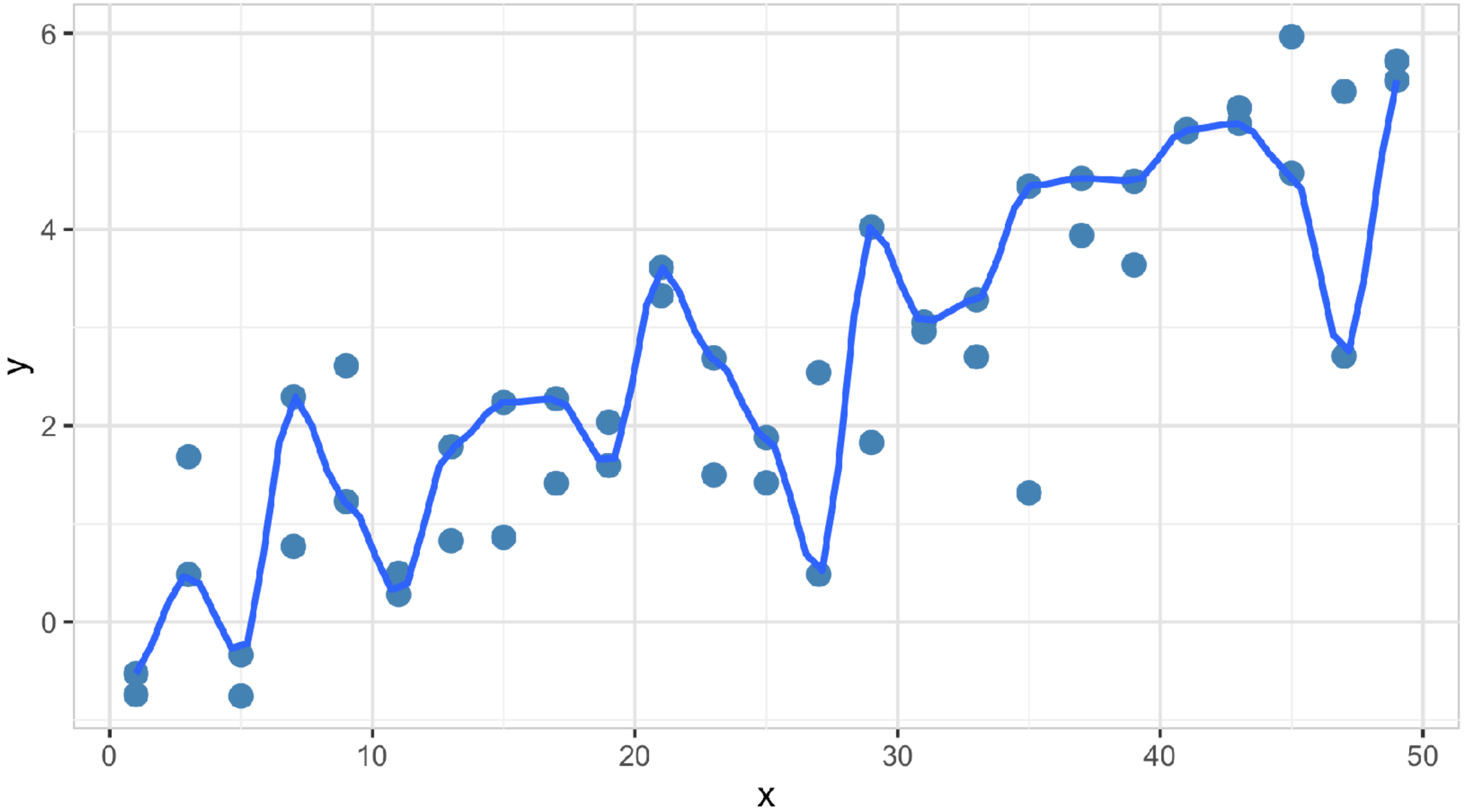
The challenge of over fitting



The challenge of over fitting



The challenge of over fitting



How to screw-up regression analysis

- Non-Linear relationships
  - Assumes relationship is linear and additive
- Skewed residuals
  - normality of the error distribution
- Variables are not Independent
  - should have little to no multicollinearity
- Homoscedasticity
  - Variance of error is consistently constant

O'REILLY®
Security
BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

Training and Testing

Reducing Overfitting

Question:

How could we avoid overfitting a model?

Answer:

Reducing Overfitting

Question:

How could we avoid overfitting a model?

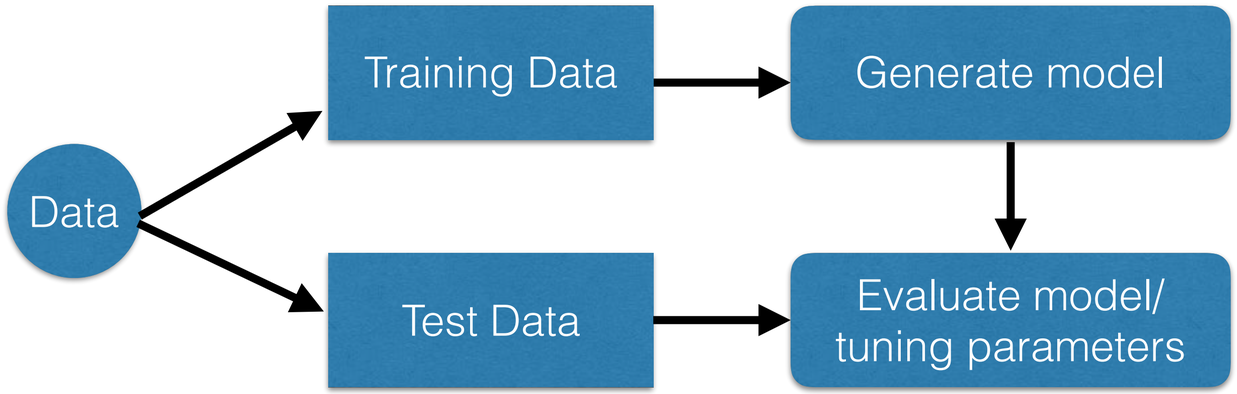
Answer:

Evaluate the performance of model with new data
(or data that was not used to train the model)

Data Splitting

Create two data sets by randomly assigning observations into either a training or testing data set.

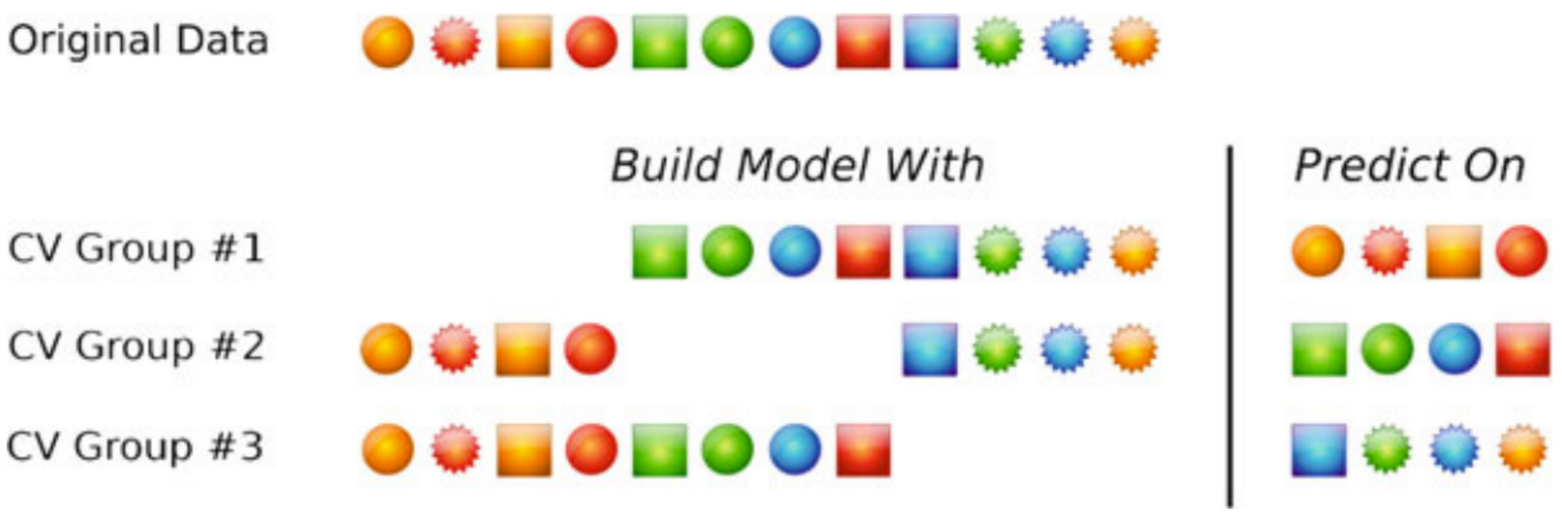
- **Training Data**: observations used to produce a model
- **Test Data**: observations used to measure model performance and set tuning parameters



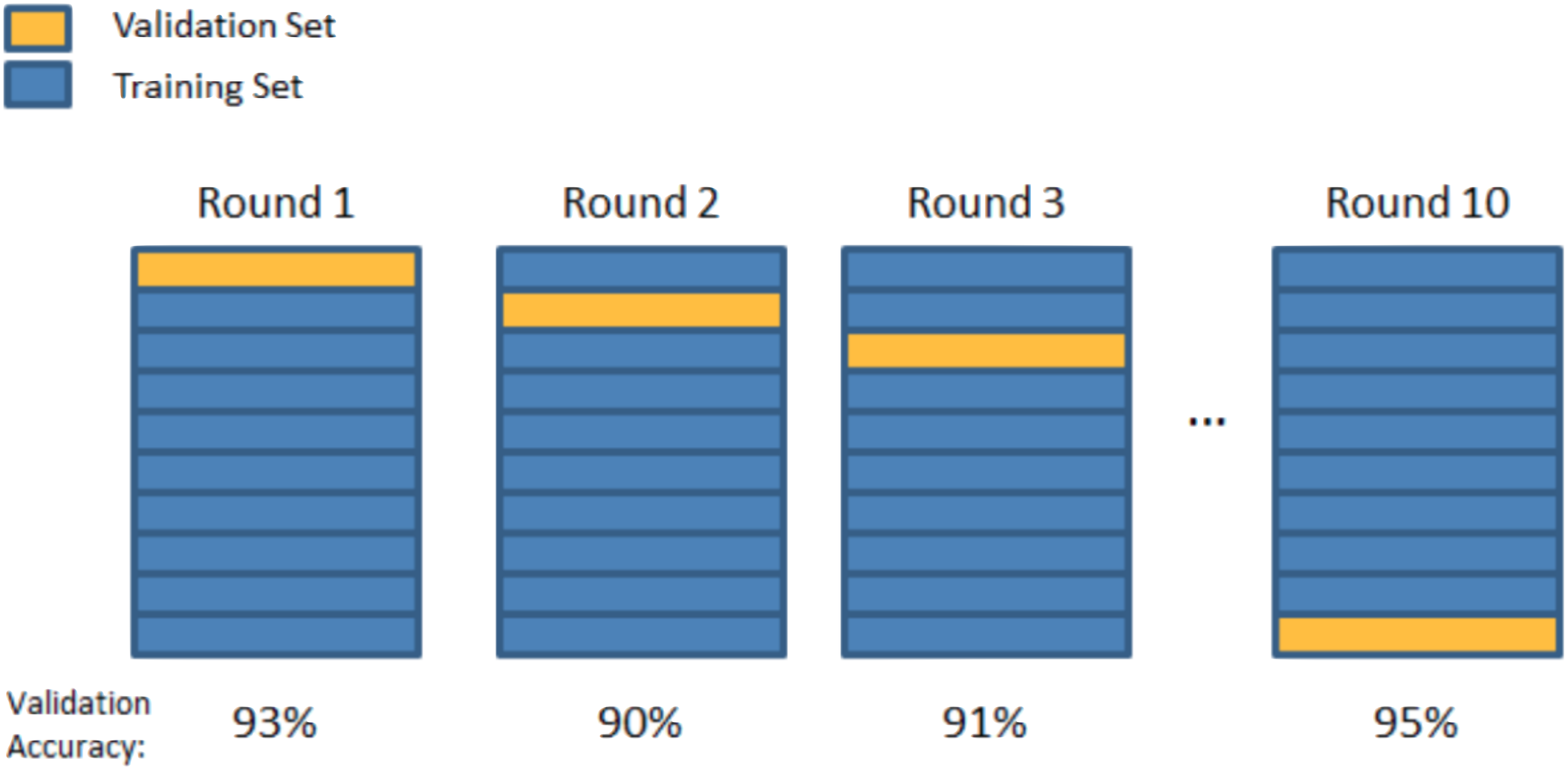
K-Fold Cross-Validation

Cross-Validation is useful to compare models or find tuning parameters

- Data is partitioned into K sets of roughly equal size
- loop through k-times training with the main set, testing with the hold out set
- when k = n, “leave-one-out” cross-validation



K-Fold Cross-Validation



Other ways to (not) split

Bias may be introduced with non-random samples, but some alternatives:

- Split on some element of time
  - If collected by days, use all but the last day to train and test on the last day.
- Split on another variable (example: data from departments)

A Non-Random split makes the **assumption** that there is no difference between whatever is being split upon.

(e.g. if all the data was collected on weekdays, but the last day is a weekend, or departments have differences in the data collected)

Missing Values

Security Data is generally EXTREMELY messy and is often a "convenience" sample, there will be **missing values**…

- See if you can figure *why* they are missing
  - Indication of larger problem
  - Maybe missing values are informative (missing p2p)
- Ways to deal with missing data:
  - Leave it missing (some models deal with [NA/None] well)
  - Drop incomplete cases within the data
  - *Impute* new data (create a model for missing data) (knn)

Other Considerations

- Zero-Variance Predictors (or NZV)
- Collinearity and Multicollinearity
- Dummy Variables (factors in R)
- Adding/Deriving variables
- Binning (resist the urge)

O'REILLY®

Security

BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

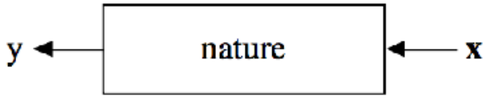
Overview of supervised learning

Statistical Modeling: The Two Cultures

Leo Breiman

1. INTRODUCTION

Statistics starts with data. Think of the data as being generated by a black box in which a vector of input variables **x** (independent variables) go in one side, and on the other side the response variables **y** come out. Inside the black box, nature functions to associate the predictor variables with the response variables, so the picture is like this:



There are two goals in analyzing the data:

Prediction. To be able to predict what the responses are going to be to future input variables;
Information. To extract some information about how nature is associating the response variables to the input variables.

