**Documentation For R Machine Learning Pipeline Using Caret**

29 October 2018

Author:

Neal Cariello

Senior Toxicologist

Integrated Laboratory Systems, Research Triangle Park, NC

Supporting:

NTP Interagency Center for the Evaluation of Alternative Toxicological Methods (NICEATM)

<https://www.niehs.nih.gov/research/atniehs/dntp/assoc/niceatm/index.cfm>

email: [cariello@niehs.nih.gov](mailto:cariello@niehs.nih.gov), [ncariello@ils-inc.com](mailto:ncariello@ils-inc.com), [neal.cariello@gmail.com](mailto:neal.cariello@gmail.com)

Contents

[Summary 2](#_Toc528674001)

[Directory Structure 3](#_Toc528674002)

[Recommendations 5](#_Toc528674003)

[Test Code First Using Data In R Packages 5](#_Toc528674004)

[Load User Data 6](#_Toc528674005)

[User Options 6](#_Toc528674006)

[Output 6](#_Toc528674007)

[Data Summary 6](#_Toc528674008)

[Text Statistics Of Variables 6](#_Toc528674009)

[Plotting Of Variables 7](#_Toc528674010)

[Visualization Of Variables Using PCA and t-SNE 7](#_Toc528674011)

[Correlation Plot Of Variables 11](#_Toc528674012)

[Variable Importance Plot 12](#_Toc528674013)

[Model Tuning Plot 13](#_Toc528674014)

[Performance Plots For Models 14](#_Toc528674015)

[Classification 14](#_Toc528674016)

[Confusion Matrix Statistics 15](#_Toc528674017)

[Regression 16](#_Toc528674018)

[Statistical Comparison Of Models 18](#_Toc528674019)

[R Environment Is Saved 19](#_Toc528674020)

[Datasets In R For Initial Testing 20](#_Toc528674021)

[Classification 20](#_Toc528674022)

[Regression 21](#_Toc528674023)

[Miscellaneous 22](#_Toc528674024)

[Caret functions 22](#_Toc528674025)

[Main author-defined functions 22](#_Toc528674026)

[ModelFit() 22](#_Toc528674027)

[trn\_ctl() 23](#_Toc528674028)

[MissingValueImpute() 24](#_Toc528674029)

[Required packages and libraries 24](#_Toc528674030)

[Required files 25](#_Toc528674031)

[Table of Contents of The Book "Applied Predictive Modeling" 25](#_Toc528674032)

# Summary

This program allows rapid assessment of a variety of machine learning algorithms for classification and regression predictions. The difference between the classification and regression is data type is being predicted:

* For classification, what is being predicted has a binary outcome (0/1, TRUE/FALSE, toxic/non-toxic, etc.)
  + The datatype must be a factor with 2 levels
* For regression, what is being predicted is a number.
  + The datatype can be integer or numeric

With the exception of what you are trying to predict, **all data must be an integer or numeric data type**. Factor variables in the dataset are automatically deleted. The next iteration of this program will allow additional factor variables.

The classification or regression model is selected as follows:

MODEL <- 'CLASSIFICATION'

or

MODEL <- 'REGRESSION'

Data for modeling must not contain any missing values. Missing values are automatically detected and imputed as follows:

* Delete rows with > 10% missing values
* Delete columns with > 10% missing values
* Missing values are imputed using the k-Nearest Neighbor (kNN) method in the package DMwR
  + If the data has so many missing variables that the kNN method fails, median imputation is used

Multiple machine learning algorithms can be used to easily evaluate different models sequentially using the syntax:

MLmethods <- c('rf', 'svmRadial', 'xgbLinear', ETC). Not all algorithms will work with a given dataset.

R Markdown is used so all code and output is in a single HTML file. The R environment is saved so that the code does not have to be executed to examine the models. The code can take many hours to execute depending on the size of the data and the machine learning methods selected.

The workflow uses the R package caret which greatly simplifies coding and presents a unified interface over 200 machine learning algorithms. The caret package varies tuning parameters when appropriate for a given machine learning method and also calculates variable importance.

