Machine Learning Algorithms Using R's Caret Package

Earl F Glynn

Kansas City R Users Group

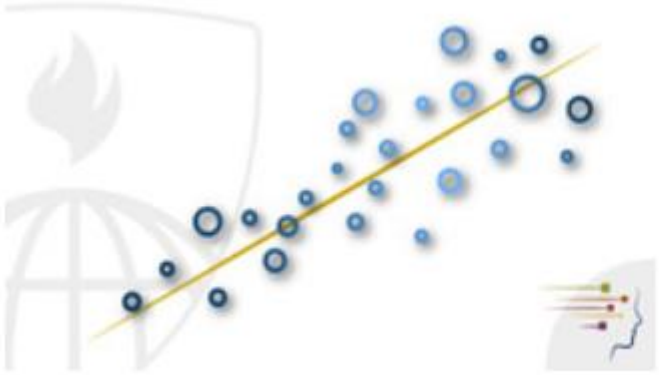
6 September 2014

Machine Learning Algorithms Using R's Caret Package

- Regressions Models vs Machine Learning
- Caret Package Features
  - ~170 Classification and Regression Models
  - Preprocessing
  - Data Splitting
  - Parallel Processing
- Examples Using Samsung Human Activity Dataset

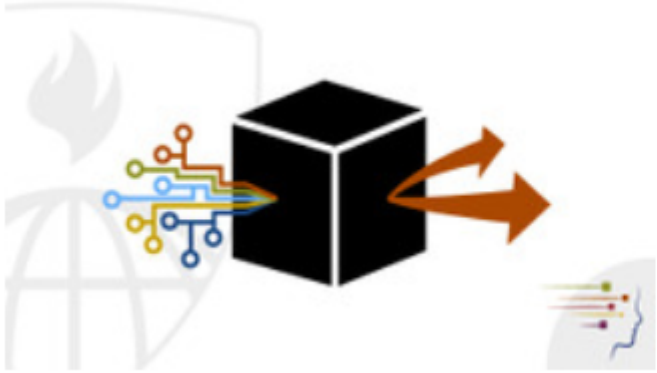
Regression Models vs. Practical Machine Learning

coursera



Johns Hopkins University
Regression Models

Focus: Interpretability



Johns Hopkins University
Practical Machine Learning

Focus: Accurate Predictions

https://www.coursera.org/specialization/jhudatascience/1

Caret Package:
Classification And Regression Training

http://topepo.github.io/caret/index.html

Features

- Uniform interface.
- Standardized common tasks.

Caret Package: Models

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

Almost 170 different models …

Model	method Argument Value	Type	Packages	Tuning Parameters
Random Forest	rf	Dual Use	randomForest	mtry
Random Ferns	rFerns	Classification	rFerns	depth
Factor-Based Linear Discriminant Analysis	RFlda	Classification	HiDimDA	q
Ridge Regression	ridge	Regression	elasticnet	lambda
Random k-Nearest Neighbors	rknn	Dual Use	rknn	k, mtry
Random k-Nearest Neighbors with Feature Selection	rknnBel	Dual Use	rknn, plyr	k, mtry, d
Robust Linear Model	rlm	Regression	MASS	None
ROC-Based Classifier	rocc	Classification	rocc	xgenes
CART	rpart	Dual Use	rpart	cp
CART	rpart2	Dual Use	rpart	maxdepth

Caret Package: Pre-Processing

- Creating Dummy Variables
- Zero- and Near Zero-Variance Predictors
- Identifying Correlated Predictors
- Linear Dependencies
- Centering and Scaling
- Imputation

