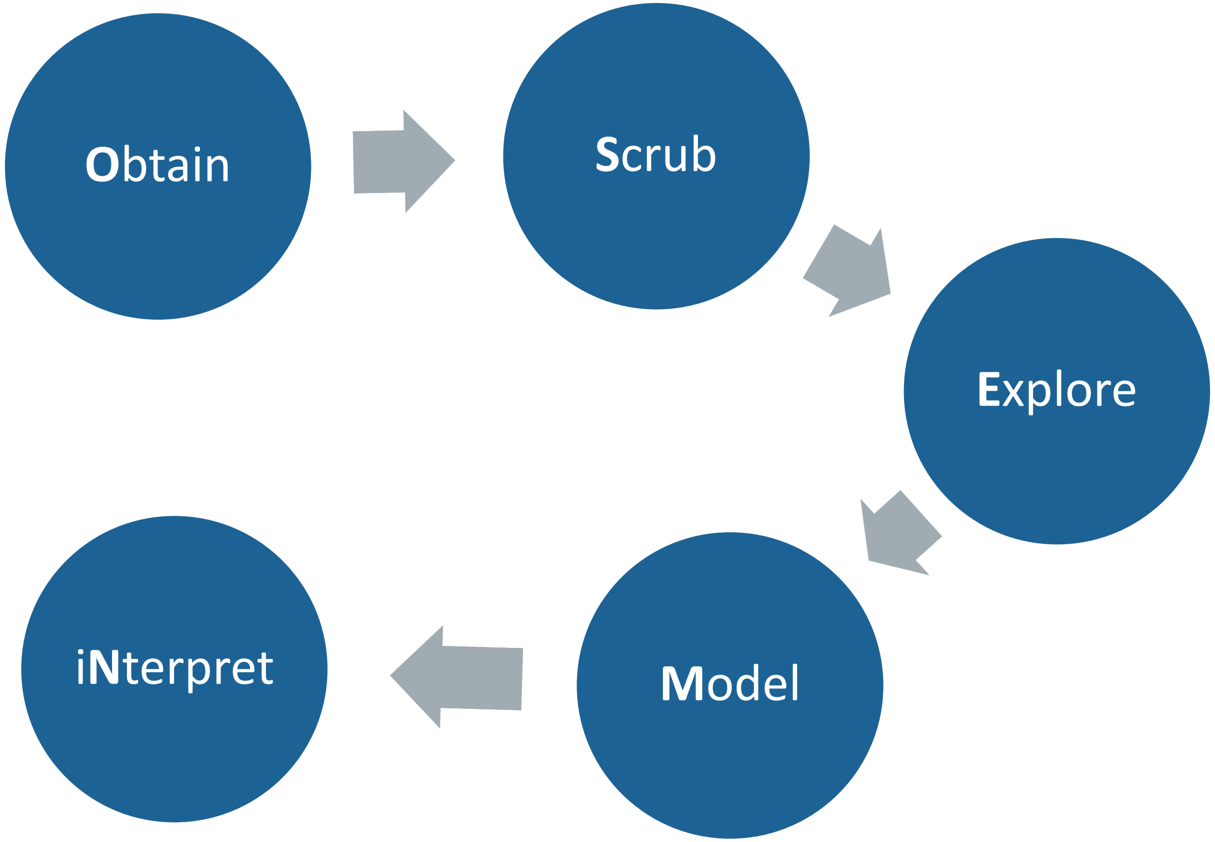
Machine Learning Overview
Part I

School of Information Studies
Syracuse University

The Machine Learning Process
OSEMN (Rhymes With Possum)

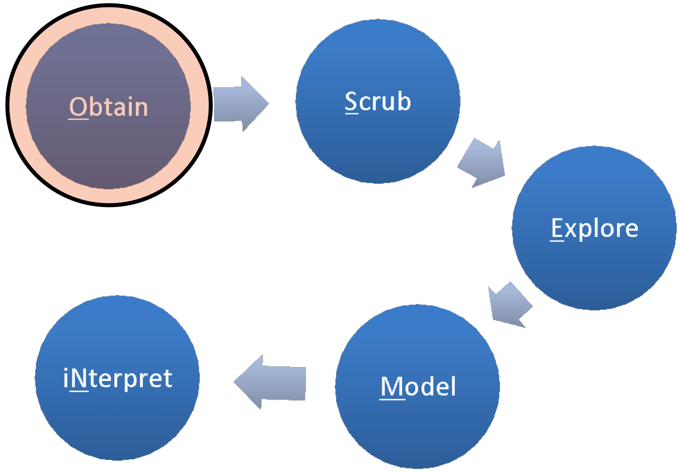


Hilary Mason and Chris Wiggins, 2010 - http://www.dataists.com/2010/09/a-taxonomy-of-data-science

OSEMN Phase: Obtain

Collect data

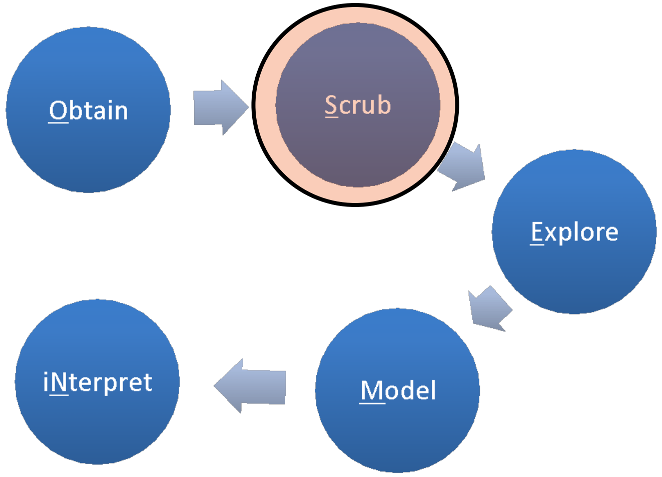
- Deal with file formats and how to read data
- Query databases or data repositories
- Extract data from other sources
- Generate data (e.g., surveys, sensors)