The main caret page is at <http://topepo.github.io/caret/index.html>

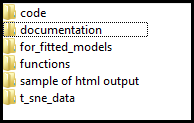
A book by Max Kuhn who is the caret package author is **highly recommended**. The title is "Applied Predictive Modeling" and it is available from Amazon.com:

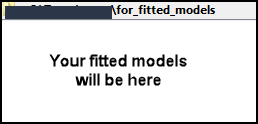
<https://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485>

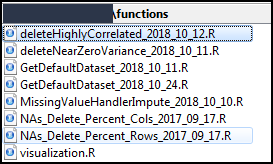
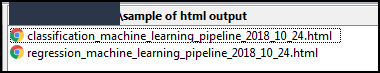
This book provides very good explanations of machine learning principles and the code examples use the caret package.

# Directory Structure

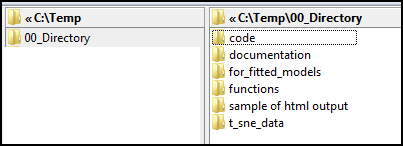
After downloading from Git, the following subdirectories and contents should exist.



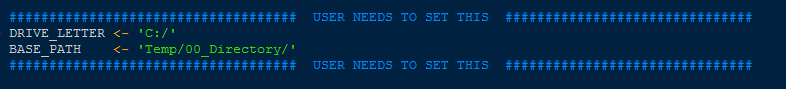
  

You need to set the DRIVE LETTER and BASE\_PATH to the directory above the main subdirectories. For example, if the subdirectory structure is as follows



Then the code would be



# Recommendations

* Scale the input data to create a standardized normal distribution for each variable (column) if the variables have very different ranges. Each variable will then have a mean of 0 and a standard deviation of 1, this is also called a Z-Score[[1]](#footnote-1)
  + This is the default and scaling occurs in the Model Fit() function.
* Remove highly-correlated variables
  + It is important to removed highly-correlated variables since they may be reporting on the same features and not removing them may lead to over-optimistic performance metrics, meaning that the model is actually worse that it appears.
* Remove variables with near-zero variance
  + Features with near-zero variance contribute little to the model and will increase compute time.

# Test Code First Using Data In R Packages

The recommendation is to test the code first with known datasets, namely caret::segmentationData for classification and mlbench ::BostonHousing for regression. The datasets are described in section *Datasets In R For Initial Testing.* If this data runs correctly, then the user can load their dataset of interest.

To load the segmentationData or BostonHousing, set USE\_DEFAULT\_DATA to TRUE as is shown below.

if (USE\_DEFAULT\_DATA == TRUE) {

dataRaw <- returnDefaultDataset(MODEL, ADD\_NA)

}

Setting ADD\_NA to TRUE adds NAs to the dataset. This is used to test data imputation.

# Load User Data

Assuming that the test data described in *Test Code First Using Data In R Packages* runs correctly, the user can load their data. There **\*MUST\*** be a column labelled "toPredict", which is what is being predicted (d'oh). The dataframe **\*MUST\*** be named "dataRaw".

Not all machine learning algorithms will work with a given dataset, a trial-and-error approach must be used.

# User Options

User options which are specified in code include:

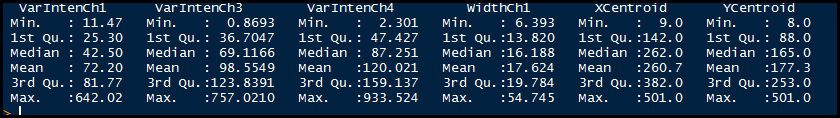
* Use parallel processing
* Selection of machine learning methods.
* Remove low variance variables
  + These variables add little to the model and increase CPU time for little benefit.
* Remove highly correlated variables
  + These variables may be reporting on the same property and if allowed in the model it will inflate model performance. That is, the model will look better than it really is.
* Data transformation before model building
  + A variety of transformation and scaling options are available as arguments to the caret::train() functions

# Output

R Markdown is used so all code and output is in a single HTML file.