http://topepo.github.io/caret/preprocess.html

Caret Package: Data Splitting

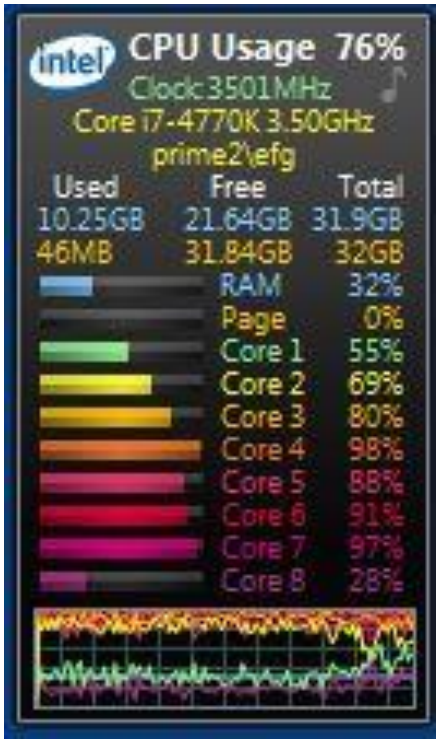
The function **createDataPartition** can be used to create balanced splits of the data.

http://topepo.github.io/caret/splitting.html

Caret Package: Parallel Processing

On a PC …

```
# Setup parallel processing
# Let's use 6 cores
library(doParallel)
rCluster <- makePSOCKcluster(6)
registerDoParallel(rCluster)
```



Samsung Human Activity Dataset

Human Activity Recognition Using Smartphones Data Set

Download: Data Folder, Data Set Description

Abstract: Human Activity Recognition database built from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	10299	**Area:**	Computer
Attribute Characteristics:	N/A	**Number of Attributes:**	561	**Date Donated**	2012-12-10
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	110387

Goal is to predict 1 of 6 human activities based on 561 “features.”

http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

Samsung Human Activity Dataset

GitHub Repository:

https://github.com/earlglynn/kc-r-users-caret

Files:

0-Download-UCI-Dataset.R

1-Load-UCI-Data.R

2-caret-machine-learning.R

Samsung Human Activity Dataset

Process raw data into single data.frame

- 0-Download-UCI-Dataset.R
- 1-Load-UCI-Data.R

Data recorded about 6 activities for 30 subjects.

10,299 records of 561 features.

	Lying	Sit	Stand	Walk	WalkDown	WalkUp
1	50	47	53	95	49	53
2	48	46	54	59	47	48
3	62	52	61	58	49	59
4	54	50	56	60	45	52
5	52	44	56	56	47	47
6	57	55	57	57	48	51
7	52	48	53	57	47	51
8	54	46	54	48	38	41
9	50	50	45	52	42	49
10	58	54	44	53	38	47
11	57	53	47	59	46	54
12	60	51	61	50	46	52
13	62	49	57	57	47	55
14	51	54	60	59	45	54
15	72	59	53	54	42	48
16	70	69	78	51	47	51
17	71	64	78	61	46	48
18	65	57	73	56	55	58
19	83	73	73	52	39	40
20	68	66	73	51	45	51
21	90	85	89	52	45	47
22	72	62	63	46	36	42
23	72	68	68	59	54	51
24	72	68	69	58	55	59
25	73	65	74	74	58	65
26	76	78	74	59	50	55
27	74	70	80	57	44	51
28	80	72	79	54	46	51
29	69	60	65	53	48	49
30	70	62	59	65	62	65

Samsung Human Activity Dataset

561 features listed in “features.txt”:

1 tBodyAcc-mean()-X
2 tBodyAcc-mean()-Y
3 tBodyAcc-mean()-Z
4 tBodyAcc-std()-X
5 tBodyAcc-std()-Y
6 tBodyAcc-std()-Z
7 tBodyAcc-mad()-X
8 tBodyAcc-mad()-Y
9 tBodyAcc-mad()-Z
10 tBodyAcc-max()-X
. . .

. . .
552 fBodyBodyGyroJerkMag-meanFreq()
553 fBodyBodyGyroJerkMag-skewness()
554 fBodyBodyGyroJerkMag-kurtosis()
555 angle(tBodyAccMean,gravity)
556 angle(tBodyAccJerkMean),gravityMean)
557 angle(tBodyGyroMean,gravityMean)
558 angle(tBodyGyroJerkMean,gravityMean)
559 angle(X,gravityMean)
560 angle(Y,gravityMean)
561 angle(Z,gravityMean