OSEMN Phase: Scrub

Get data into a useable format structure

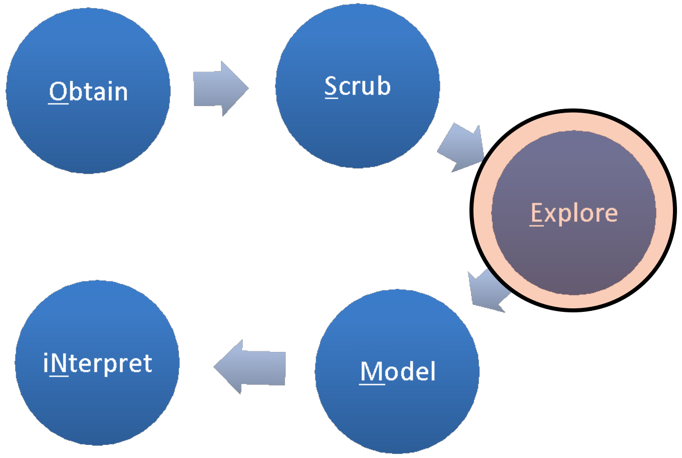
- Filter/subset
- Extract attributes
- Replace/handle missing, illegal, and anomalous values
- Transform/bin/code data attributes



OSEMN Phase: Explore

Explore patterns and trends

- Start to "understand" the data, detect outliers
- Visualize attributes (e.g., scatter plots, histograms)
- Calculate/visualize descriptive statistics (e.g., distributions)
- Feature selection (what attributes are most interesting)



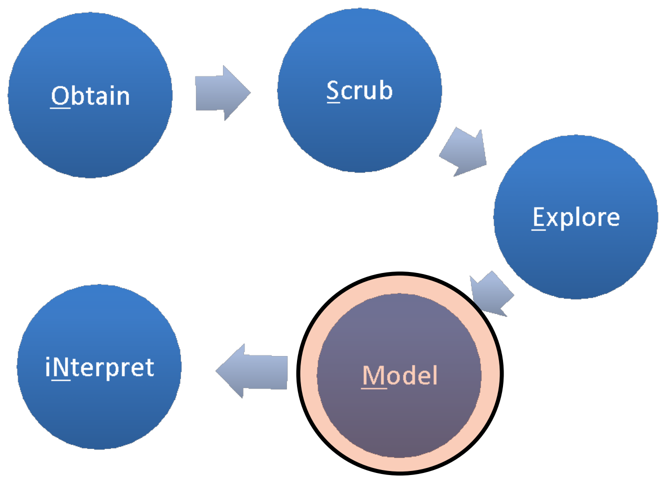
OSEMN Phase: Model

Build predictive models (machine learning)

- Type of modeling:
  - Supervised (classification, regression)
  - Unsupervised (e.g., clustering)

  → *Will discuss shortly*

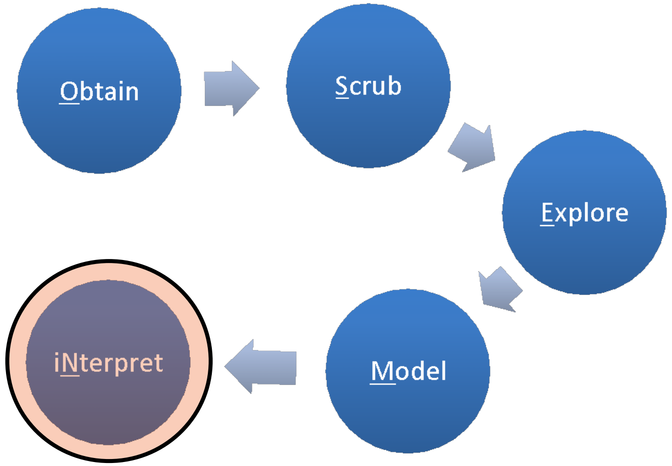
- Model tuning and comparing candidate models



OSEMN Phase: Interpret

Understand/explain the results

- Draw conclusions from and data models
- Evaluate the meaning of results
- Communicate the results
- Ensure actionable insight



Question

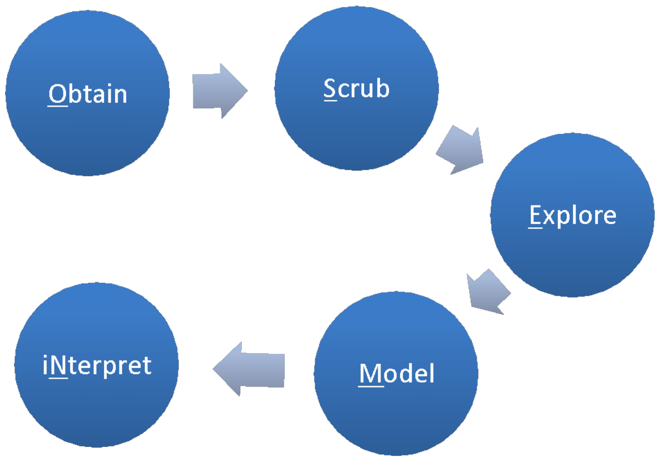
How is the OSEMN process different from or the same as the overall data science process?

Machine Learning Overview
Part II

School of Information Studies
Syracuse University

Question

How is the OSEMN process different from or the same as the overall data science process?



Supervised Machine Learning in a Nutshell

"Supervised" refers to the idea that there is a **criterion** used during an algorithm training phase.

A supervised algorithm uses a set of input variables to optimize the prediction of an outcome variable.

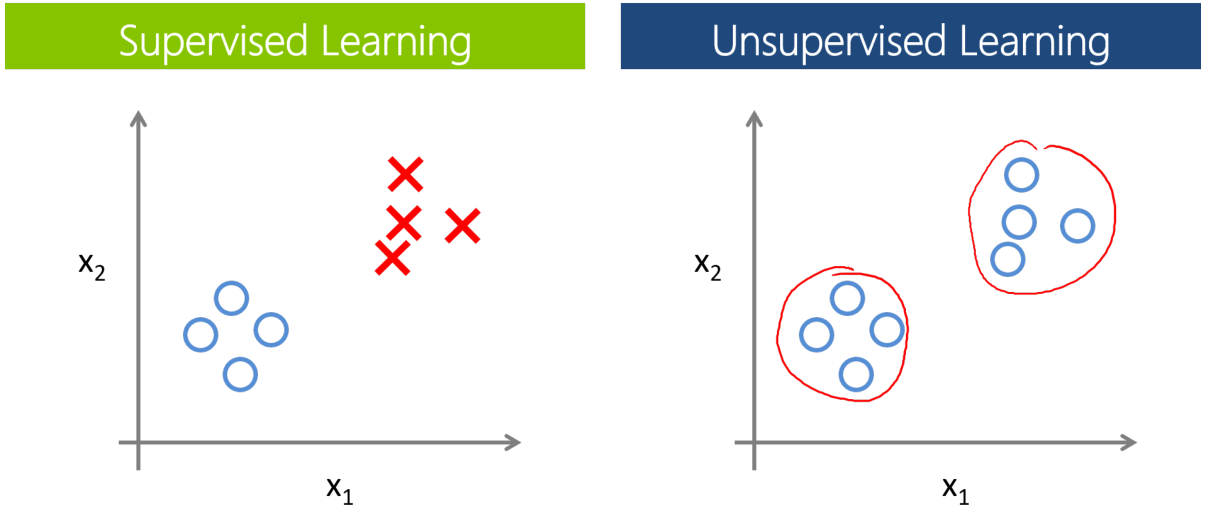


Image credit: Olivia Klose

Examples of Machine Learning Techniques

Unsupervised learning includes a variety of machine learning techniques that do not use a criterion or dependent variable, but rather look for patterns solely among "independent" variables.