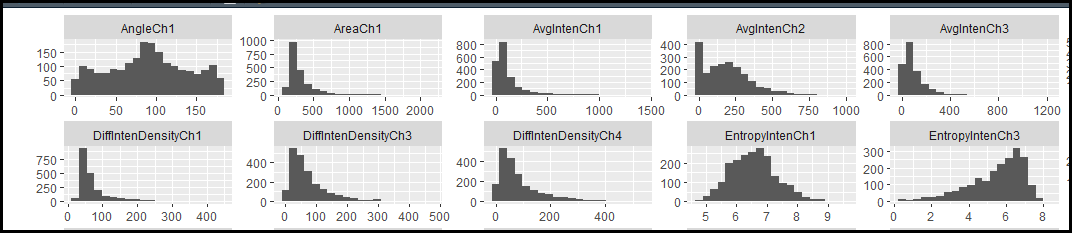
# Data Summary

## Text Statistics Of Variables



## Plotting Of Variables

If you have more than 75 variables, the plot will get pretty squished.



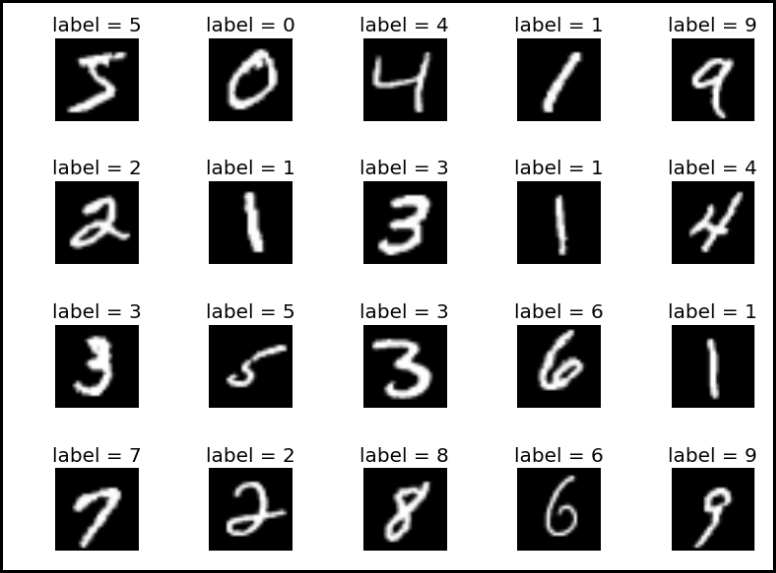
## Visualization Of Variables Using PCA and t-SNE

The default is to scale the data to mean 0 and standard deviation 1 (also called a Z-Score transform).

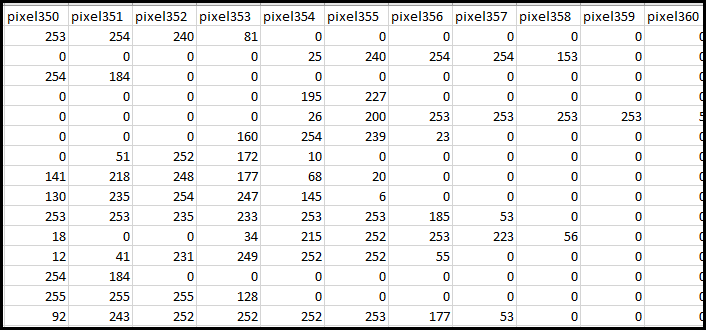
Both Principle Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are data visualization techniques. Visualization is controlled in code by VISUALIZATION == TRUE / FALSE and is applied only to Classification data.

The main difference between the two is that PCA uses linear relationships between variables while t-SNE uses non-linear relationships between variables. Depending on the dataset, the results can be strikingly different. The difference can be seen nicely using the classic MNIST dataset for handwritten digit recognition.