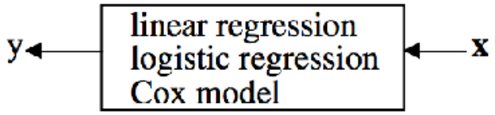
There are two different approaches toward these goals:

The Data Modeling Culture

The analysis in this culture starts with assuming a stochastic data model for the inside of the black box. For example, a common data model is that data are generated by independent draws from

response variables = f(predictor variables, random noise, parameters)

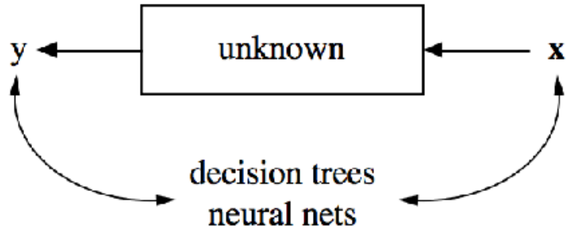
The values of the parameters are estimated from the data and the model then used for information and/or prediction. Thus the black box is filled in like this:



Model validation. Yes–no using goodness-of-fit tests and residual examination.
Estimated culture population. 98% of all statisticians.

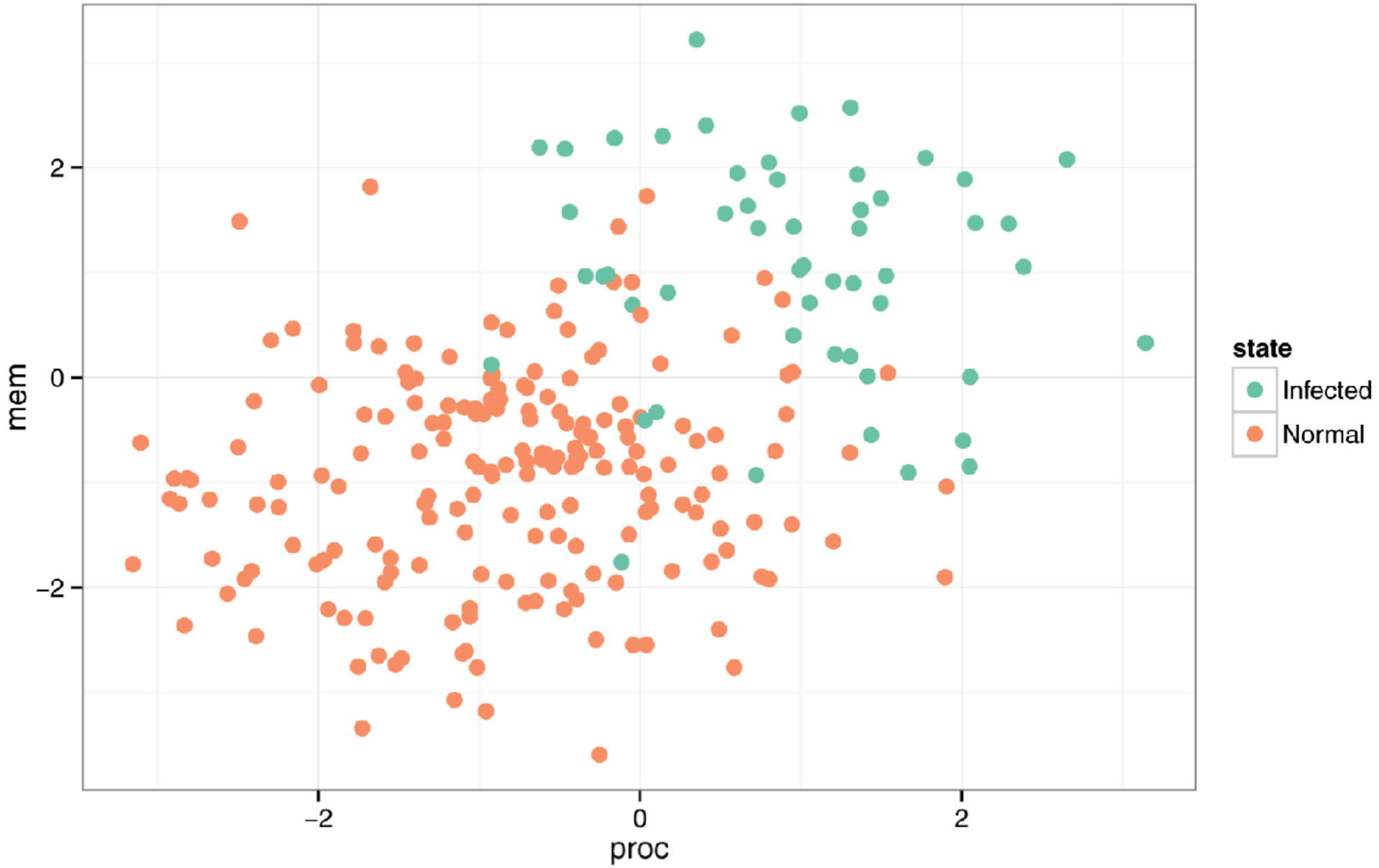
The Algorithmic Modeling Culture

The analysis in this culture considers the inside of the box complex and unknown. Their approach is to find a function $f(\mathbf{x})$—an algorithm that operates on **x** to predict the responses **y**. Their black box looks like this:

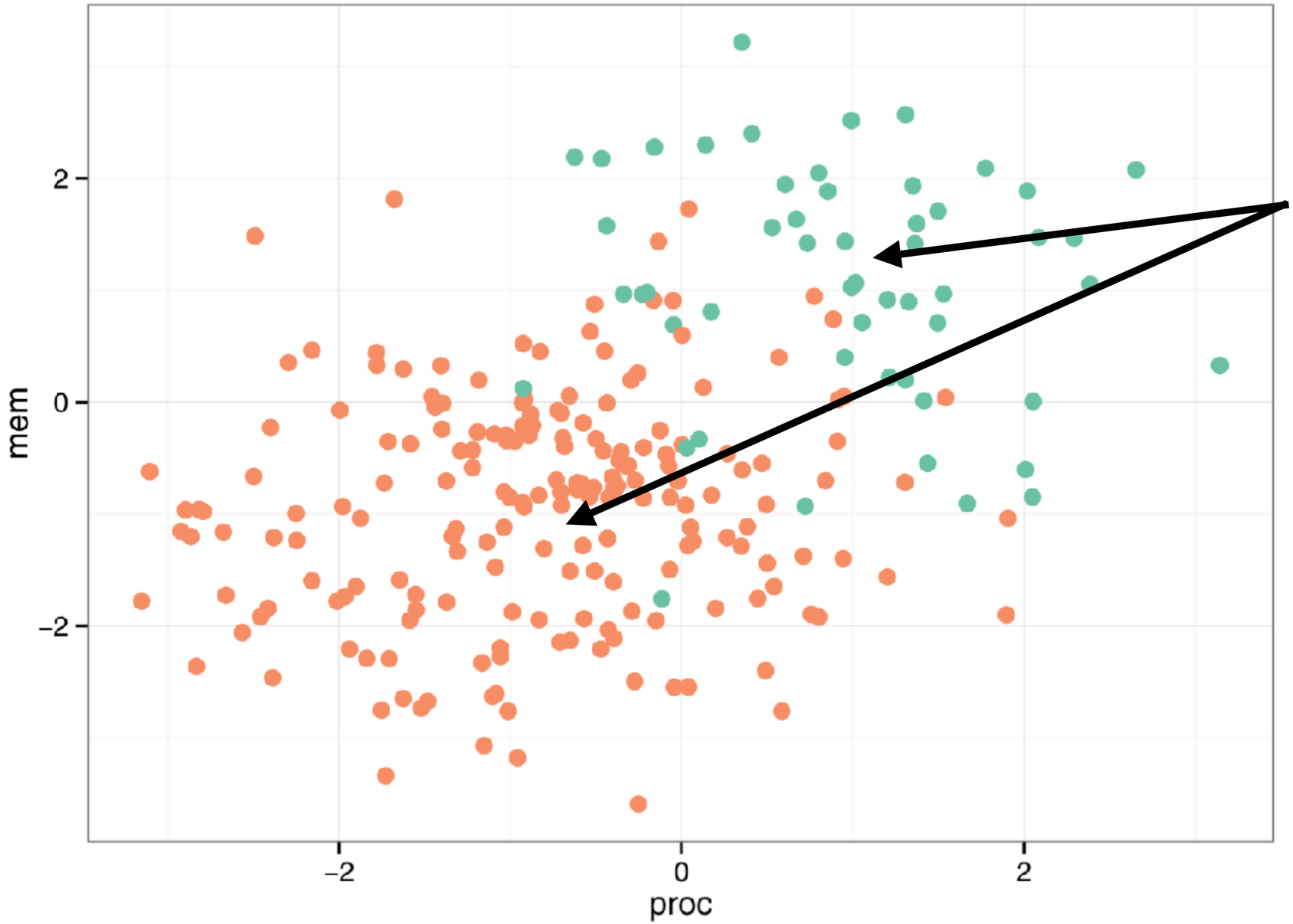


Model validation. Measured by predictive accuracy.
Estimated culture population. 2% of statisticians, many in other fields.

Teaching the machine to learn



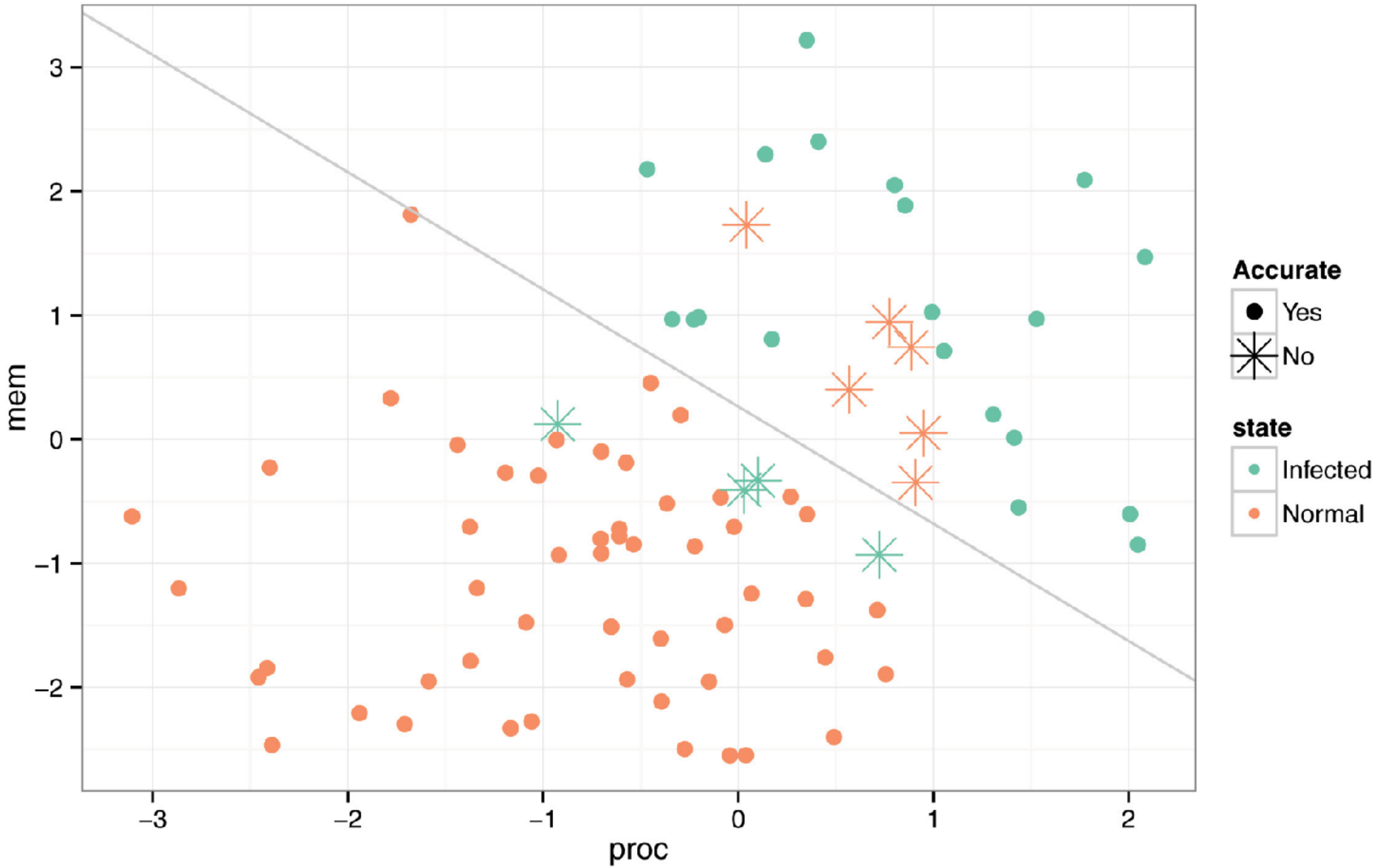
Teaching the machine to learn



1) Get mean of *proc* and *mem* for both classes

2) Assign class based on which centroid is closer

Teaching the machine to learn…



Nearest Neighbors

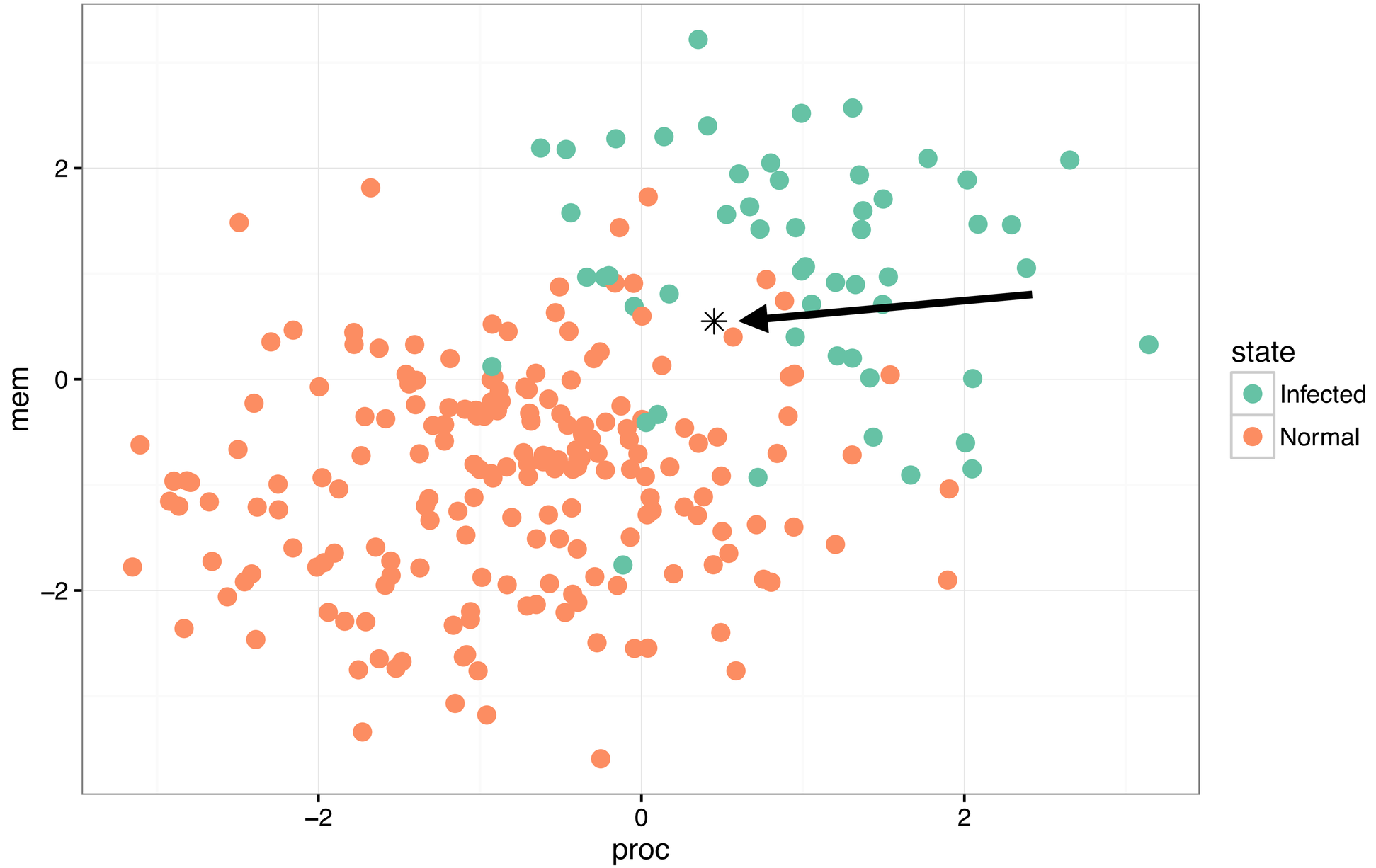
What if rather than look at all of the variables as a group, we just assume that any unknown value will probably be like its neighbors?