- Unsupervised learning
  - Components analysis
  - K-means clustering
  - Self-organizing maps
  - Association rules mining
- Supervised learning
  - Neural networks
  - Support vector machines
  - Boosted regression trees
  - Classification & regression trees
  - General additive models

Supervised learning is parallel in concept to the predictive statistical techniques used by many social science researchers, such as linear regression, but without the restriction of only exploring linear relationships.

Another form of learning is known as "**reinforcement learning**." Evolution of these models depends on success/failure cues from real or simulated environments.

Question

What is the meaning of the word “supervised” in the context of data mining?

Machine Learning Overview Part III

School of Information Studies
Syracuse University

Two Types of Prediction
(For Supervised Learning)

1. **Classification problems**
   - **Predicting membership in two or more categories** based on a set of predictors (model features)
     - **Examples of criteria to predict**
       - Medical diagnosis
       - Employment outcomes (e.g., hiring)
       - Financial outcome (e.g., customer default on a loan)
2. **Regression problems**
   - **Predicting a continuous numeric outcome** based on a set of predictors. **Examples of criteria to predict:** sales volume, employee engagement, customer satisfaction

Traditional approaches

Classification problems

- Logistic regression
- Discriminant analysis

Regression problems

- Ordinary least squares regression (lm() models)
- Generalized linear models
- Lasso regression

Supervised Learning Example

Train a machine learning algorithm to predict the weather

- Collect weather data over a period of time
  - Sunny, cloudy
  - Temperature
  - Barometer
  - Wind speed and direction
- Train a machine learning algorithm with these collected variables
- Collect more weather data and predict the weather via our trained algorithm
  - Classification would be predicting good weather or bad weather
  - Regression would be predicting the temperature
- Then validate the prediction

The Modeling Process

Use a substantial number of training cases

- The machine learning algorithm can use that data to build a model

Use the results of this process (i.e., the model) on test data set to determine how well algorithm performed

- Validate the model on new data

The result is a model that can be used for prediction

- Predict data that was not used during training
- Predict future instances of data

*Note: The model is **not always useful** for explaining results to managers.*

In some cases, algorithms produce results that are not easy to interpret or visualize; for some algorithms there is no output that is like a regression coefficient.

Using Classification and Regression Trees

School of Information Studies
Syracuse University

An Example: CART
Classification and Regression Trees

A family of data mining techniques for developing predictions on **either** continuous outcome variables or class outcome variables

Can be used either for classification with unordered categories, or pseudo-continuous ordered outcomes

Uses iterative model building techniques, typically:

- Develops "splits" in the data where the level of a predictor is used to divide the data set into two (or more) partitions according to the status of the outcome variable
- The resulting models can be represented as binary trees

Example of rpart: For Titanic, Part I

```
library(rpart)

load("titanic.raw.rdata")
titanic <- titanic.raw

#Build the model
cartTree <- rpart(Survived ~ ., data = titanic)
```

Example of rpart: For Titanic, Part II

cartTree

n= 2201

node), split, n, loss, yval, (yprob)

*Denotes terminal node

```
1) root 2201 711 No (0.6769650 0.3230350)
2) Sex=Male 1731 367 No (0.7879838 0.2120162)
    4) Age=Adult 1667 338 No (0.7972406 0.2027594) *
    5) Age=Child 64  29 No (0.5468750 0.4531250)
        10) Class=3rd 48  13 No (0.7291667 0.2708333) *
        11) Class=1st,2nd 16   0 Yes (0.0000000 1.0000000) *
3) Sex=Female 470 126 Yes (0.2680851 0.7319149)
   6) Class=3rd 196  90 No (0.5408163 0.4591837) *
   7) Class=1st,2nd,Crew 274  20 Yes (0.0729927 0.9270073) *
```

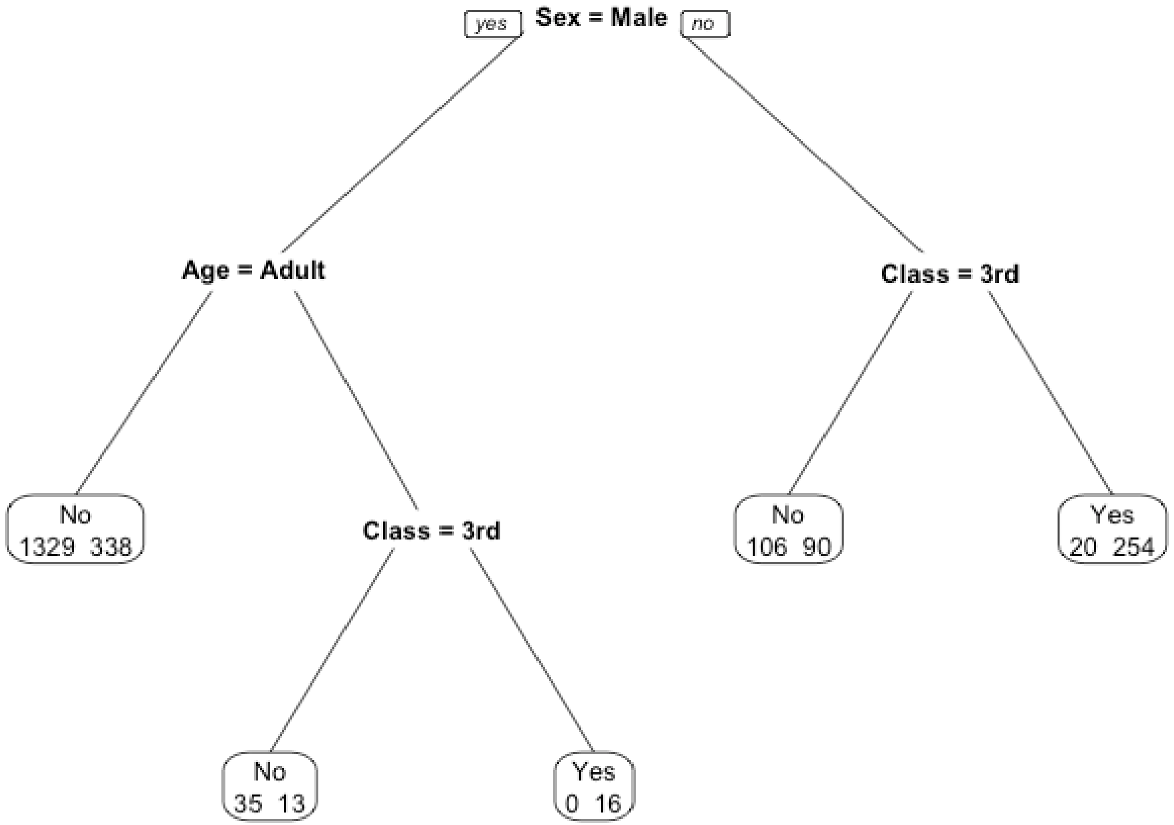
Example of rpart: For Titanic, Part III

```
library(rpart.plot)
prp(cartTree, faclen = 0, cex = 0.8, extra = 1)
```