Load Samsung Data

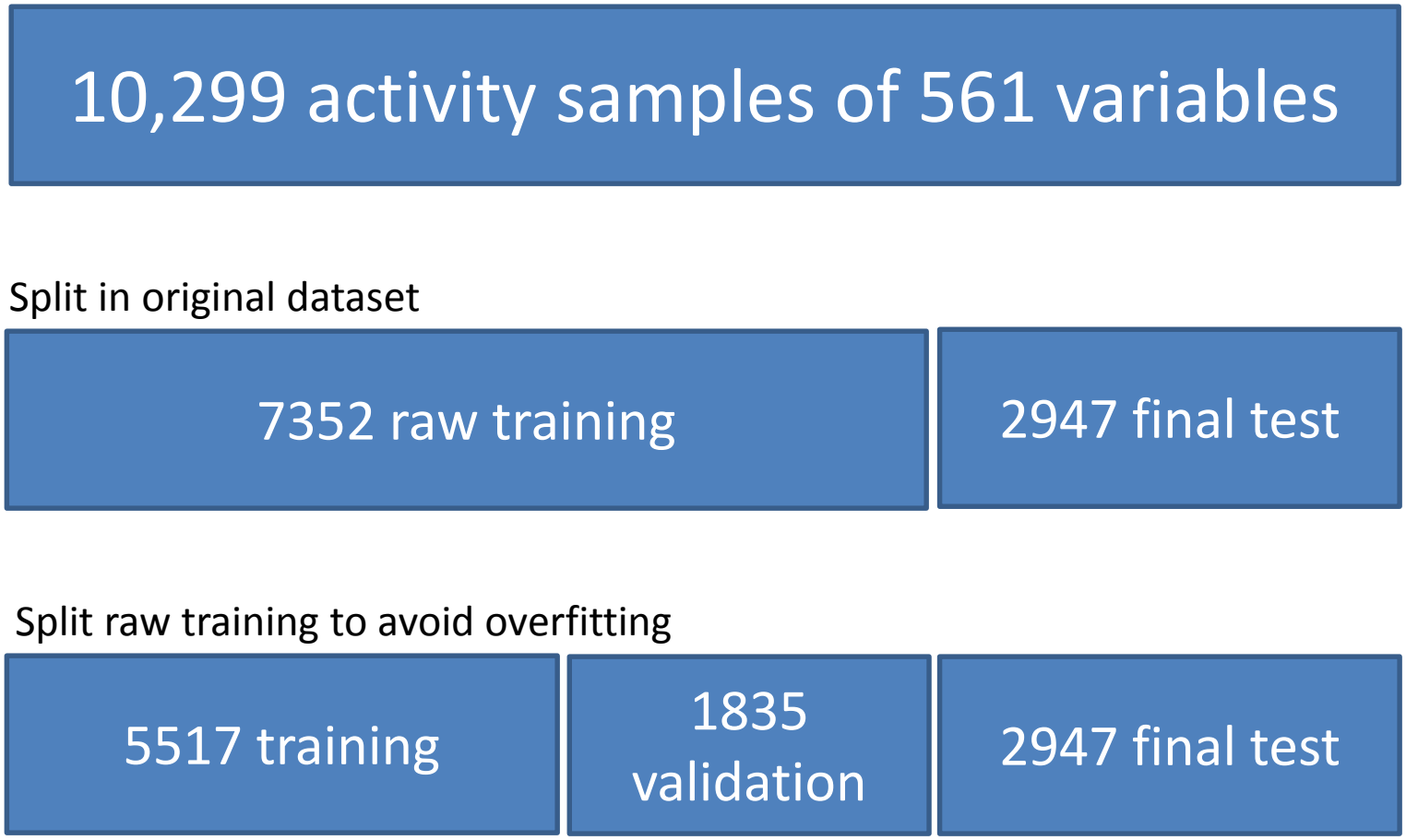
```
Samsung <- read.csv("Samsung-Human-Activity.csv")
dim(Samsung)    # 10299 564

# Extract training set
rawTrain  <- Samsung[Samsung$source == "train",-1:-2]
dim(rawTrain)    # 7352  562

# Extract "final" test set only to be used once
finalTest <- Samsung[Samsung$source == "test", -1:-2]
dim(finalTest)    # 2947  562
```