Example: k-nearest neighbors

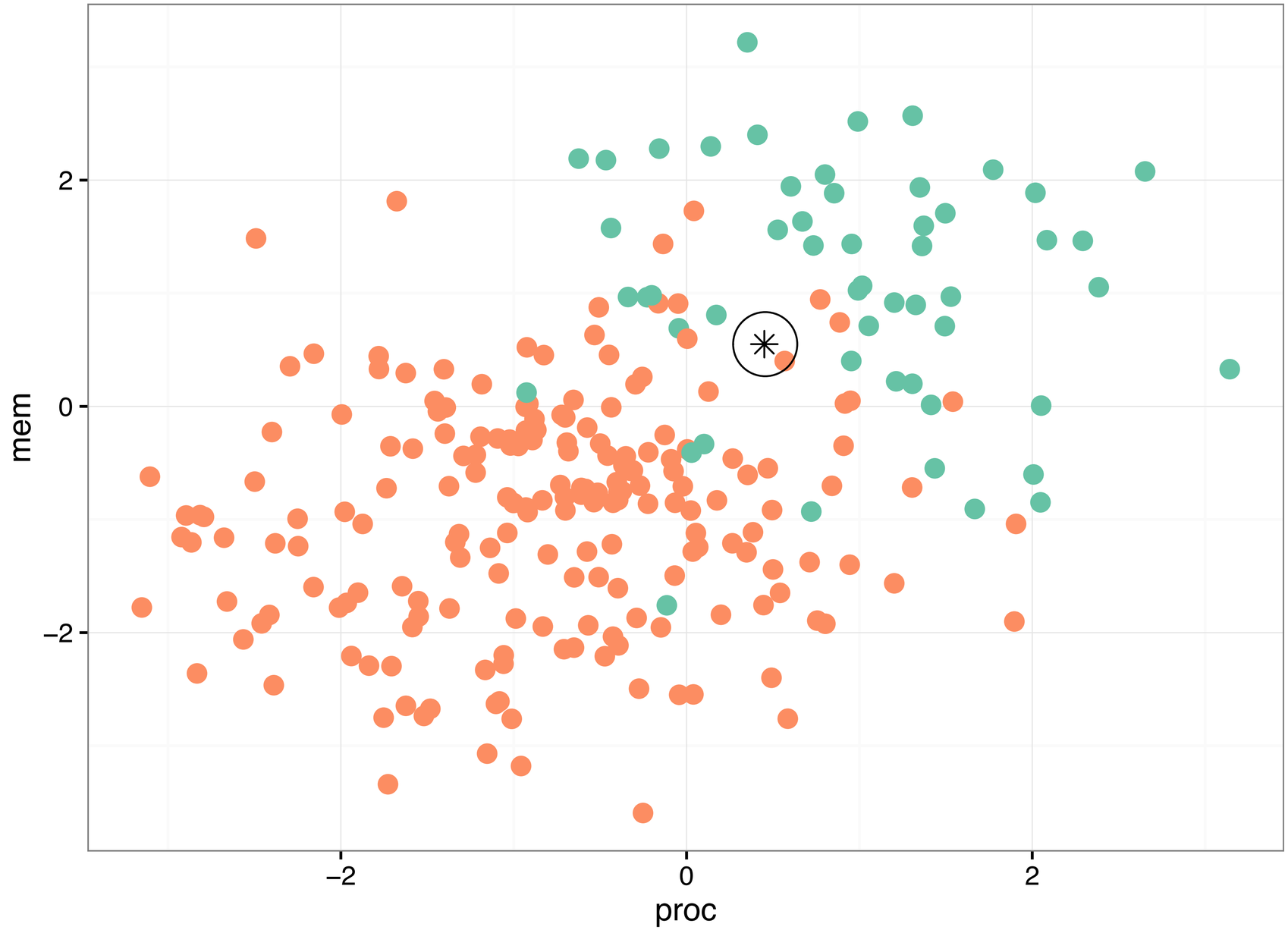
Define K, and for every unknown observation:

1. Find k-nearest neighbors (by *distance*)
2. Assign class based on dominate class
3. Optional weight by distance

k-nearest neighbors



k-nearest neighbors

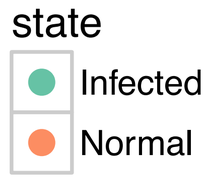


k = 1

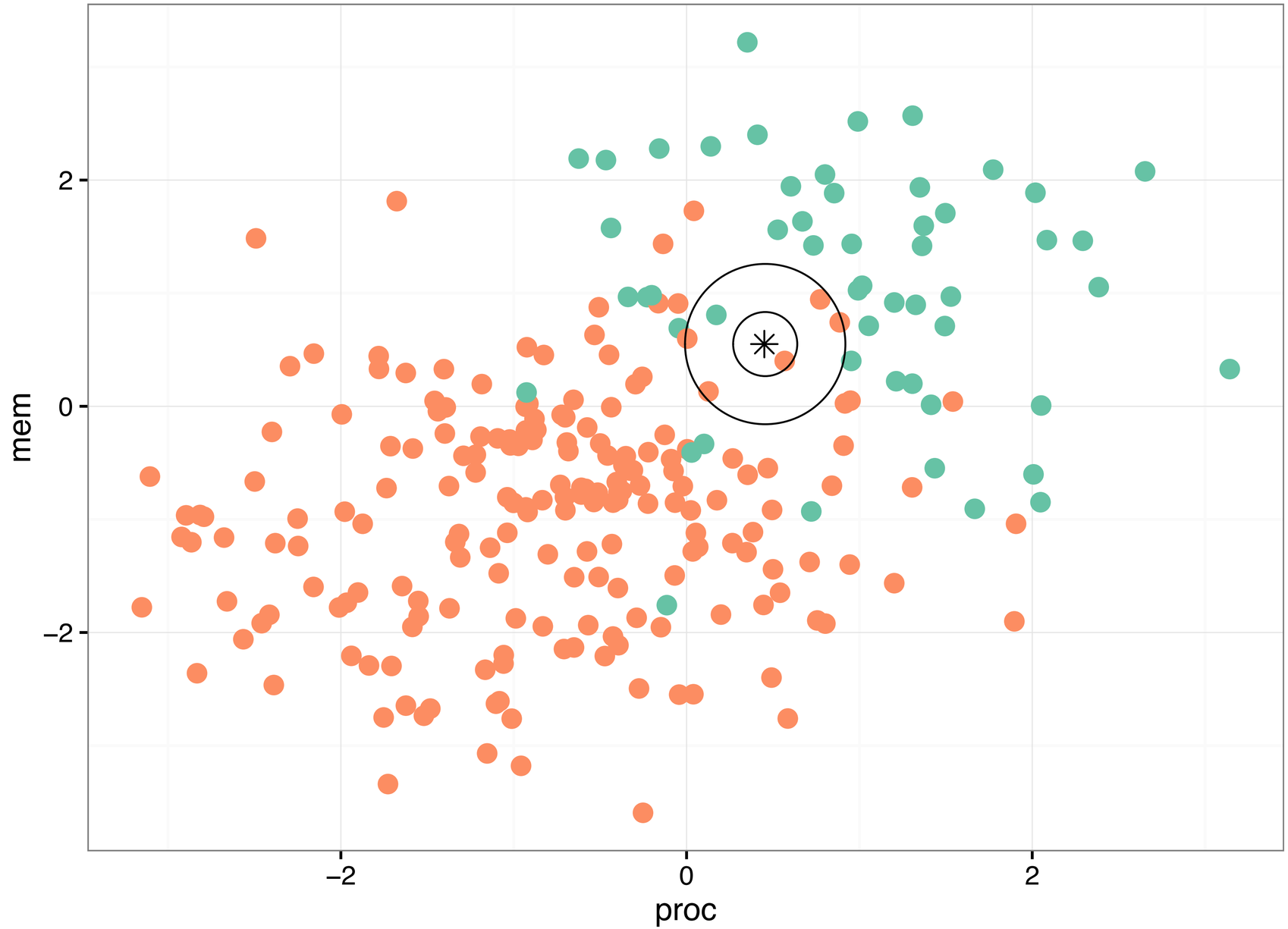
1 Normal

0 Infected

Assign “Normal”



k-nearest neighbors

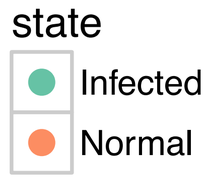


k = 5

4 Normal

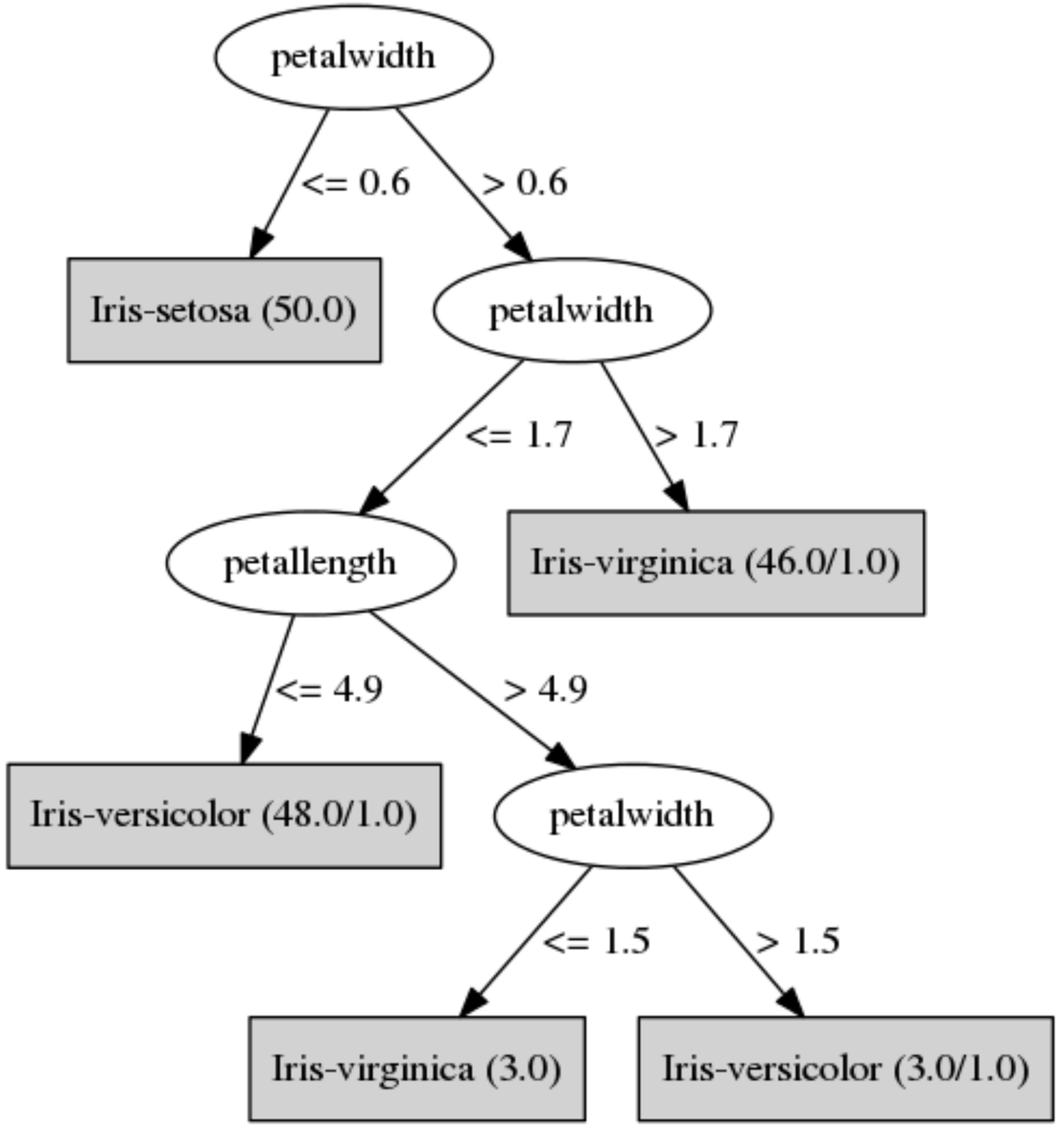
1 Infected

Assign “Normal”

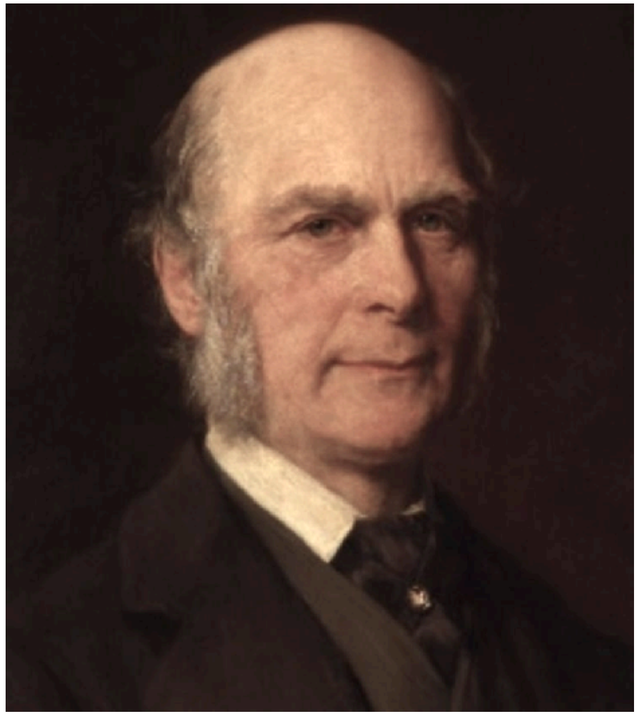


Decision Tree

- “uses a tree-like graph or model of decisions and their possible consequences”
- Use one or more variable, one or more branches
- Can split on continuous or categorical variable
- Final result is probability (between 0-1)