For left pointing leaves:

- The first number: number of correctly classified cases
- The second number: number of incorrectly classified cases

For right pointing leaves, vice versa.



Predicting Values Using the Model

```
predictedSurvived <- predict(cartTree, newdata=titanic,  type = ”class")

titanic[1,]
    Class   Sex    Age    Survived
3    3rd   Male   Child      No

predictedSurvived[1]
 1
No
Levels: No Yes
```

What Was the Accuracy?

```
actualSurvived <- as.factor(titanic$Survived == "Yes")
confMatrix <- table(predictedSurvived,actualSurvived)
confMatrix
                actualSurvived
predictedSurvived   FALSE   TRUE
               No    1470    441
              Yes      20    270

accuracy <- 1 - (sum(confMatrix)  -  sum(diag(confMatrix)))  / sum(confMatrix)
accuracy
0.7905498
```

Questions

Is 79% good?

Using Classification and Regression Trees (cont.)

School of Information Studies
Syracuse University

Using Caret to Create Training and Test Sets

```
library(caret)

#Makes the sampling predictable
set.seed(111)

#Randomly sample elements to go into a training data set
trainList <- createDataPartition(y=titanic$Survived, p=.80, list=FALSE)

#Include all of those elements in the training set
trainSet <- titanic[trainList,]

#Construct test set from everything that didn't go into the training
testSet <- titanic[-trainList,]
```

Check Trainset

head(trainList)

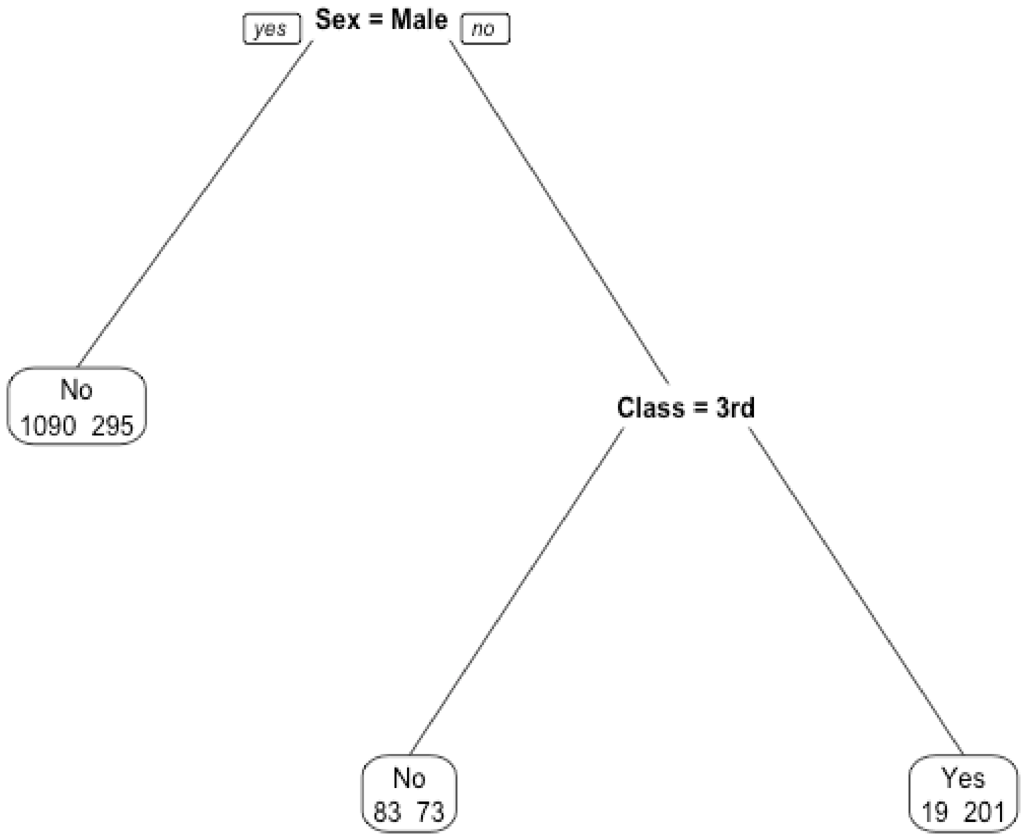
```
     Resample1
[1,]         1
[2,]         2
[3,]         4
[4,]         5
[5,]         6
[6,]         7
```

Note that some observations are skipped.

Use the Trainset to Build a Model Using ‘train’

```
cartTree  <- train(Survived ~ ., data=trainSet,
                   method="rpart")

prp(cartTree$finalModel, faclen = 0,
                   cex = 0.8, extra = 1)
```



Evaluate Using the Testset

```
#Note use of “raw”
predictValues <- predict(cartTree,newdata=testSet,  type = "raw”)

#Simpler to do confusion matrix
confusion <- confusionMatrix(predictValues, testSet$Survived)

confusion$overall[1]
Accuracy
0.7954545
```

Using all the data: the accuracy was 0.7905498

Exploring the Model Build via Caret

cartTree

CART

1761 samples

   3 predictor

   2 classes: 'No', 'Yes'

No pre-processing

Resampling: bootstrapped (25 reps)

Summary of sample sizes: 1761, 1761, 1761,...