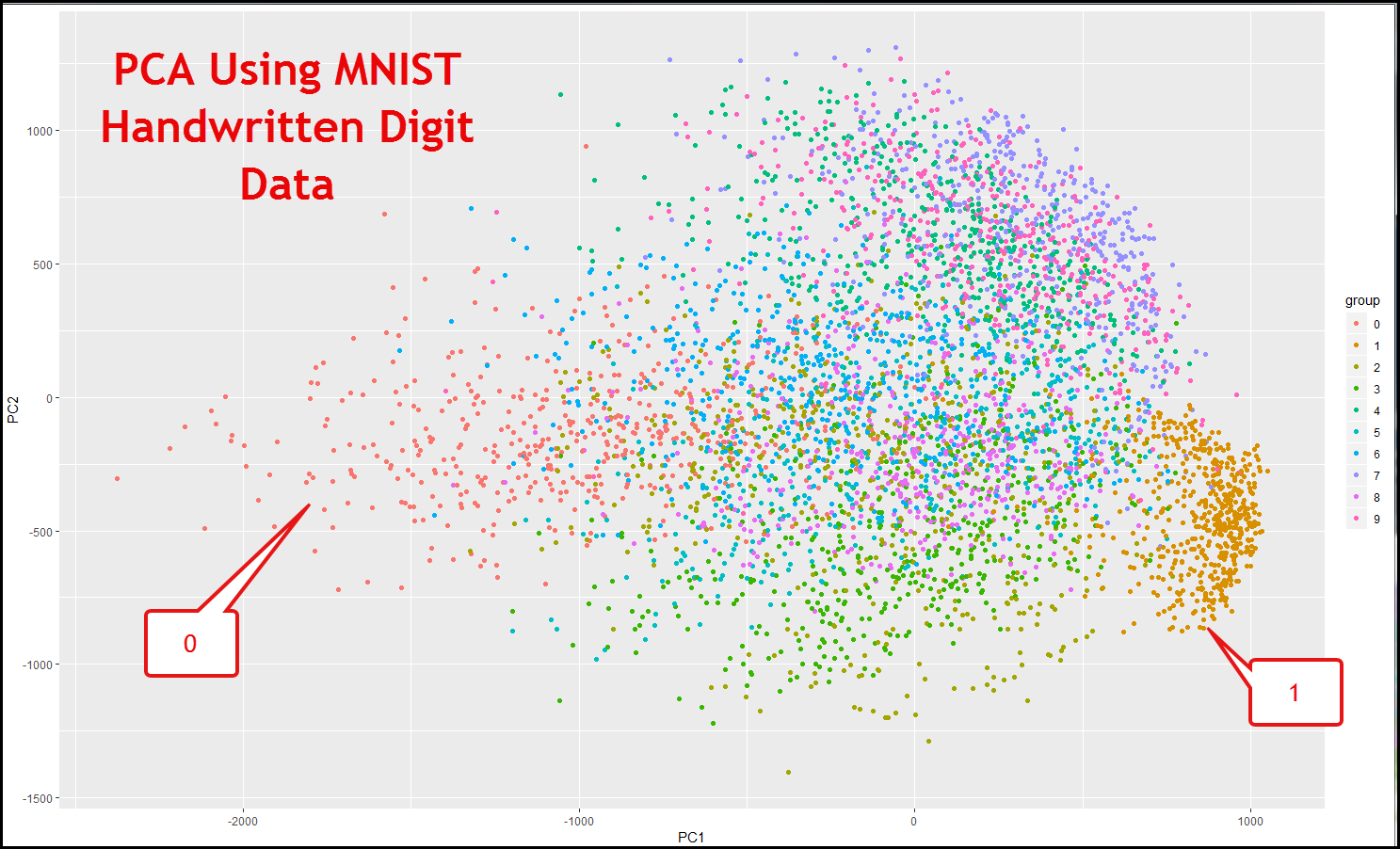
The data is labeled 0 - 9.