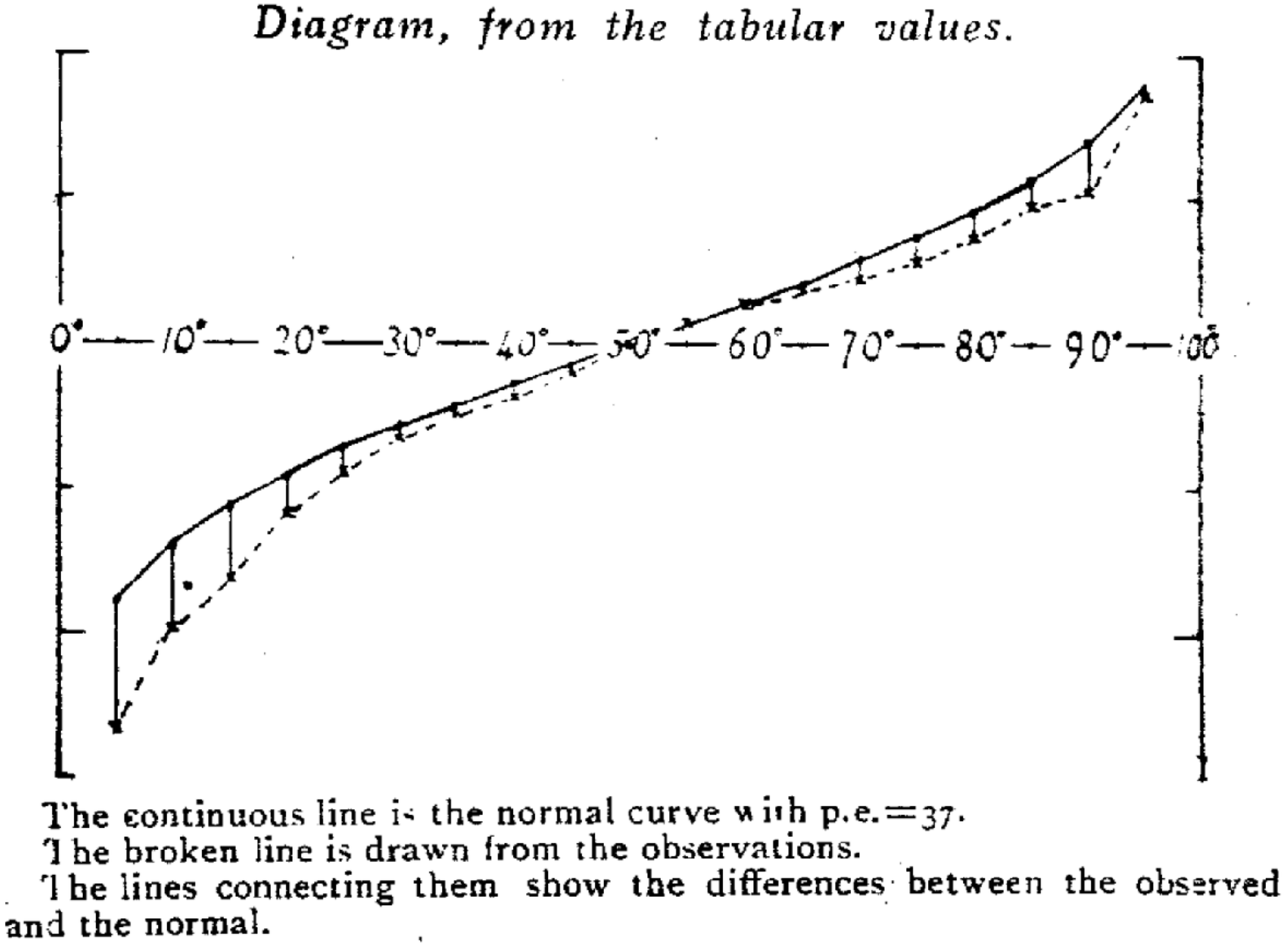


Wisdom of the crowds (ensemble learning)



"The material about to be discussed refers to a small matter…"

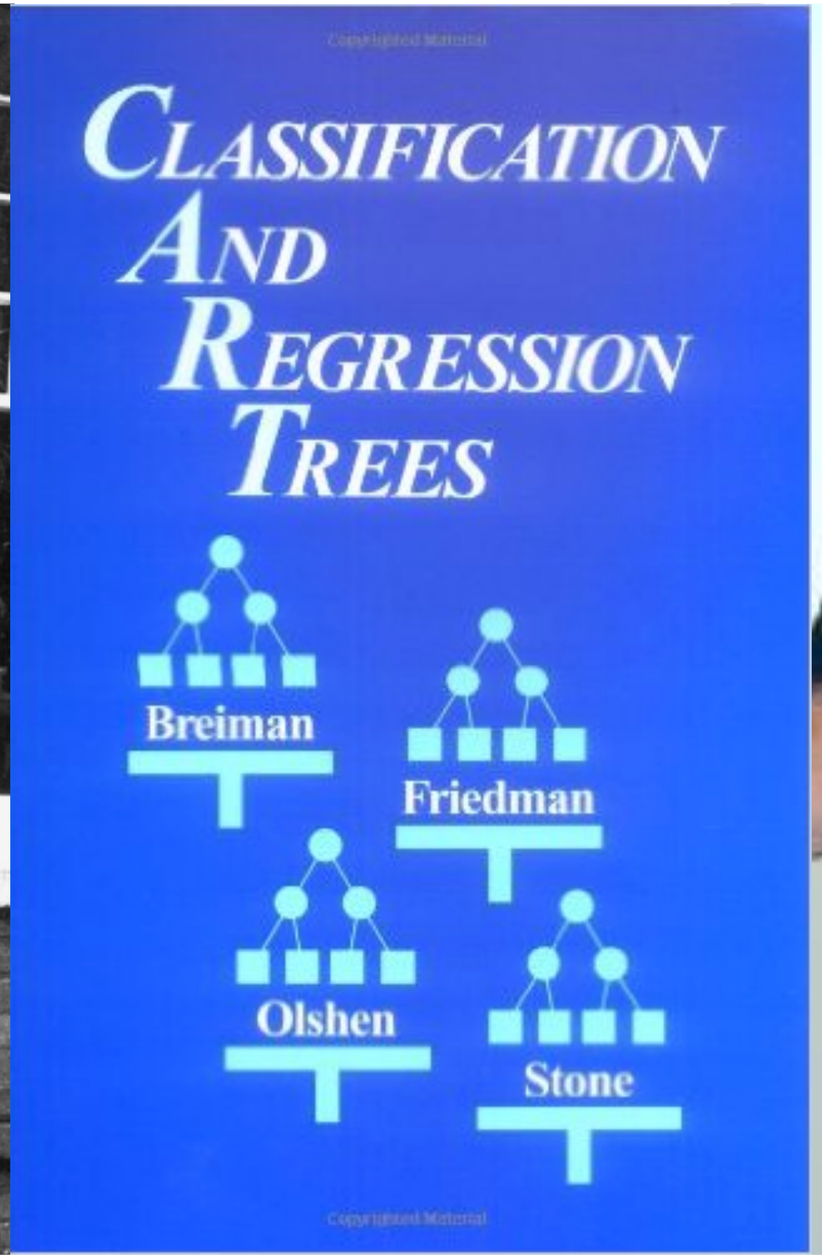
— Sir Francis Galton



http://galton.org/essays/1900-1911/galton-1907-vox-populi.pdf?page=7

Leo Breiman


KARL
PEARSON
1857–1936
Pioneer
Statistician
lived here


CLASSIFICATION
AND
REGRESSION
TREES
Breiman
Friedman
Olshen
Stone

How Random Forest Works

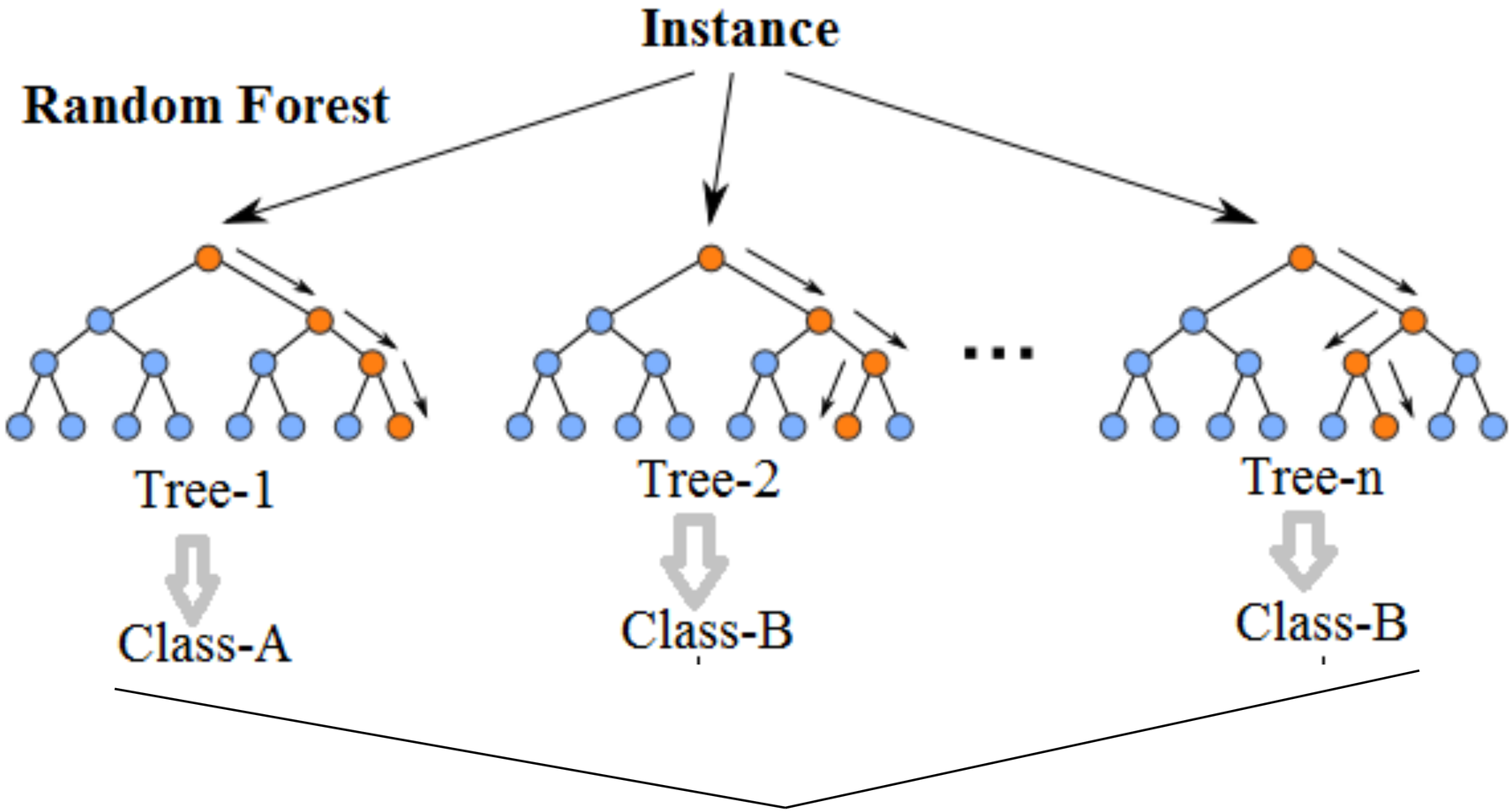
1. Set how many decision trees to grow (ntree)
2. For each tree:
    1. Grab m predictor variables (sqrt(p)) at random
    2. Set break point in continuous or class if categorical
    3. Derive a “vote” for each branch of the tree

To predict a new observation:

1. Allow each tree to “vote” given observation

https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

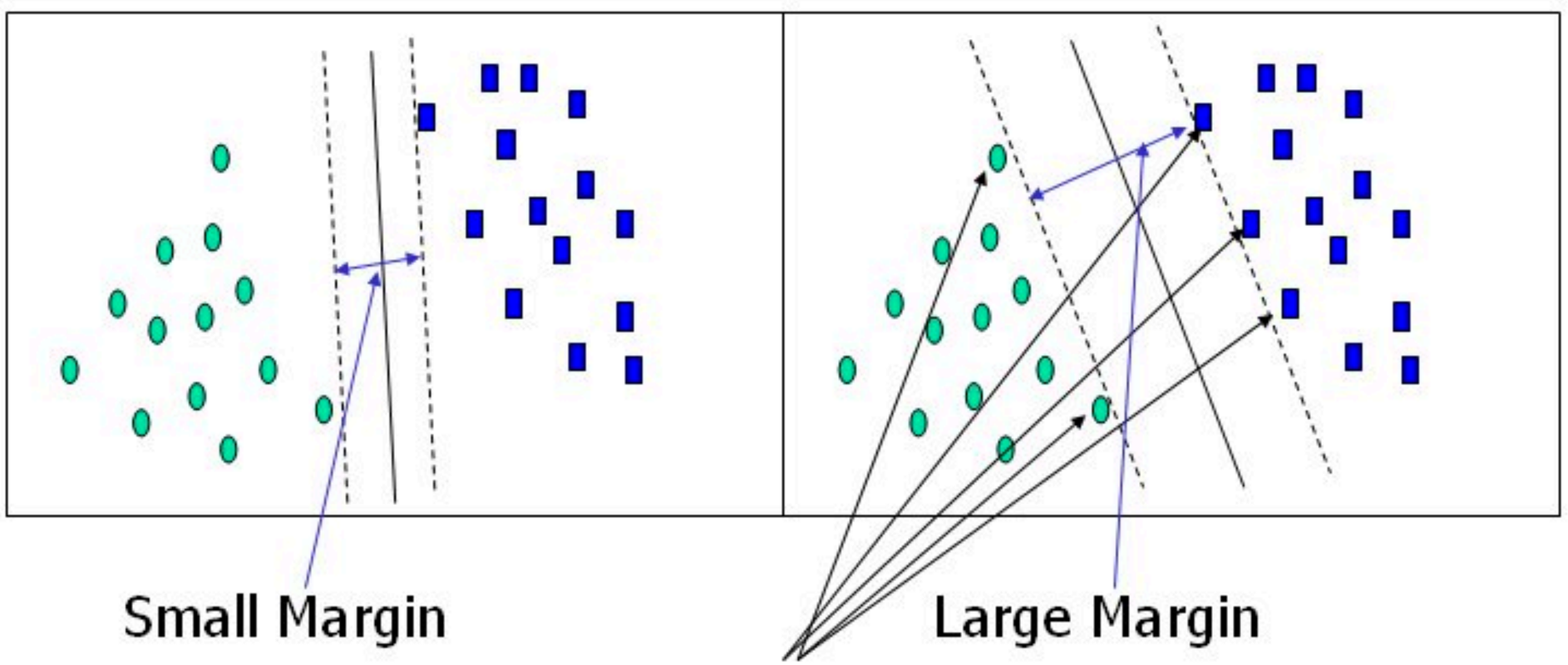
Random Forest



Class A: p% Class B: 1-p%

Majority Win

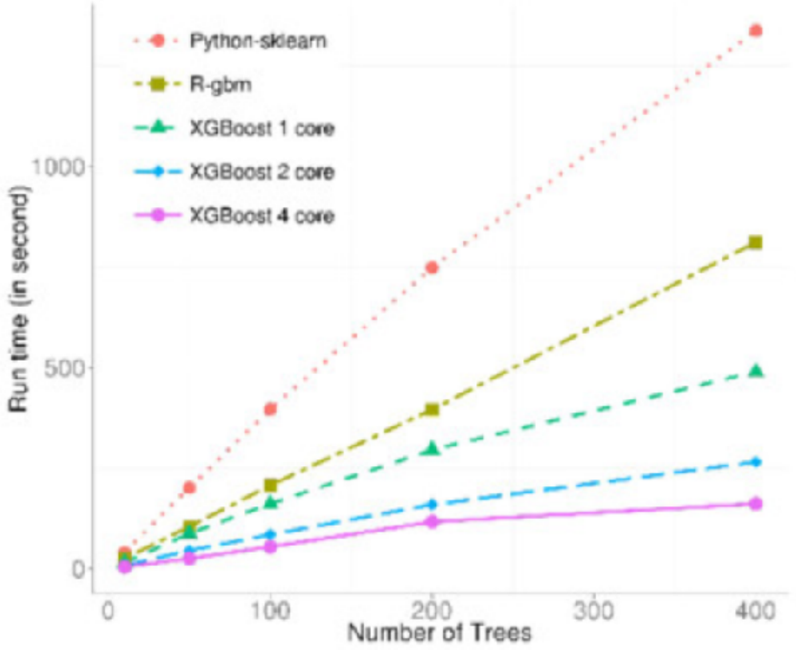
Support Vector Machine



…Can make some fancy margins with kernel densities and hyperplanes in multidimensional data.

What is XGBoost

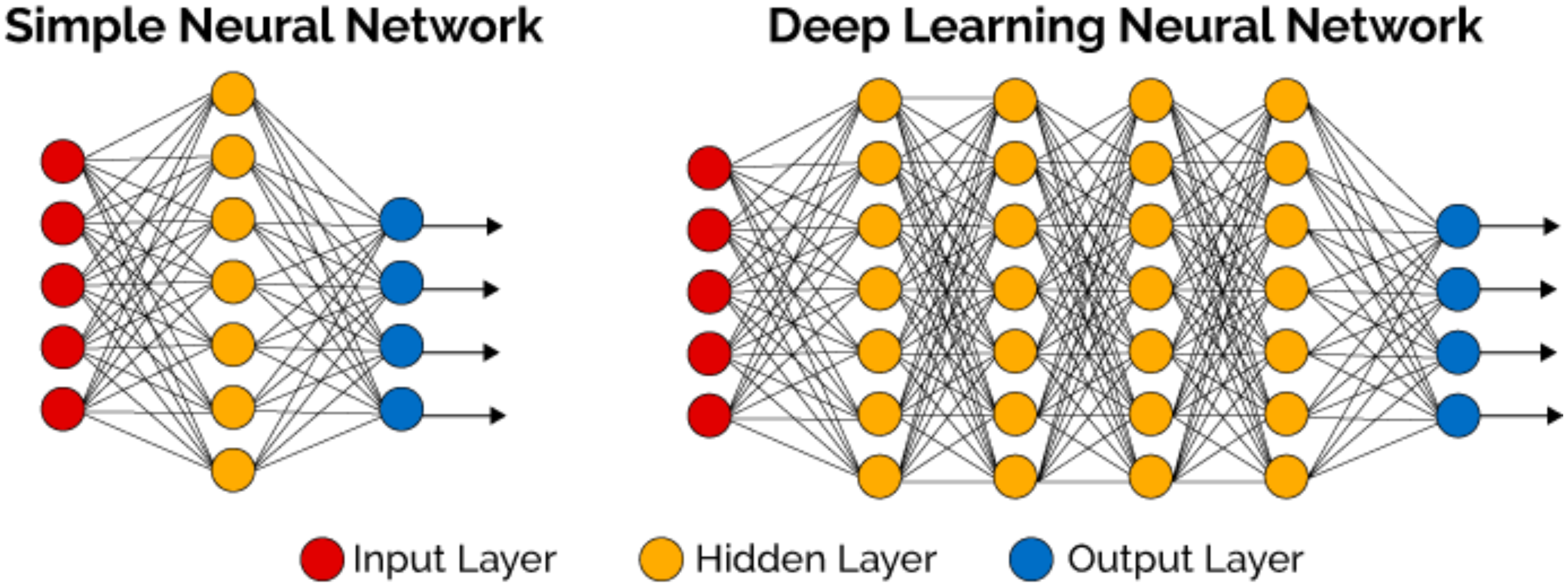
- A Scalable System for Learning Tree Ensembles
  - Model improvement
    - Regularized objective for better model
  - Systems optimizations
    - Out of core computing
    - Parallelization
    - Cache optimization
    - Distributed computing
  - Algorithm improvements
    - Sparse aware algorithm
    - Weighted approximate quantile sketch.



- *In short, faster tool for learning better models*

http://datascience.la/xgboost-workshop-and-meetup-talk-with-tianqi-chen/

Deep Learning/Neural Nets



O'REILLY®

Security

BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

Classifiers in Code

Python Supervised Learning Process

The basic process for any supervised learning is the same for any supervised learning in Python.

1. Create the Classifier or Regressor object.

```
clf = RandomForestClassifier()
```

2. Call the `.fit()` method using the **training feature matrix (X)** and the **training target vector (y)**.

```
clf.fit( X,y )
```

3. Call the `.predict()` method using a feature matrix

```
#Returns vector of predictions
clf.predict( X_test )
```

R Supervised Learning Process (via ‘caret’)

The basic process for any supervised learning is the same for any supervised learning in R.

1. Create the Classifier or Regressor object.

```
clf = RandomForestClassifier()
```

2. Call the `.fit()` method using the **training feature matrix (X)** and the **training target vector (y)**.

```
clf.fit( X,y )
```

3. Call the `.predict()` method using a feature matrix

```
#Returns vector of predictions
clf.predict( X_test )
```

Tuning Parameters in Python

Python provides two ways to automatically find optimal hyper parameter values using either Grid Search or RandomizedSearch.

```
parameter_candidates = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]

# Create a classifier object with the classifier and parameter candidates
clf = GridSearchCV(estimator=svm.SVC(), param_grid=parameter_candidates, n_jobs=-1)

# Train the classifier on data1's feature and target data
clf.fit(data1_features, data1_target)
```

O'REILLY®

Security

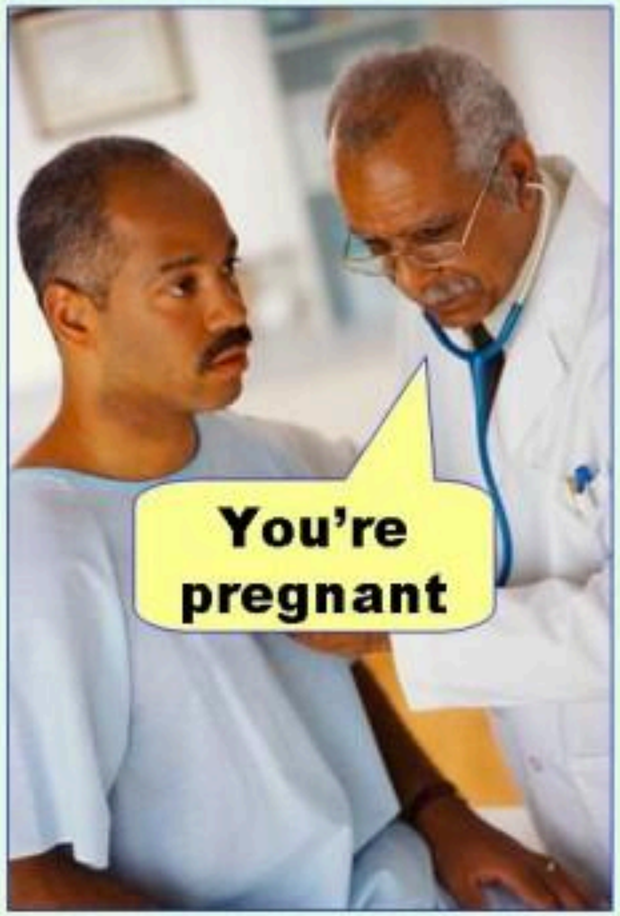
BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

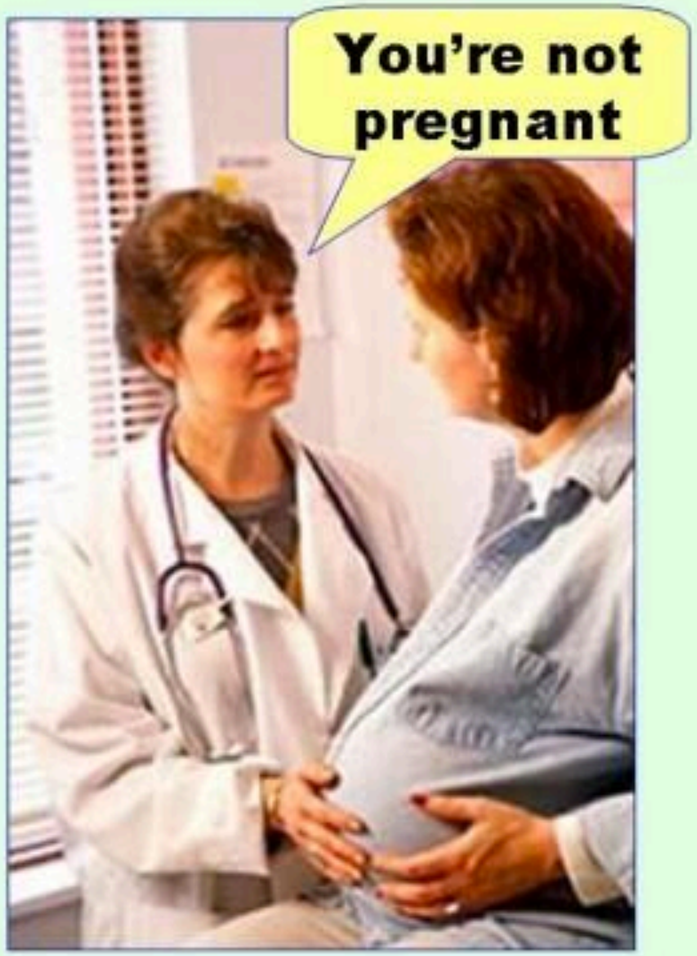
Measuring Performance of a Classifier

Evaluating how well a classifier performs

Type I error
(false positive)



Type II error
(false negative)



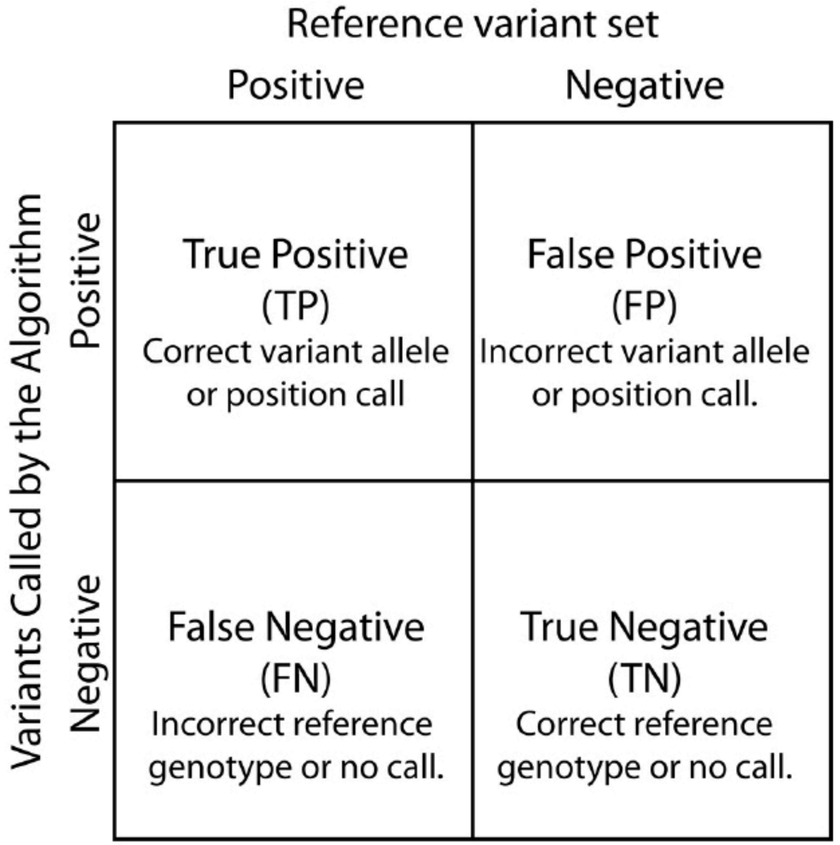
Evaluating how well a classifier performs: Confusion Matrix

Every package/tool should createsome type of “confustion matrix”

```
Confusion Matrix and Statistics

          Reference
Prediction nivdort alexa
   nivdort    6701  1591
   alexa       283  5409
```

[Predicted]

[Reality]