Resampling results across tuning parameters:

  ...

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was cp = 0.009666081.

```
cartTree$finalModel
n= 1761

1) root 1761 569 No (0.67688813 0.32311187)
2) SexMale>=0.5 1385 295 No (0.78700361 0.21299639) *
3) SexMale< 0.5 376 102 Yes (0.27127660 0.72872340)
   6) Class3rd>=0.5 156  73 No (0.53205128 0.46794872) *
   7) Class3rd< 0.5 220  19 Yes (0.08636364 0.91363636) *
```

Exploring the Model Build via Caret (cont.)

```
predictValues <- predict(cartTree1,newdata=testSet, type = "raw")

confusionMatrix(predictValues, testSet$Survived)
```

Confusion matrix and statistics

Prediction	Reference FALSE	Reference TRUE
FALSE	297	89
TRUE	1	53

Accuracy : 0.7955

95% CI : (0.7547, 0.8322)

No information rate : 0.6773

p value [Acc > NIR] : 2.382e-08

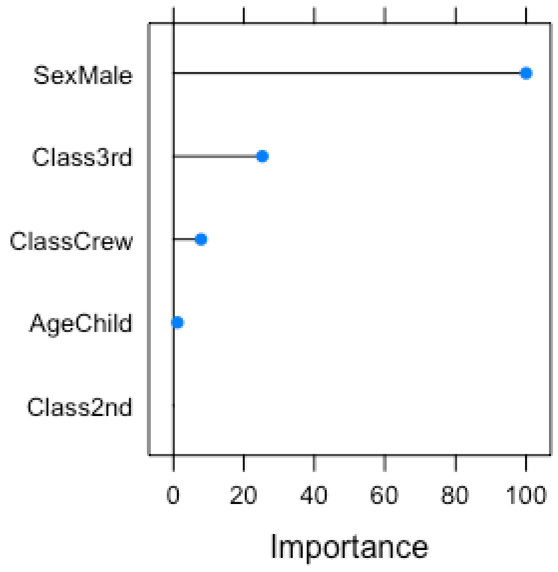
Exploring Variable Importance

varImp(cartTree)

rpart variable importance

	Overall
SexMale	100.000
Class3rd	25.233
ClassCrew	7.825
AgeChild	1.136

plot(varImp(cartTree))



Advanced Tree Example

School of Information Studies
Syracuse University

The German Credit Database

From the UCI machine learning repository

Built-in data set stored in the caret package

1,000 cases and 62 variables

"Class" is the outcome variable: binary creditworthiness ("good" or "bad")

eDuration	Age	NumberExistingCredits	NumberPeopleMaintenance	Telephone	ForeignWorker	Class
4	67	2	1	0	1	Good
2	22	1	1	1	1	Bad
3	49	1	2	1	1	Good
4	45	1	2	1	1	Good
4	53	2	2	1	1	Bad
4	35	1	2	0	1	Good
4	53	1	1	1	1	Good
2	35	1	1	0	1	Good

Showing 1 to 9 of 1,000 entries

The German Credit Database (cont.)

Example predictors

- Checking account status, duration, credit history, purpose of the loan, amount of the loan, savings accounts or bonds, installment rate in percentage of disposable income, other installment plans, number of existing credits
- Employment duration, personal information, other debtors/guarantors, residence duration, property, age, housing, job information, number of people being liable to provide maintenance for, telephone, and foreign worker status

For convenience, we will only use the first 10 columns, including “Class”

We will try “treebag”, a bagged CART; there are more than 200 fitting algorithms

See http://topepo.github.io/caret/modelList.html

Create Training and Test Sets

```
#Just grab a subset of the data for the demo
data("GermanCredit")
subCredit <- GermanCredit[,1:10]

#Makes the sampling predictable
set.seed(111)

#Randomly sample elements to go into a training data set
trainList <- createDataPartition(y=subCredit$Class,p=.80,list=FALSE)

#Create test and train data sets
trainSet <- subCredit[trainList,]
testSet <- subCredit[-trainList,]
```

Train Using a Treebag Model

```
fit1 <- train(Class ~ ., data=trainSet, method="treebag",
                                   preProc=c("center","scale"))
```

The **train()** command trains the specified model (here it is “**treebag**”)

Class ~ . This is the standard model specification syntax; the dot after the tilde includes all of the other variables as predictors; otherwise spell them out separated with plus signs

preProc= allows preprocessing of the input data, in this case taking the precaution of centering and scaling to put every input variable on the same scale

Interpret Results: Variable Importance

varImp(fit1)

```
treebag variable importance

                          Overall
Amount                     100.00
Age                         76.73
Duration                    57.75
ResidenceDuration           35.73
InstallmentRatePercentage   34.33
NumberExistingCredits       20.84
Telephone                   13.91
NumberPeopleMaintenance     11.77
ForeignWorker                0.00
```