File: 2-caret-machine-learning.R

Approach Using Samsung Data

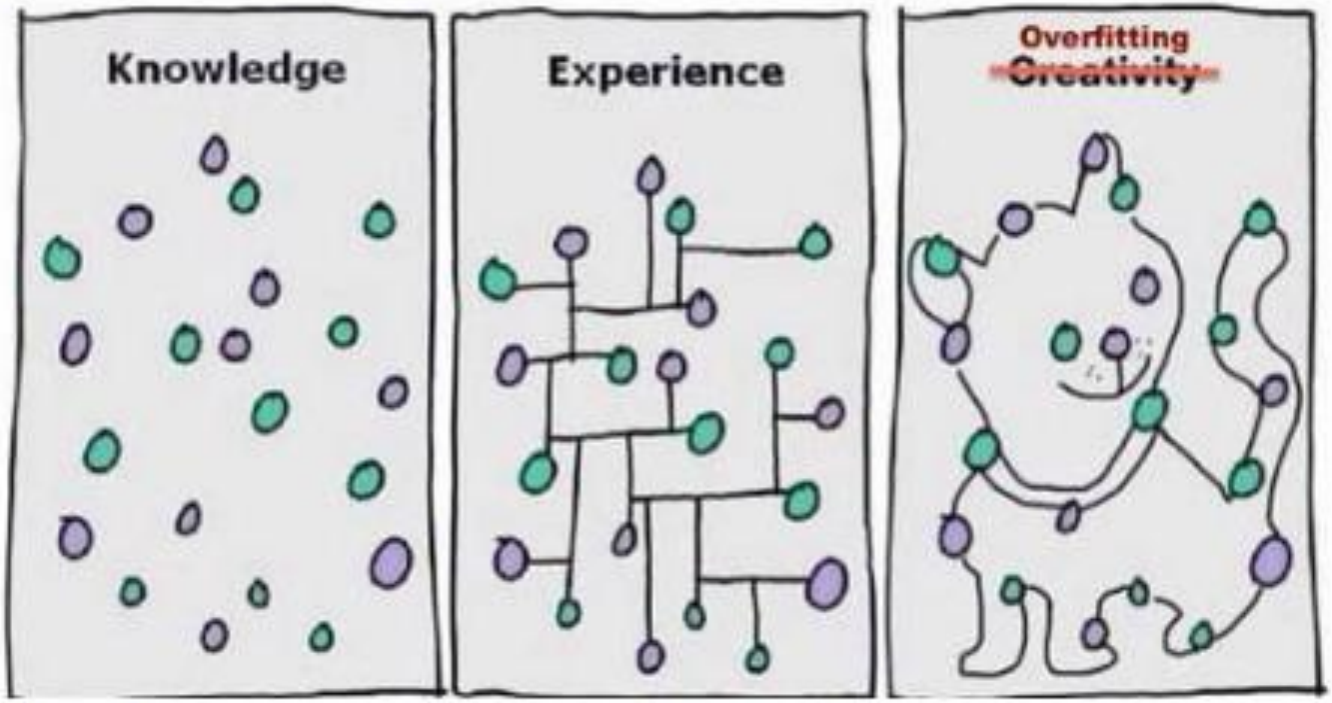


Why avoid overfitting?

Retweeted by Software Carpentry

Alberto Cairo @albertocairo · Aug 11

So cool MT @DiegoKuonen: @Ted_Underwood: With apologies to @ResearchMark, an edited version of this popular diagram. pic.twitter.com/IqfuPfmmyH



Expand Reply Retweet Favorite More

Remove near-zero variance variables

```
nzv <- nearZeroVar(rawTrain, saveMetrics=TRUE)
count.nzv <- sum(nzv$nzv)
count.nzv
if (count.nzv > 0)
{
  rawTrain  <- rawTrain[,  !nzv$nzv]
  finalTest <- finalTest[, !nzv$nzv]
}
```

File: 2-caret-machine-learning.R

Remove variables with high correlation

```
CUTOFF <- 0.90
cor.matrix <- cor(rawTrain[,-1])
cor.high   <- findCorrelation(cor.matrix, CUTOFF)

high.cor.remove <-
row.names(cor.matrix)[cor.high]
high.cor.remove
length(high.cor.remove)

rawTrain <- rawTrain[,   -cor.high]
finalTest <- finalTest[, -cor.high]
```