Coding a Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix( y_true, y_predictions )
```

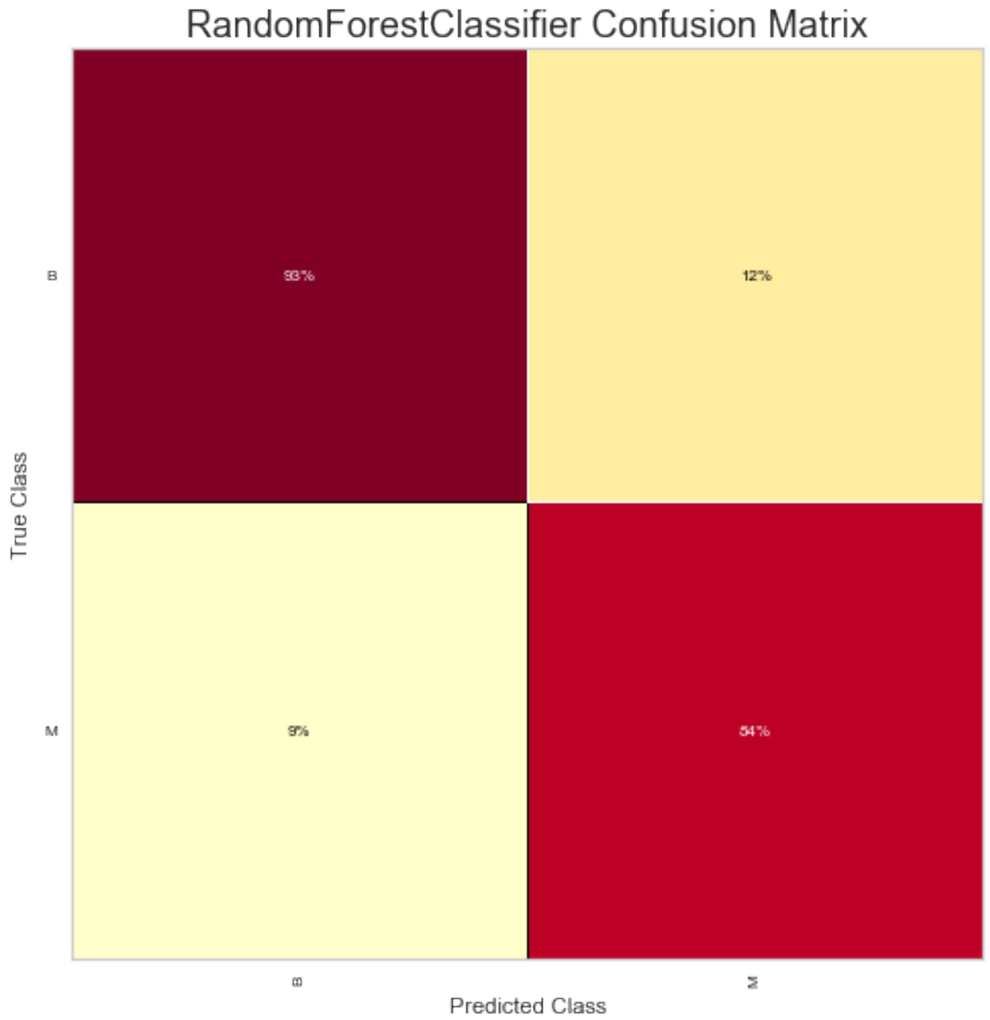
Python

```
library(caret) # should have been loaded already

confusionMatrix
```

R

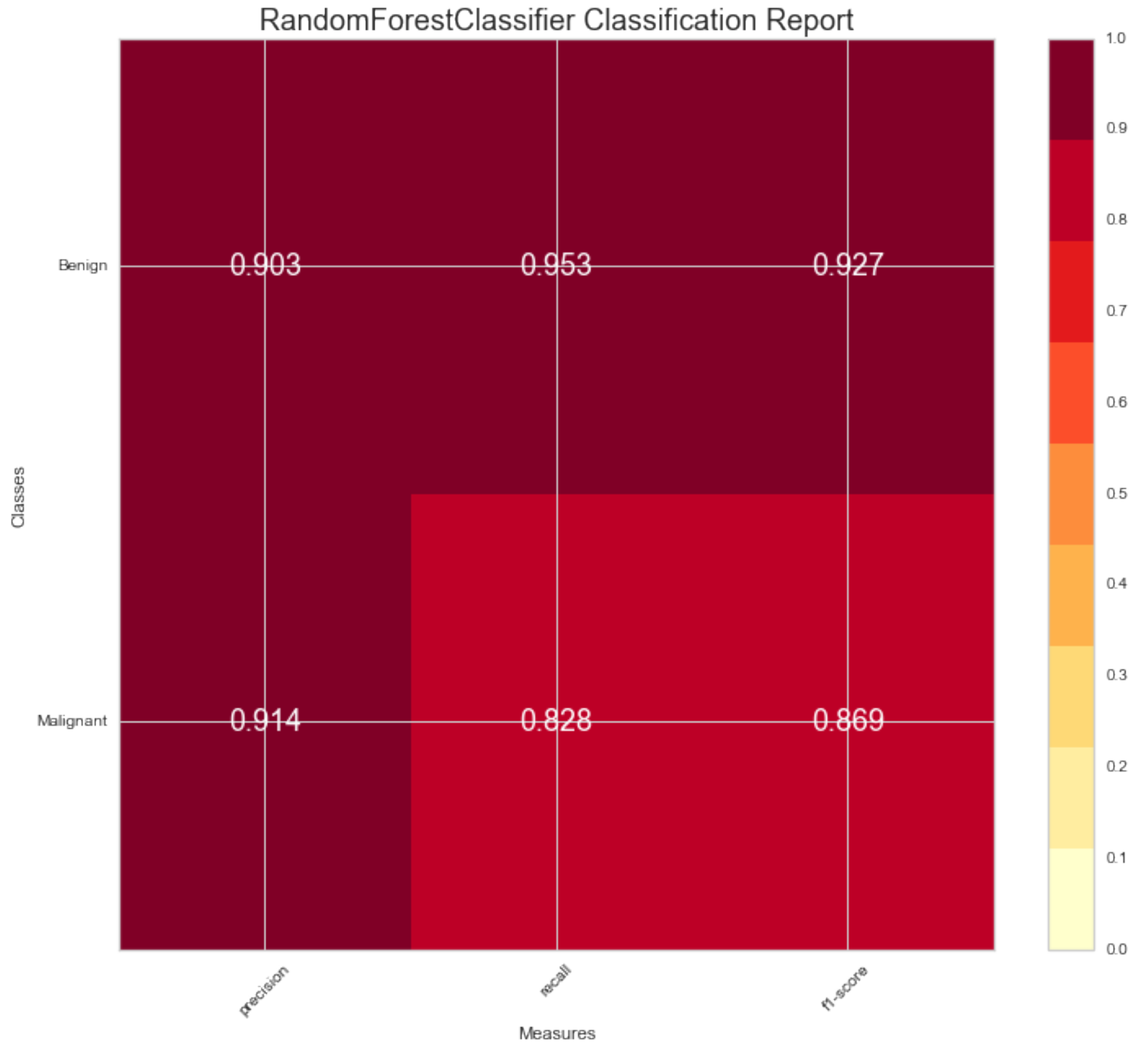
Coding a Confusion Matrix: Pretty Python

```
from yellowbrick.classifier import ConfusionMatrix

#Create the models for both the Random Forest
random_forest_model = RandomForestClassifier()

random_forest_conf_matrix = ConfusionMatrix(
                                random_forest_model )
random_forest_conf_matrix.fit( X_train, y_train )
random_forest_conf_matrix.score( X_test, y_test )
random_forest_conf_matrix.poof()
```

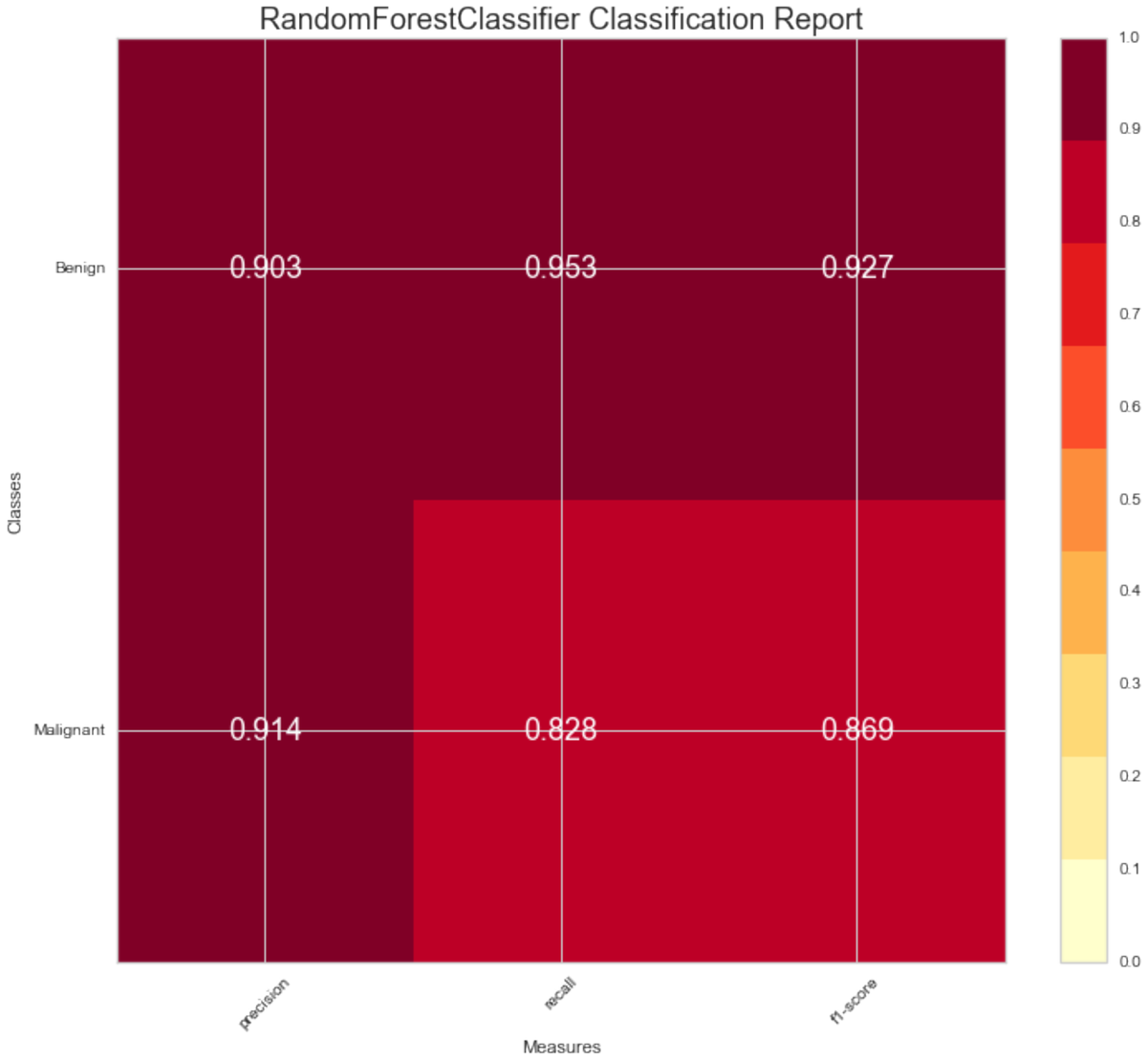


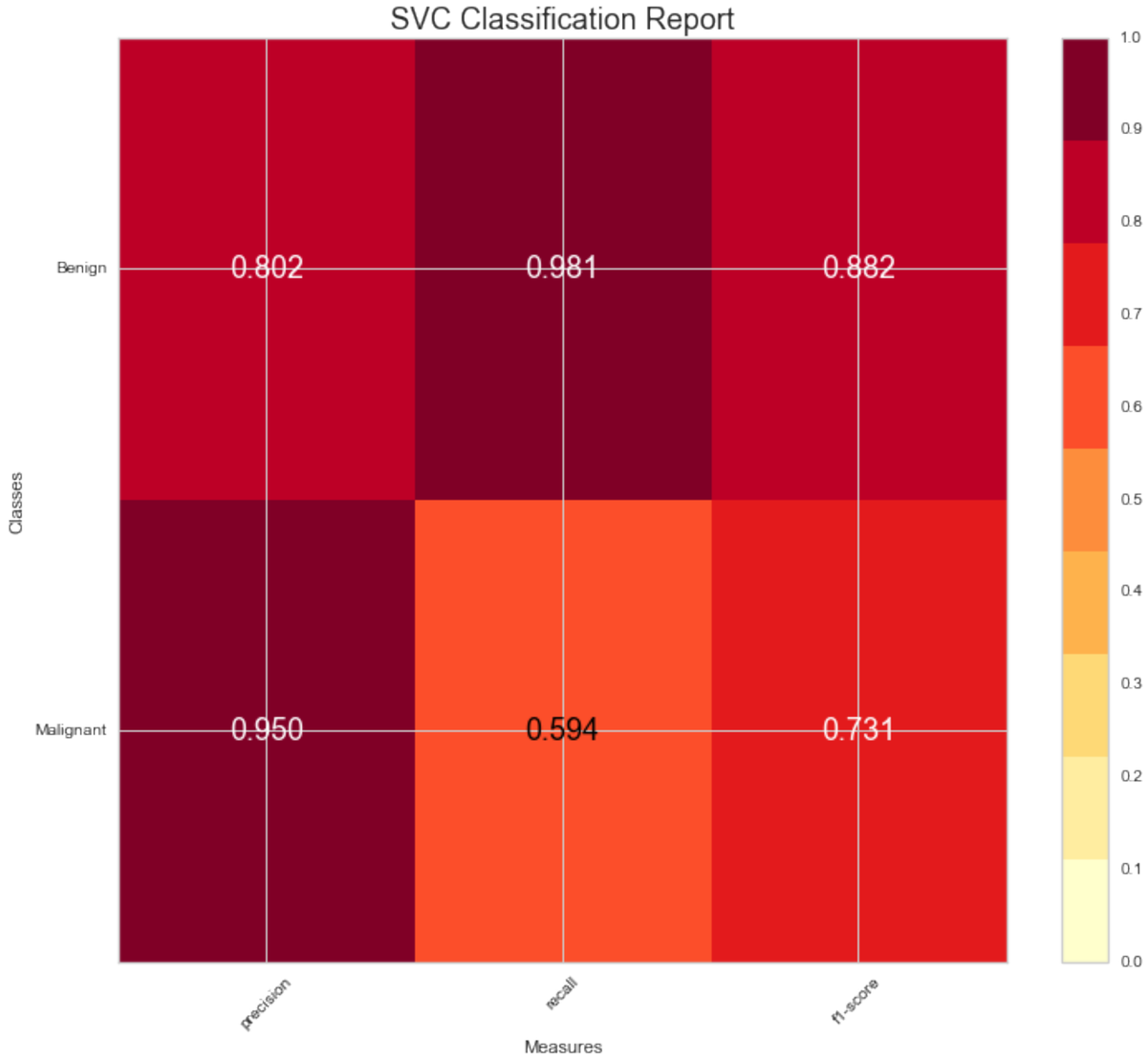
Coding a Classification Report: Pretty Python

```
from yellowbrick.classifier import
ClassificationReport

random_forest_class_report =
ClassificationReport( random_forest_model,
                          classes=['Benign',
        'Malignant'])
random_forest_class_report.fit(X_train, y_train)
random_forest_class_report.score(X_test, y_test)
random_forest_class_report.poof()
```



Coding a Classification Report: Pretty Python





Evaluating how well a classifier performs