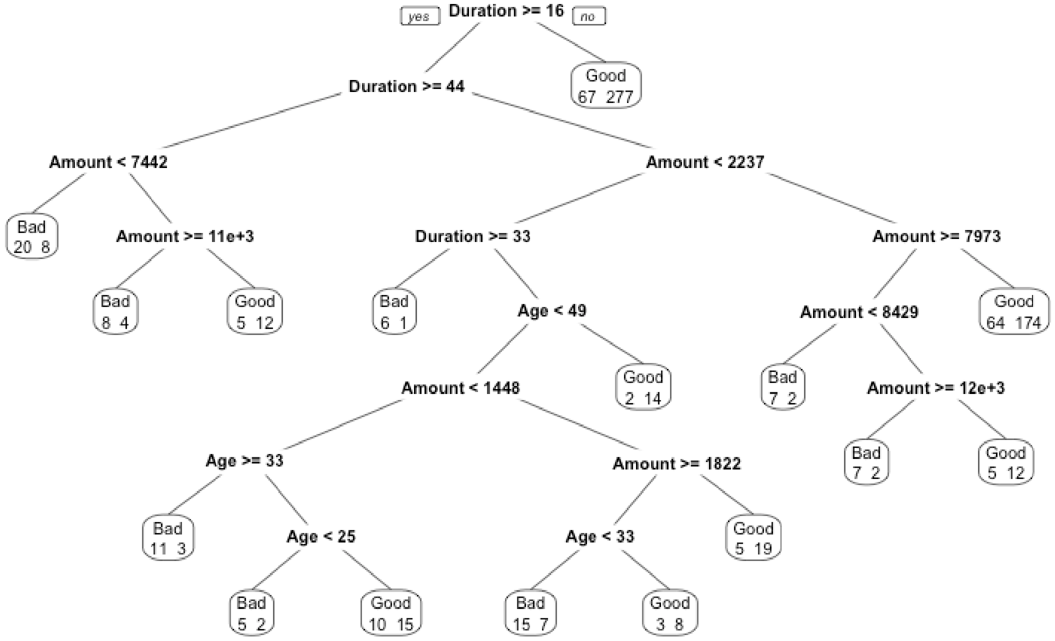
Interpret Results: Use rpart

Use rpart to display a simple CART tree with key variables

This tree gives a general idea of how variables are working, but is different from the treebag model (which has 25 trees!)

```
cartTree  <- rpart(Class ~ Amount + Age + Duratio
                        data = trainSet, method="(
```

```
prp(cartTree, faclen = 0, cex = 0.8, extra = 1)
```




```
treebag variable importance
                         Overall
Amount                    100.00
Age                        76.73
Duration                   57.75
. . .
```

Step 4: Assess Fit With New (Test) Data

```
predOut <- predict(fit1, newdata=testSet)
confusion <- confusionMatrix(predOut, testSet$Class)
confusion
```

Confusion matrix and statistics

```
          Reference
Prediction Bad Good
      Bad   20   18
      Good  40  122

     Accuracy : 0.71
```

Question

Is this a good model?

Confusion matrix and statistics

	Reference	
Prediction	Bad	Good
Bad	20	18
Good	40	122

Accuracy : 0.71

95% CI : (0.6418, 0.7718)

No information rate : 0.7

p value [Acc > NIR] : 0.412322

Advanced Tree Example (cont.)

School of Information Studies
Syracuse University

Interpret Results: Use rpart

No Information Rate: 0.7

Model Accuracy: 0.71

p value [Acc > NIR] : 0.412322

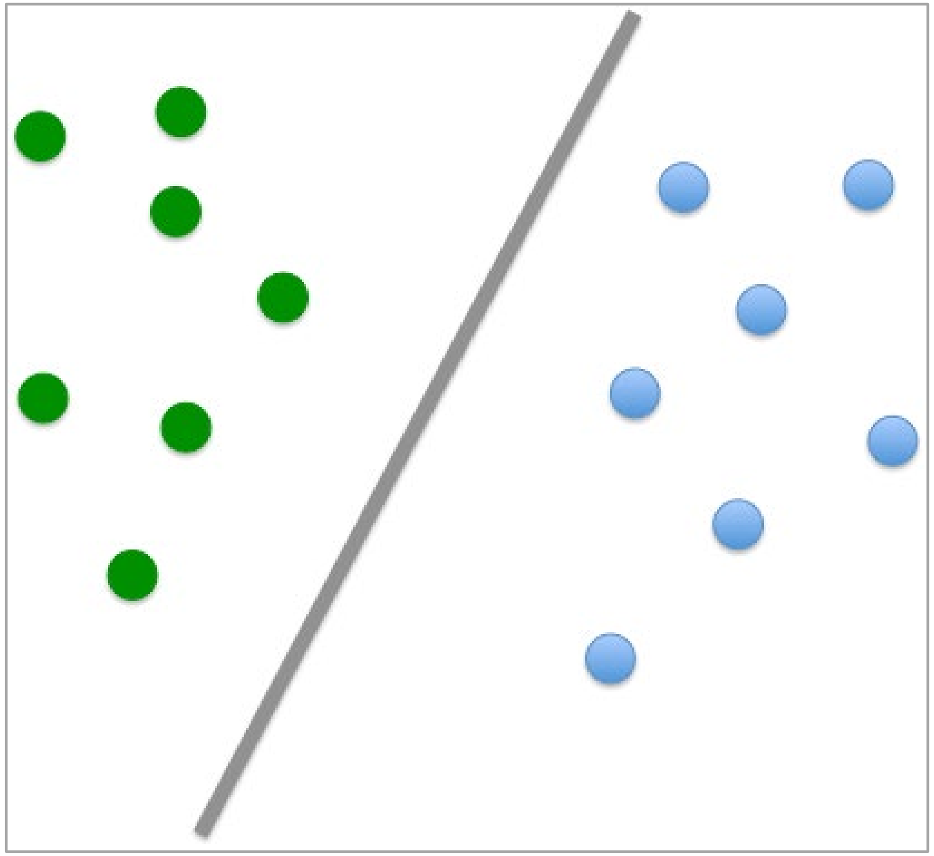
→ this suggests the improvement is not significant