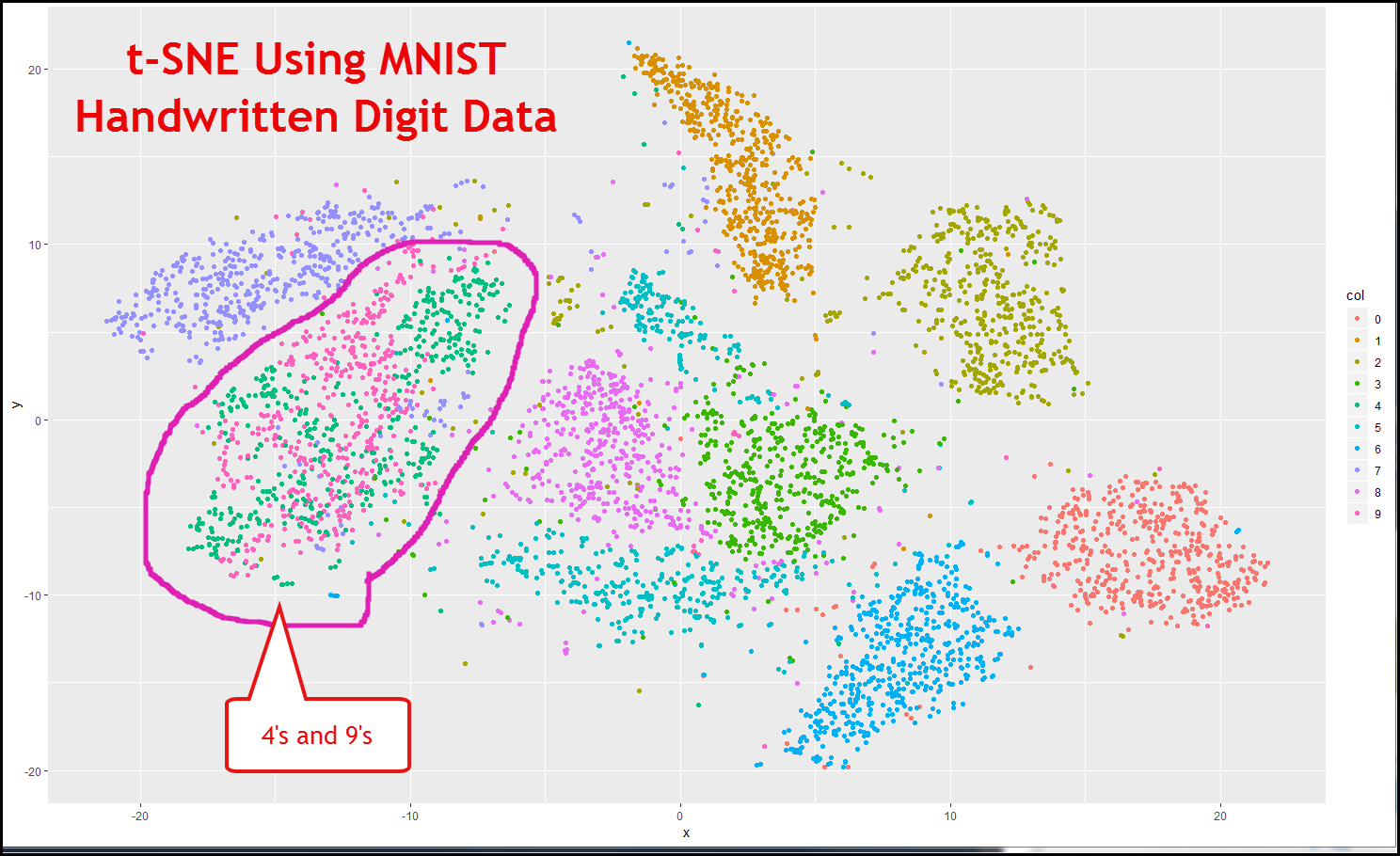
Each picture (number) is given as a row with 784 numbers. The numbers range from 0 (pure white) to 255 (pure black).



With the exception of 0's and 1's, PCA does not resolve the digits as shown below. Looking at the PCA plot, the experimenter may conclude that there is very little structure in the data.



However, the situation is quite different using t-SNE as shown below.



4's and 9's are not well-resolved, which is not surprising when you think about how the digits are drawn. Looking at the t-SNE plot, the experimenter would conclude that the data has significant structure. Code to reproduce these figures is in the visualization.R file and the MNIST data is in the t\_sne\_data subdirectory.

For more t-SNE information see

<https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/>

<https://www.displayr.com/using-t-sne-to-visualize-data-before-prediction/>

# Correlation Plot Of Variables

Highly correlated variables may be reporting on the same property and if allowed in the model it will inflate model performance. That is, the model will look better than it really is. The correlation between variables is shown in Figure 1.

If more than ~75 variables are used in the model the plot may be too crowded and the variable names may be difficult to read.