Partition into training and validation sets

```
TRAIN.PERCENT <- 0.75
inTrainSetIndex <-
createDataPartition(y=rawTrain$activity,
                    p=TRAIN.PERCENT, list=FALSE)

training   <- rawTrain[ inTrainSetIndex,]
dim(training)

validation <- rawTrain[-inTrainSetIndex,]
dim(validation)
```

File: 2-caret-machine-learning.R

Use LDA with caret's train

```
PREPROCESS <- NULL
PREPROCESS <- c("center", "scale")
METHOD <- "lda"
fit <- train(activity ~ ., data = training,
             preProcess=PREPROCESS, method=METHOD)
OutOfSample  <- predict(fit, newdata=validation)

confusion <- confusionMatrix(validation$activity,
   OutOfSample)

dotPlot(varImp(fit), main="lda:  Dotplot of variable
importance values")
```

File: 2-caret-machine-learning.R

Results using Linear Discriminant Analysis

```
Confusion Matrix and Statistics

          Reference
Prediction Lying Sit Stand Walk WalkDown WalkUp
  Lying       351   0     0    0        0      0
  Sit           5 295    21    0        0      0
  Stand         0  14   329    0        0      0
  Walk          0   0     0  305        0      1
  WalkDown      0   0     0    2      236      8
  WalkUp        0   0     0    3        0    265

Overall Statistics

               Accuracy : 0.9706
                 95% CI : (0.9618, 0.9778)
```

Results using Linear Discriminant Analysis

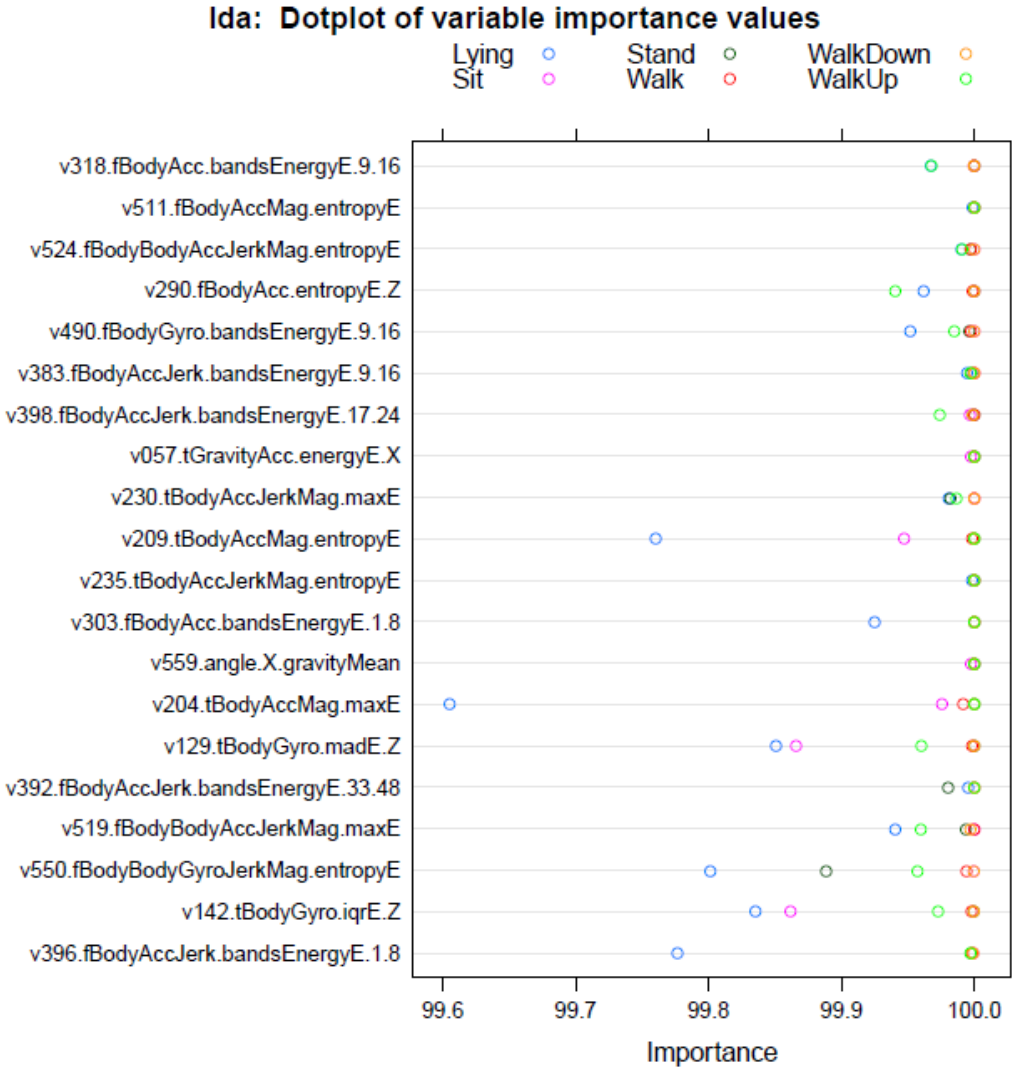
Statistics by Class:

	Class: Lying	Class: Sit	Class: Stand	Class: Walk	Class: WalkDown	Class: WalkUp
Sensitivity	0.9860	0.9547	0.9400	0.9839	1.0000	0.9672
Specificity	1.0000	0.9830	0.9906	0.9993	0.9937	0.9981
Pos Pred Value	1.0000	0.9190	0.9592	0.9967	0.9593	0.9888
Neg Pred Value	0.9966	0.9908	0.9859	0.9967	1.0000	0.9943
Prevalence	0.1940	0.1684	0.1907	0.1689	0.1286	0.1493
Detection Rate	0.1913	0.1608	0.1793	0.1662	0.1286	0.1444
Detection Prevalence	0.1913	0.1749	0.1869	0.1668	0.1341	0.1460
Balanced Accuracy	0.9930	0.9688	0.9653	0.9916	0.9969	0.9826

http://en.wikipedia.org/wiki/Sensitivity_and_specificity

		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence = Σ Condition positive / Σ Total population	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV, Precision) = Σ True positive / Σ Test outcome positive	False discovery rate (FDR) = Σ False positive / Σ Test outcome positive
	Test outcome negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = Σ False negative / Σ Test outcome negative	Negative predictive value (NPV) = Σ True negative / Σ Test outcome negative
	Positive likelihood ratio (LR+) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = Σ True positive / Σ Condition positive	False positive rate (FPR, Fall-out) = Σ False positive / Σ Condition negative	Accuracy (ACC) = Σ True positive + Σ True negative / Σ Total population	
	Negative likelihood ratio (LR−) = FNR/TNR	False negative rate (FNR) = Σ False negative / Σ Condition positive	True negative rate (TNR, Specificity, SPC) = Σ True negative / Σ Condition negative		
	Diagnostic odds ratio (DOR) = LR+/LR−				

Results using Linear Discriminant Analysis



Results From Variety of Machine Learning Methods

Method Name	Caret Method	Run time	Accuracy
Stochastic Gradient Boosting	gbm	9.3 min	0.988
Support Vector Machine – Polynomial	svmPoly	16.8 min	0.984
Random Forest	rf	18.9 min	0.980
Bagged Flexible Discriminant Analysis	bagFDA	5.6 hours	0.974
Linear Discriminant Analysis	lda	40.5 sec	0.971
Bagged CART	treebag	10.7 min	0.970
Naïve Bayes	nb	14.5 min	0.820
Classification and Regression Trees (CART)	rpart	1.4 min	0.544

Machine Learning Algorithms Using R's Caret Package

Future

- Explore combining models to form hybrids. Test once with "final test" dataset.
- Explore many of the other Caret algorithms.
- Characterize accuracy, run time, and memory usage for a "toy" problem.

Machine Learning Algorithms Using R's Caret Package Summary

- Caret package provides uniform approach to using many classification and regression algorithms.
- Be cautious about applying "black box" machine learning approaches.