Using
Support Vector Machines

School of Information Studies
Syracuse University

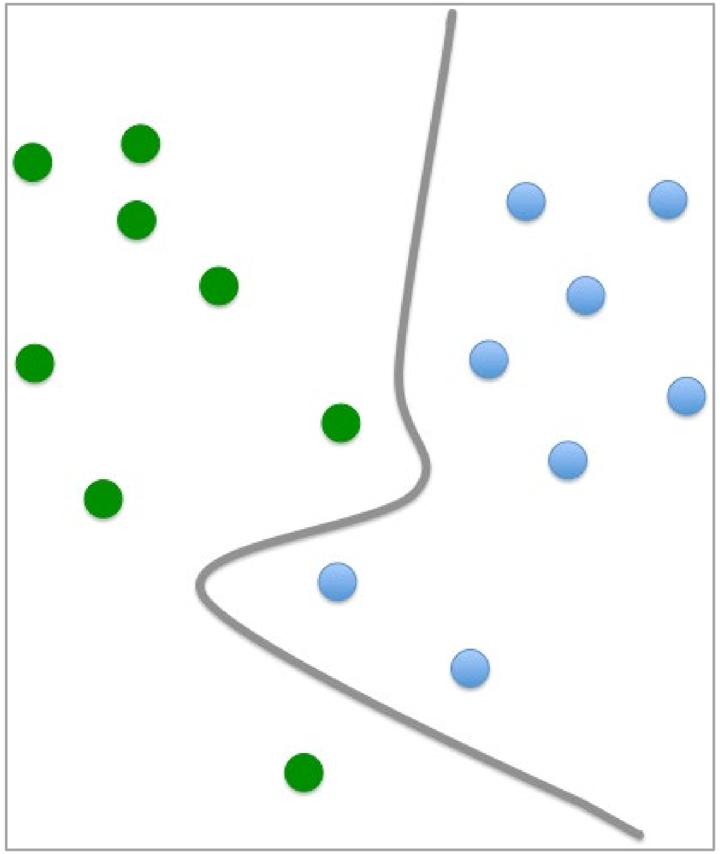
Visualizing the Algorithm, Part I

This is easy



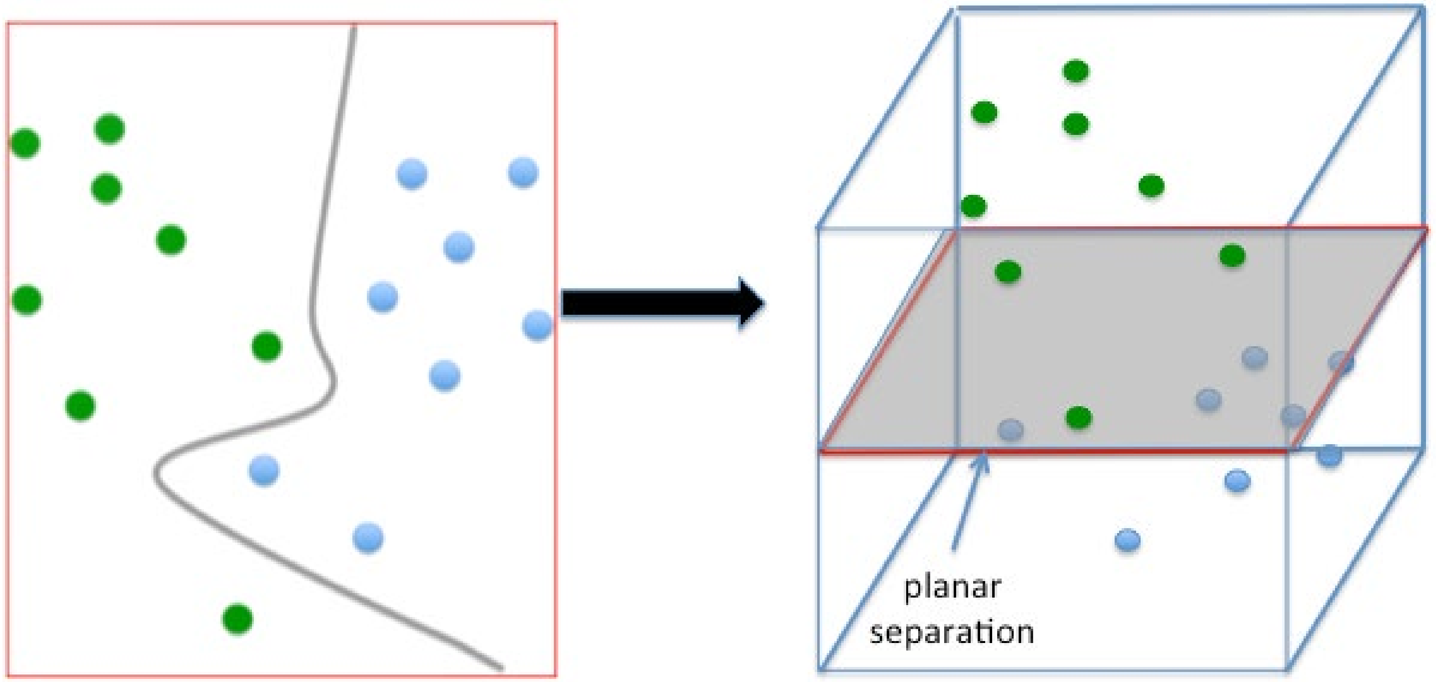
Visualizing the Algorithm, Part II

What about this?



Visualizing the Algorithm, Part III

2D to 3D mapping



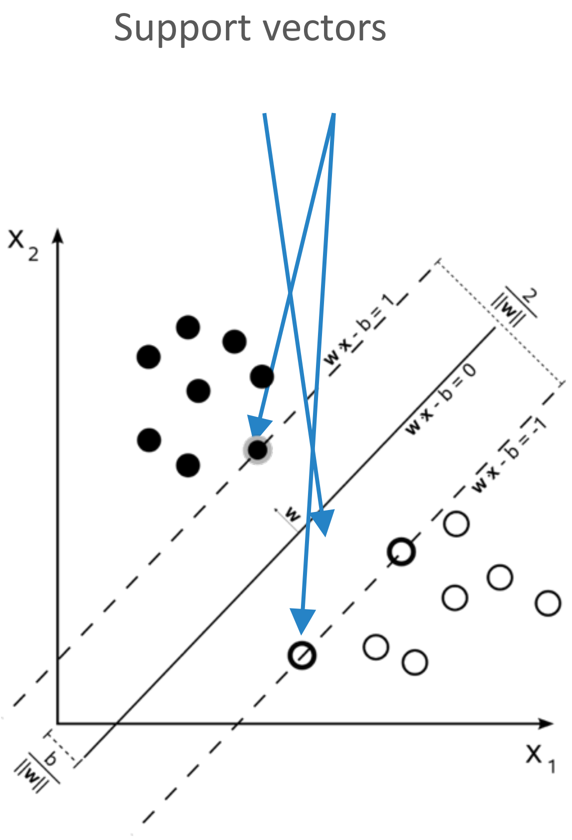
SVM: What Are Support Vectors?

Very **different approach to prediction** compared with the linear methods and tree methods

Based on **statistical learning theory**

Use the **cases that are difficult to predict** to help define a robust prediction model (because any model can handle “easy” cases!)