```
Confusion Matrix and Statistics

          Reference
Prediction nivdort alexa
   nivdort    6701  1591
   alexa       283  5409

               Accuracy : 0.866
                 95% CI : (0.8602, 0.8716)
    No Information Rate : 0.5006
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.732
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9595
            Specificity : 0.7727
         Pos Pred Value : 0.8081
         Neg Pred Value : 0.9503
             Prevalence : 0.4994
         Detection Rate : 0.4792
   Detection Prevalence : 0.5930
      Balanced Accuracy : 0.8661

       'Positive' Class : nivdort
```

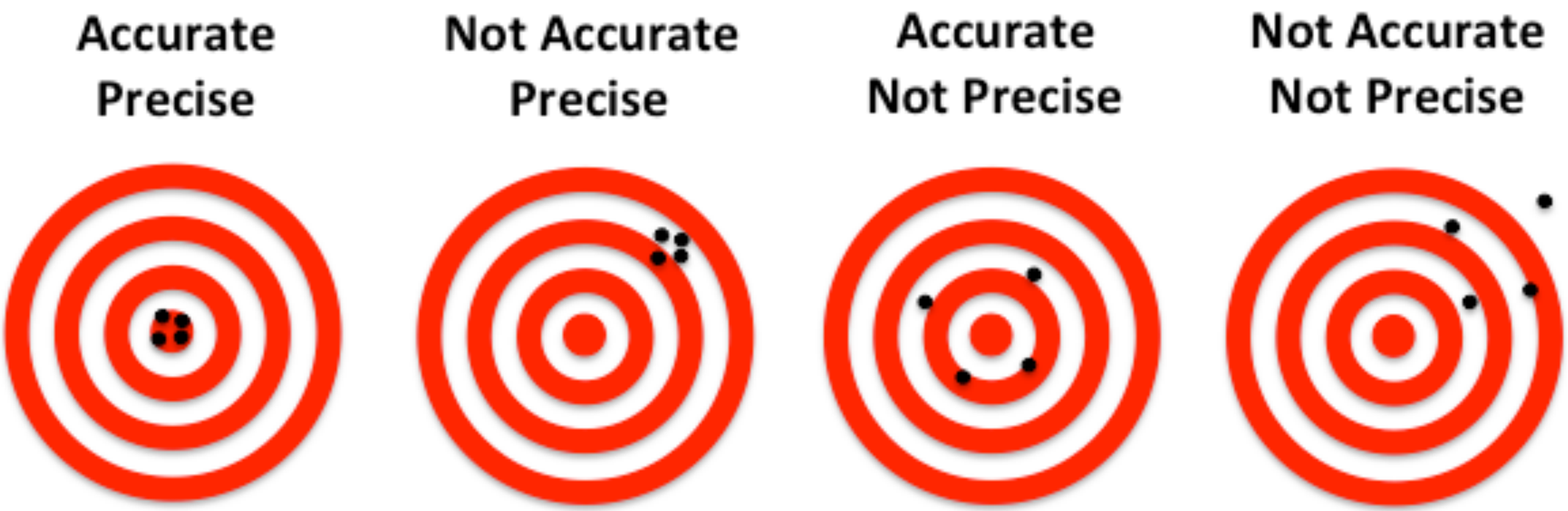
Overall Accuracy of the Classifier

95% confidence intervals around accuracy

% of prevalent class (if we say “all dga”)

p-value if Accuracy is > No-Info Rate

Evaluating how well a classifier performs



Discussion: Accuracy

You are given a large set of netflow events with some malicious events identified.

You build a classification model and have achieved an accuracy of 90%…

Should you be happy with your model performance?

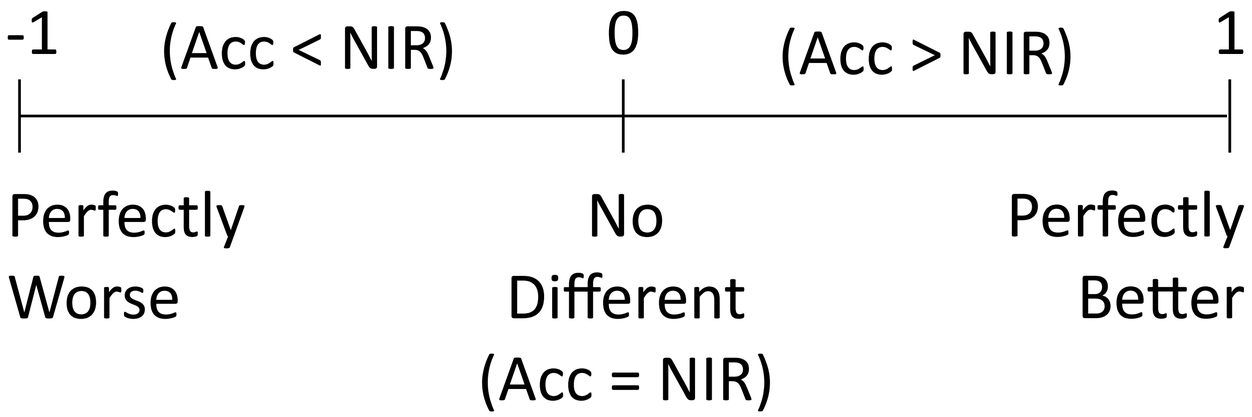
Classifying all events as non-malicious == 99.7% accuracy.

Unbalanced data does not lend itself to accuracy, look at precision/recall instead

Evaluating how well a classifier performs

```
Confusion Matrix and Statistics

          Reference
Prediction nivdort alexa
   nivdort    6701  1591
   alexa       283  5409

               Accuracy : 0.866
                 95% CI : (0.8602, 0.8716)
    No Information Rate : 0.5006
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.732
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9595
            Specificity : 0.7727
         Pos Pred Value : 0.8081
         Neg Pred Value : 0.9503
             Prevalence : 0.4994
         Detection Rate : 0.4792
   Detection Prevalence : 0.5930
      Balanced Accuracy : 0.8661

       'Positive' Class : nivdort
```

Kappa:
compares an Observed Accuracy with an Expected Accuracy (random chance).



p-value with H_0: kappa = NIR

Evaluating how well a classifier performs

```
Confusion Matrix and Statistics

          Reference
Prediction nivdort alexa
   nivdort    6701  1591
   alexa       283  5409

               Accuracy : 0.866
                 95% CI : (0.8602, 0.8716)
    No Information Rate : 0.5006
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.732
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9595
            Specificity : 0.7727
         Pos Pred Value : 0.8081
         Neg Pred Value : 0.9503
             Prevalence : 0.4994
         Detection Rate : 0.4792
   Detection Prevalence : 0.5930
      Balanced Accuracy : 0.8661

       'Positive' Class : nivdort
```

	Reference	
Predicted	Event	No Event
Event	A	B
No Event	C	D

The formulas used here are:

$$Sensitivity = \frac{A}{A+C}$$

$$Specificity = \frac{D}{B+D}$$

$$Prevalence = \frac{A+C}{A+B+C+D}$$

Sensitivity: ability to identify “positive” class

Specificity: ability to identify non-positive class

Evaluating how well a classifier performs

```
Confusion Matrix and Statistics

          Reference
Prediction nivdort alexa
   nivdort    6701  1591
   alexa       283  5409

               Accuracy : 0.866
                 95% CI : (0.8602, 0.8716)
    No Information Rate : 0.5006
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.732
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9595
            Specificity : 0.7727
         Pos Pred Value : 0.8081
         Neg Pred Value : 0.9503
             Prevalence : 0.4994
         Detection Rate : 0.4792
   Detection Prevalence : 0.5930
      Balanced Accuracy : 0.8661

       'Positive' Class : nivdort
```

	Reference	
Predicted	Event	No Event
Event	A	B
No Event	C	D

$$PPV = \frac{sensitivity \times prevalence}{((sensitivity \times prevalence) + ((1 - specificity) \times (1 - prevalence))}$$

$$NPV = \frac{specificity \times (1 - prevalence)}{((1 - sensitivity) \times prevalence) + ((specificity) \times (1 - prevalence))}$$

$$Detection\ Rate = \frac{A}{A + B + C + D}$$

$$Detection\ Prevalence = \frac{A + B}{A + B + C + D}$$

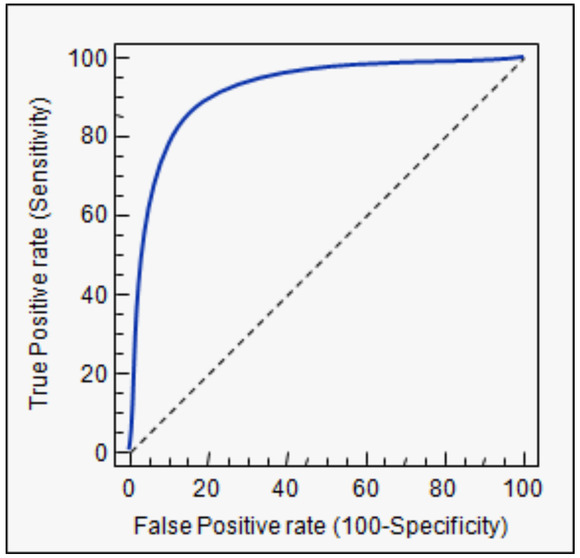
$$Balanced\ Accuracy = (sensitivity + specificity)/2$$

$$Precision = \frac{A}{A + B}$$

$$Recall = \frac{A}{A + C}$$

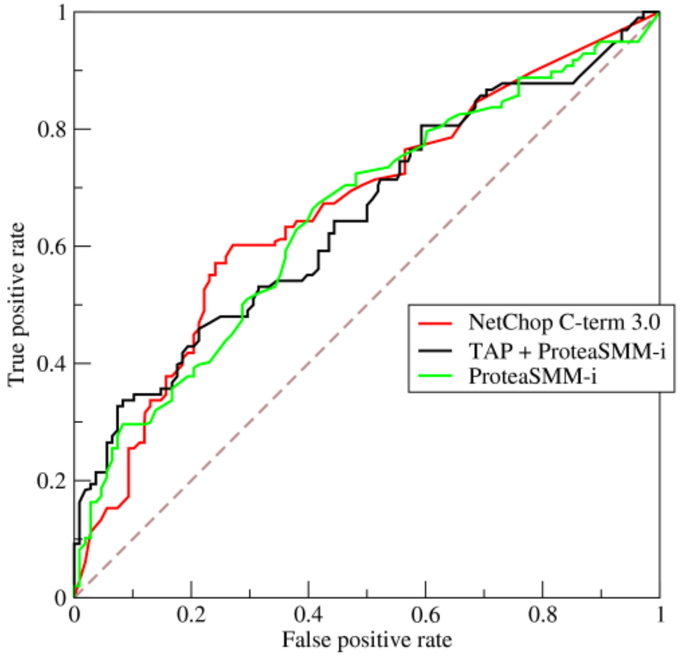
$$F1 = \frac{(1 + \beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}$$

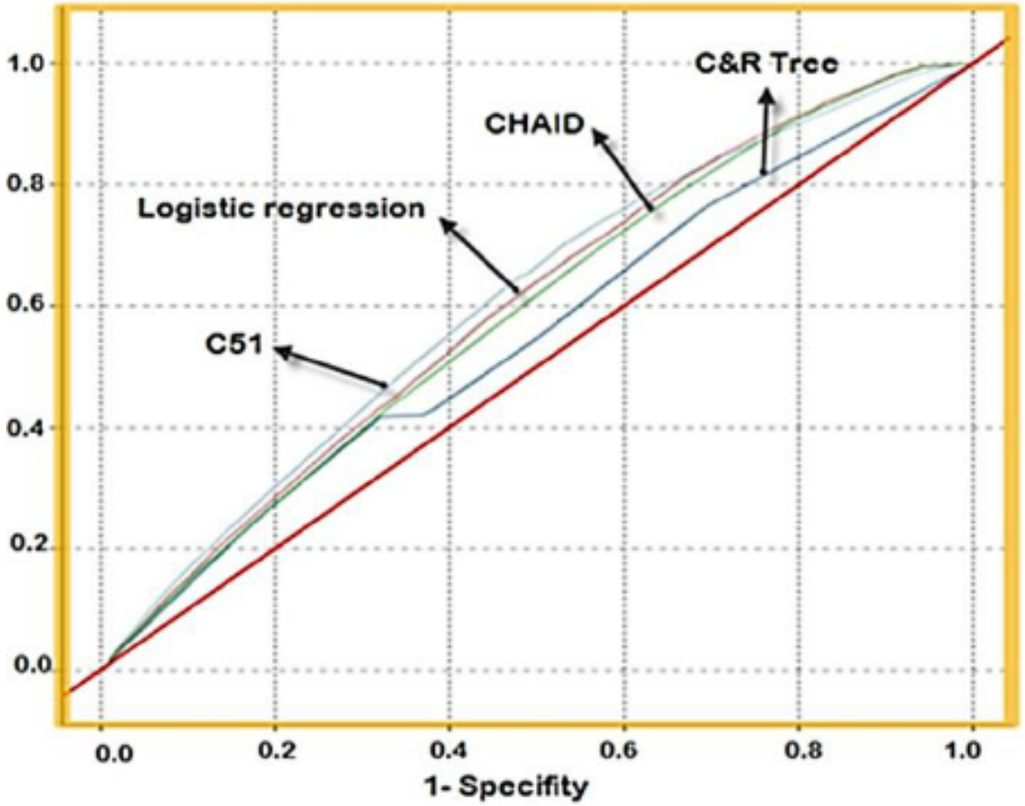
ROC Curve and AUC

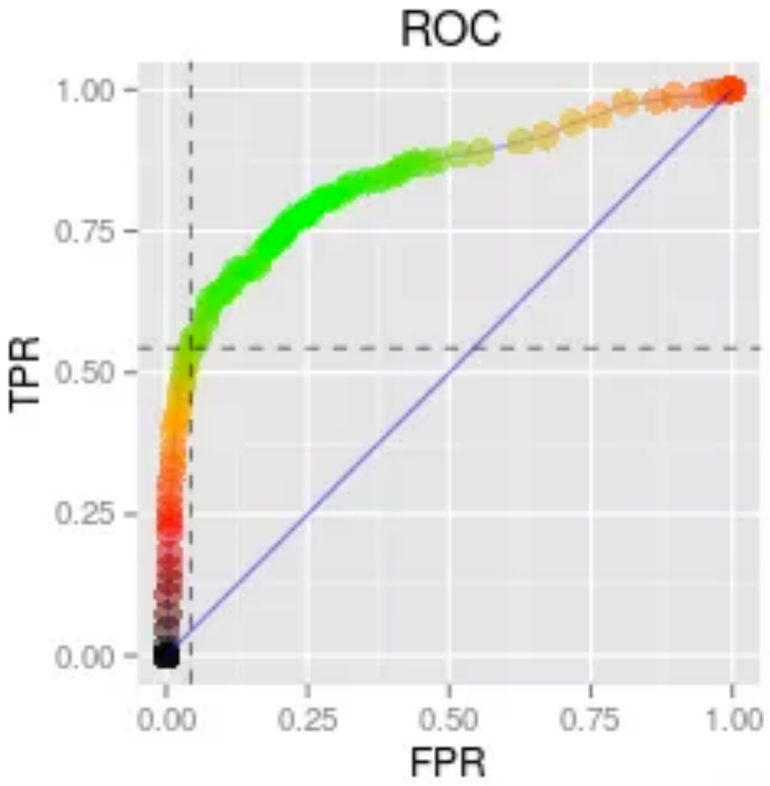


Receiver Operator Curve, (ROC)

- Plots TPR (Sensitivity) against FPR (1-Specificity)
- Area Under the Curve (AUC) is a measurement of volume under curve.

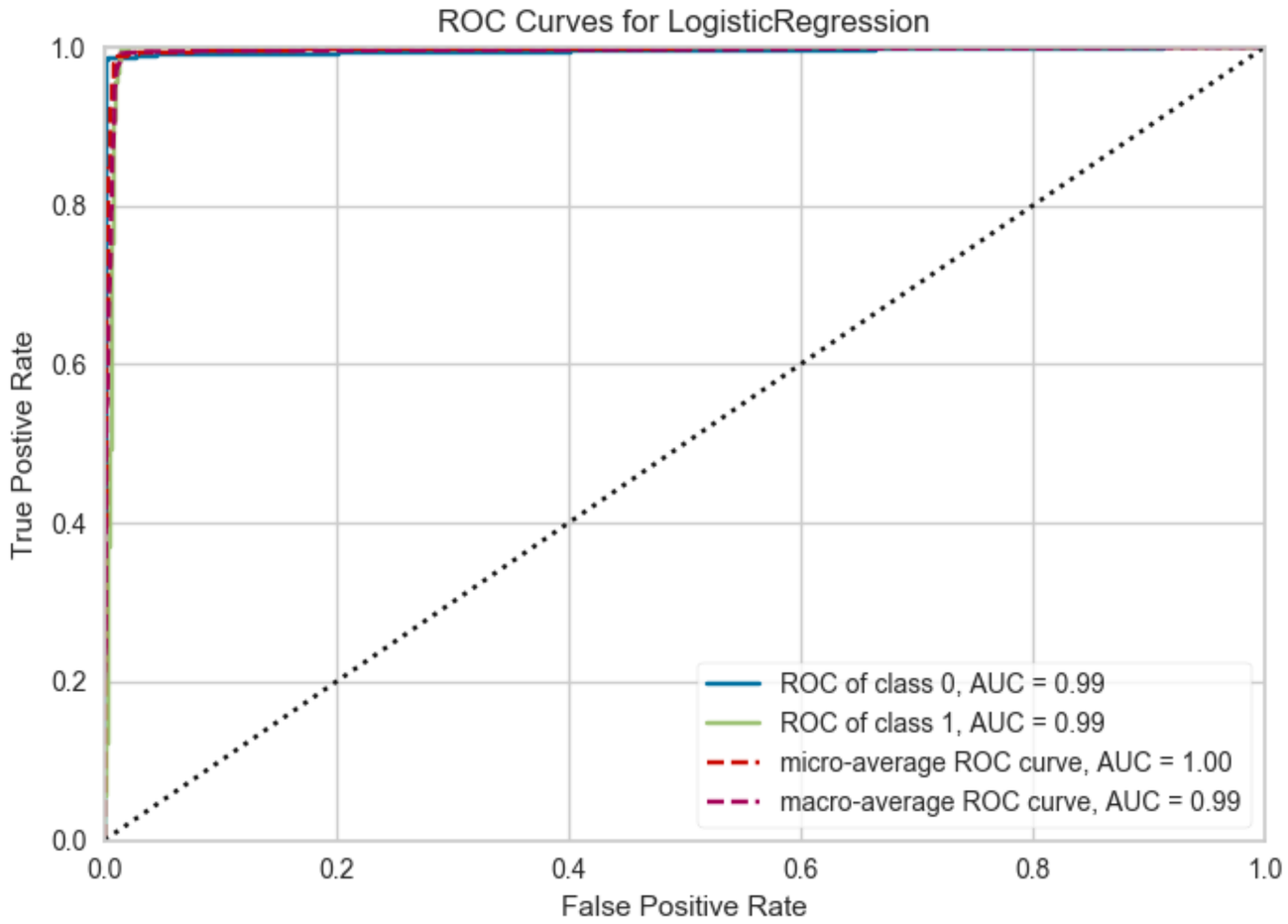






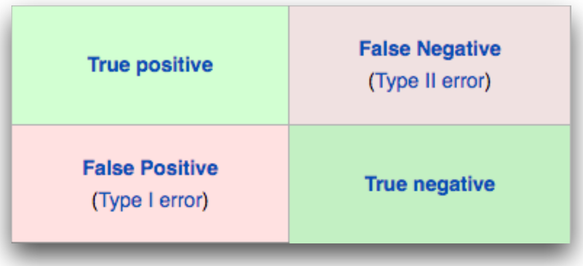
ROC Curve in Python