Support vectors ***are*** **harder-to-predict cases** (on a defined border and margin that separate the cases)



Use SVM to Explore the Credit Data

```
#Step 1: load the required packages that were already done

#Step 2: Create training and test sets
data("GermanCredit")
subCredit <- GermanCredit[,1:10]

#Makes the sampling predictable
set.seed(111)

#Randomly sample elements to go into a training data set
trainList <- createDataPartition(y=subCredit$Class,p=.80,list=FALSE)
trainSet <- subCredit[trainList,]
testSet <- subCredit[-trainList,]
```

Use SVM to Explore the Credit Data (cont.)

```
#Train the SVM model
fit2 <- train(Class ~ ., data=trainSet, method="svmRadial",
                                         preProc=c("center","scale"))

#Assess fit with new data
predOut <- predict(fit2, newdata=testSet)
```

Compare the Results

TreeBag

```
confMatrix <-table(predOut,testSet$Class)
confMatrix

predOut Bad Good
   Bad   20   18
   Good  40  122

prop.table(confMatrix)

predOut  Bad Good
   Bad  0.10 0.09
   Good 0.20 0.61

errorRate <- (sum(confMatrix) -
              sum(diag(confMatrix))) /
                      sum(confMatrix)
errorRate
[1] 0.29
```

SVM

```
confMatrix <- table(predOut,testSet$Class)
confMatrix

predOut Bad Good
   Bad   10    3
   Good  50  137

prop.table(confMatrix)

predOut  Bad Good
   Bad  0.05 0.015
   Good 0.25 0.685

errorRate <- (sum(confMatrix) -
              sum(diag(confMatrix))) /
                sum(confMatrix)
errorRate
[1] 0.265
```

Compare the Results (Varimp)

TreeBag

	Importance
Amount	100.00
Age	76.73
Duration	57.75
ResidenceDuration	35.73
InstallmentRatePerc	34.33
NumberExistingCredits	20.84
Telephone	13.91
NumberPeopleMaint.	11.77
ForeignWorker	0.00

SVM

	Importance
Duration	100.00
Age	51.92
Amount	43.20
InstallmentRatePerc	31.54
NumberExistingCredits	25.19
ForeignWorker	9.85
NumberPeopleMaint	7.01
ResidenceDuration	0.27
Telephone	0.00

Explore the Confusion Matrix

```
#Review the error—use the built in 'confusionMatrix'
confusion <- confusionMatrix(predOut, testSet$Class)
confusion
Confusion matrix and statistics

                 Reference
Prediction   Bad  Good
         Bad    10      3
        Good  50   137

            Accuracy : 0.735
              95% CI : (0.6681, 0.7948)
No information rate : 0.7
p value [Acc > NIR] : 0.1579
```

Explore the “C” Parameter

fit2

Support vector machines with radial basis function kernel

800 samples

9 predictor

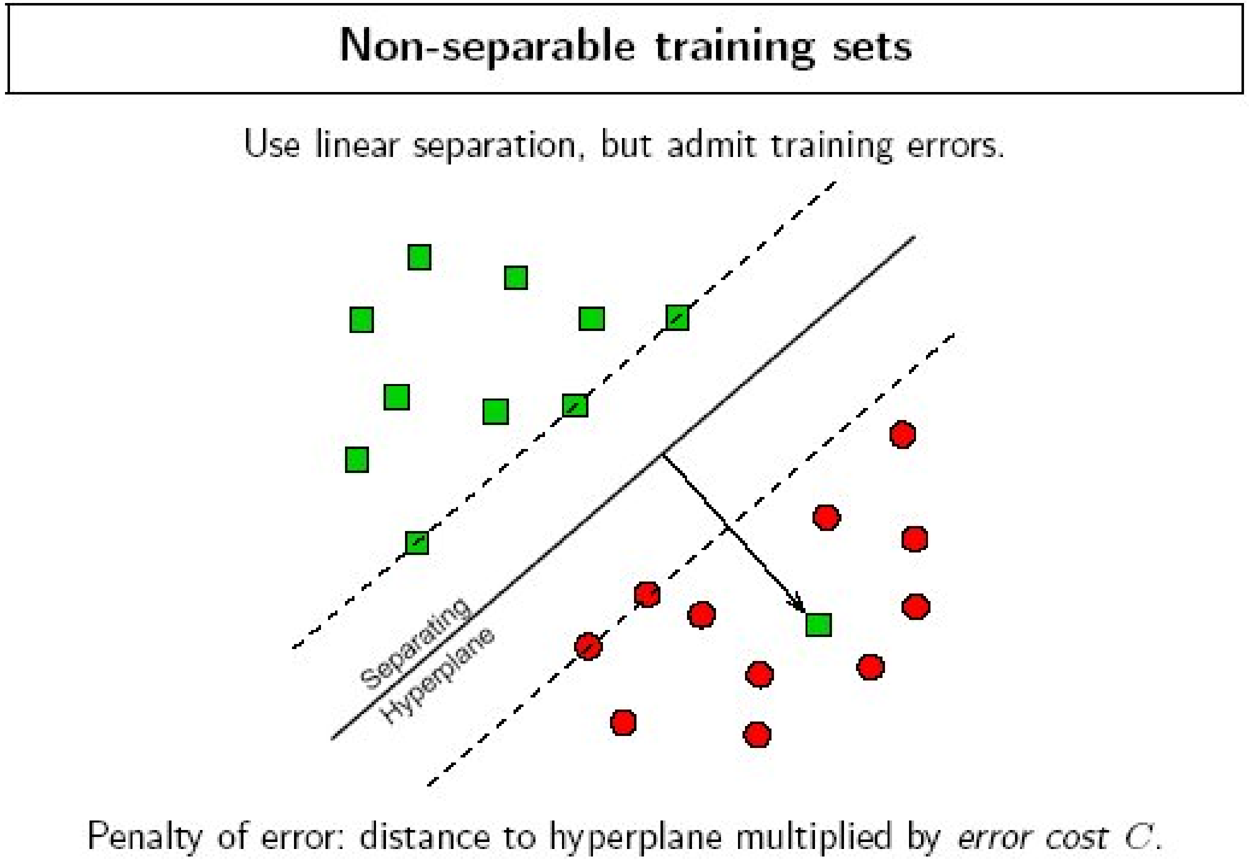
2 classes: 'Bad', 'Good'

Preprocessing: centered (9), scaled (9)

…

C	Accuracy
0.25	0.6969592
0.50	0.6974949
1.00	0.6989207

Why Error Might Not Be Bad



Cost Parameter ("C")

KSVM cost parameter impact summary

Higher cost "C" value	Fewer classification mistakes Fewer problem points	Smaller margin of separation	Specialized model	Higher cross-validation error	Lower training error
Lower cost "C" value	More classification mistakes More problem points	Higher margin of separation	Generalized model	Lower cross-validation error	Higher training error

Question

If SVM is such a ”black box”—where it is hard to know how the model actually works and what variables matter the most—why would we ever bother to use it?

Using Support Vector Machines (cont.)

School of Information Studies
Syracuse University

Answer

If SVM is such a ”black box”—where it is hard to know how the model actually works and what variables matter the most—why would we ever bother to use it?

Sometimes just need the best model possible

Sometimes can try to explain the model by showing how the model works with some specific examples