```
from yellow brick.classifier import ROCAUC

logistic = LogisticRegression()
visualizer = ROCAUC(logistic)

visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
g = visualizer.poof()
```



http://www.scikit-yb.org/en/latest/api/classifier/rocauc.html

Precision



True target

0 [[7 2]

1 [3 4]]

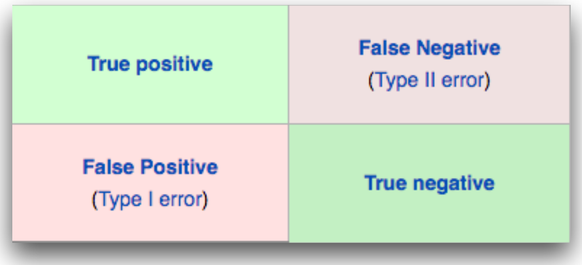
0 1

Predicted target

Precision = TP/ (TP + FP)
Quiz: Calculate precision by hand for both classes!

“Precision” is how precise your estimates are. It’s the percentage of all your positive estimates you made that are correct.

Recall



True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Recall = TP/ (TP + FN)
Quiz: Calculate recall by hand for both classes!

"Recall" is the percentage of the positive class correctly classified .

Precision and Recall

- You have a data set of 5000 files, 2000 of which are malware.
- You write a classifier and classify 1000 of the files as malware.
- Of those 1000, 820 were actually malware.

Precision: 820 of the 1000 you said were malware were indeed malware.

820/1000 = 82%

Recall: out of the 2000 malware samples, you correctly labelled 820.

820/2000 = 41%

F-Measure

What if the use case has a different value for Precision and Recall?

- SPAM: having a valid email marked as spam is much worse than allowing a spam through — Precision is very important. You want a very high proportion of blocked spam to actually be spam.

- SOC Alerts: Missing a critical alert could be catastrophic — Recall is very important - out of all the true critical alerts, you want to correctly identify as much as you can.

- SOC Alerts 2: False positives are crippling response times. — May need to more value on precision. Out of all those we label as critical, we want a high proportion to in fact, be critical

F-Measure

What if the use case has a different value for Precision and Recall?

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\beta^2 \cdot \mathrm{precision} + \mathrm{recall}}$$

Where Beta is the value of recall to precision:

Beta = 1, evan value

Beta = 2, recall is twice as important

Beta = 0.5, precision is twice as important.

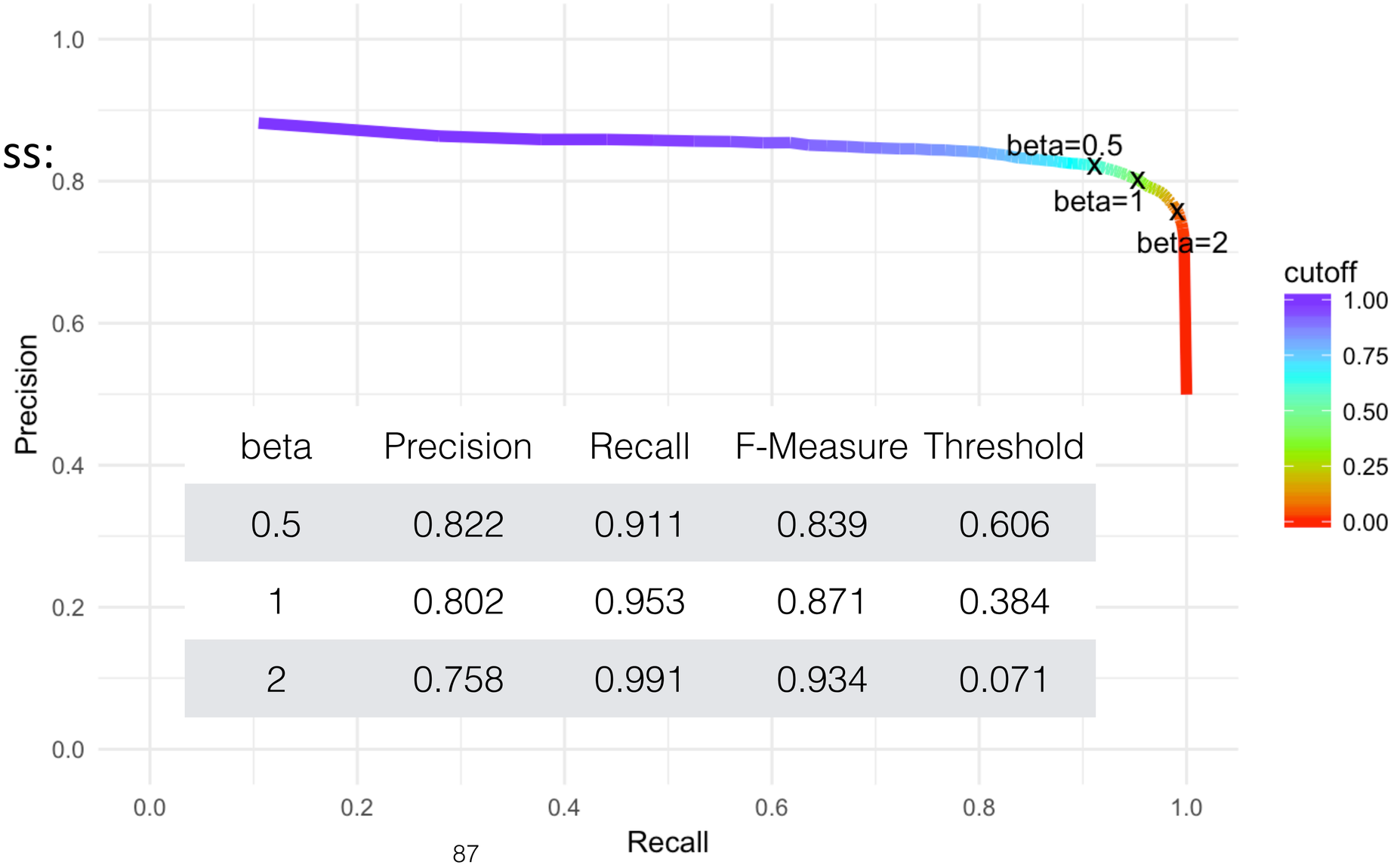
Note: There are packages for this!

F-Measure

Comparing “nivdort” to “alexa”

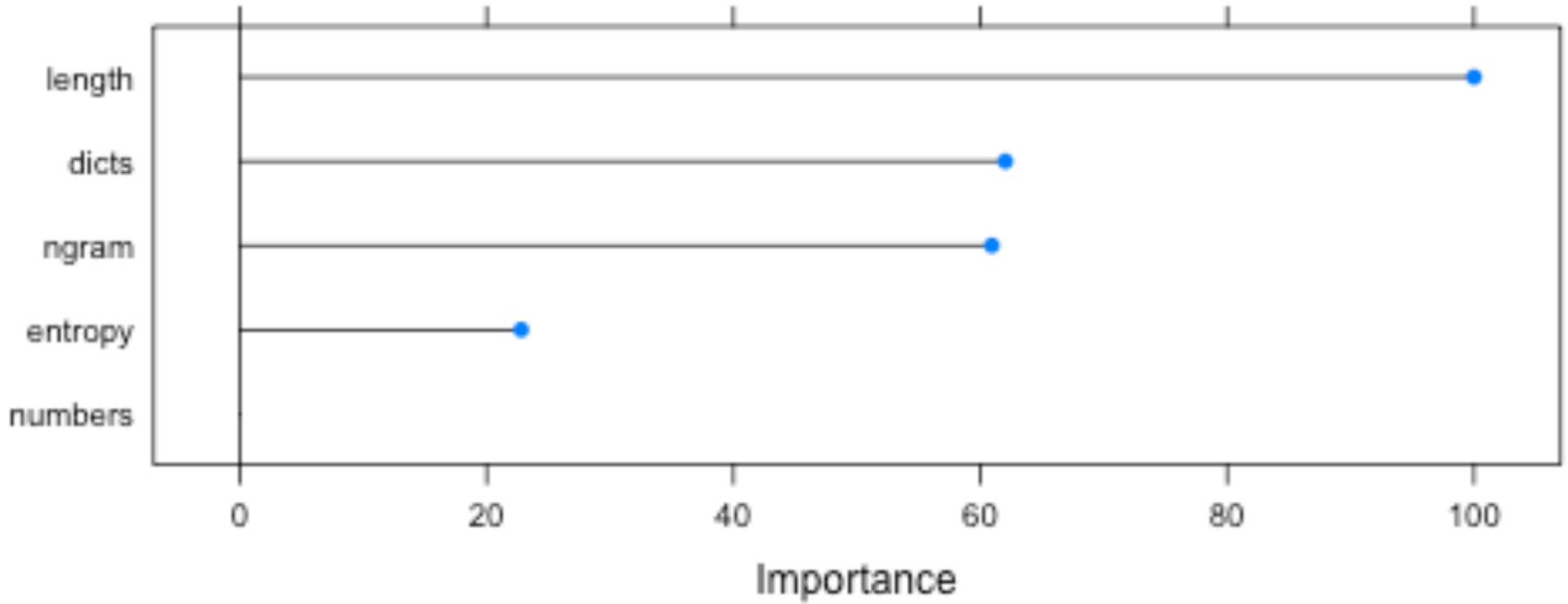
Probabilities of each class:

```
   nivdort alexa
1    0.992 0.008
2    0.928 0.072
3    0.822 0.178
4    0.838 0.162
5    1.000 0.000
6    0.850 0.150
  . . .
```



beta	Precision	Recall	F-Measure	Threshold
0.5	0.822	0.911	0.839	0.606
1	0.802	0.953	0.871	0.384
2	0.758	0.991	0.934	0.071

Variable importance…




```
plot(varImp(rfFit))
```

Note: SVM does not have a “varImp” function

{LAB} time

Look over the “example” worksheets
Evaluate a Classifer on “nivdort” and “alexa”