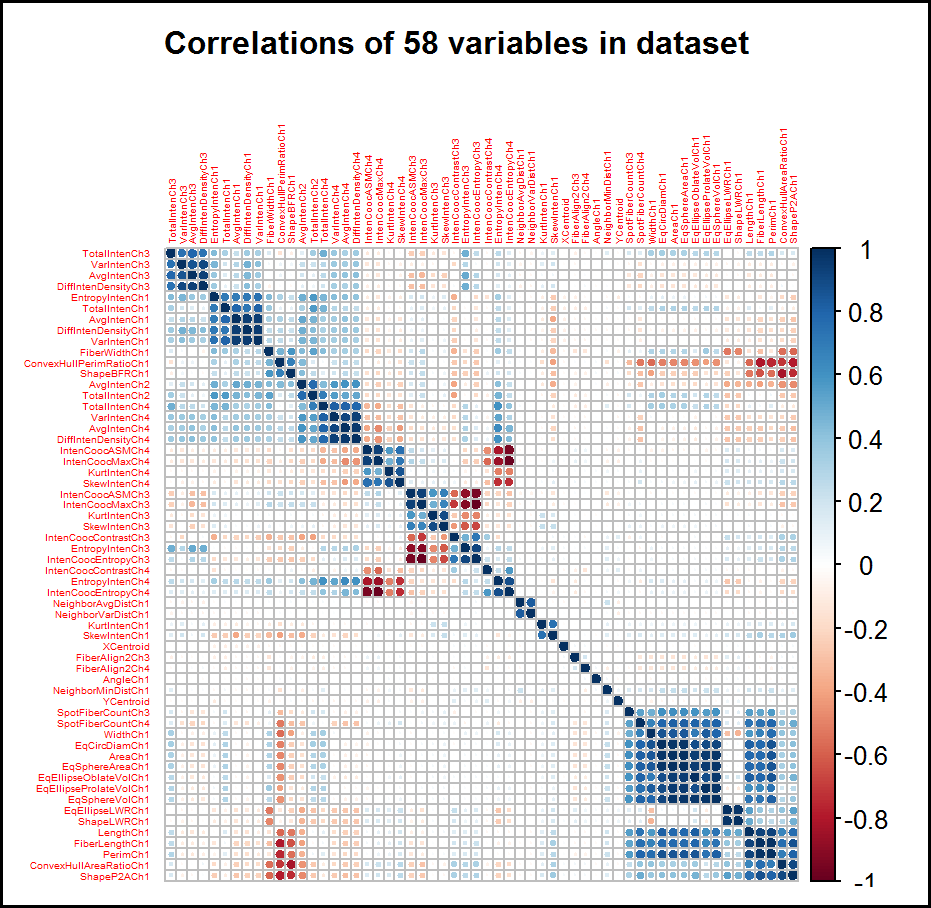


Figure . Correlation plot of variables. Positively correlated variable are in blue and anti-correlated variables are in red. Greater color intensity indicates a stronger correlation. Highly correlated variables should be considered for removal, leaving only one variable.

# Variable Importance Plot

The relative contributions of the variables to the model are plotted for each machine learning method as shown in Figure 2.

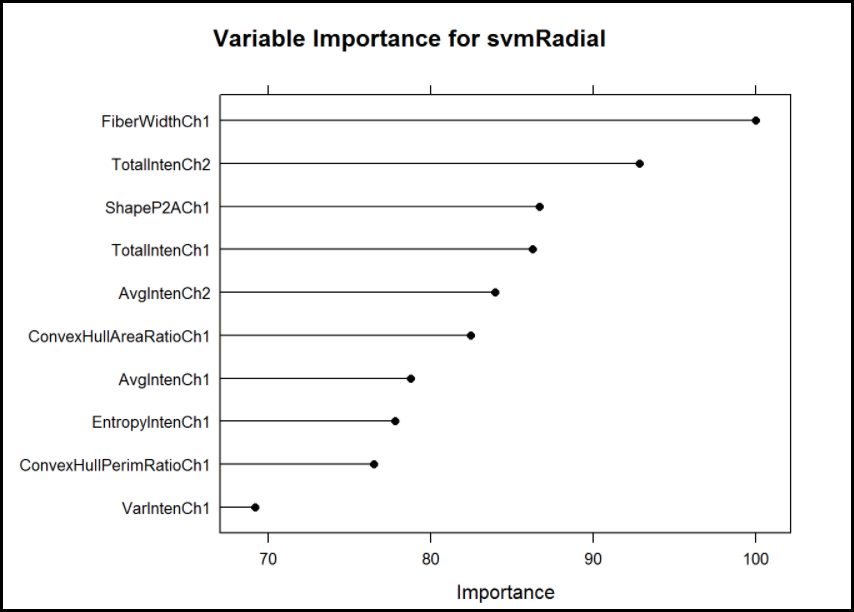
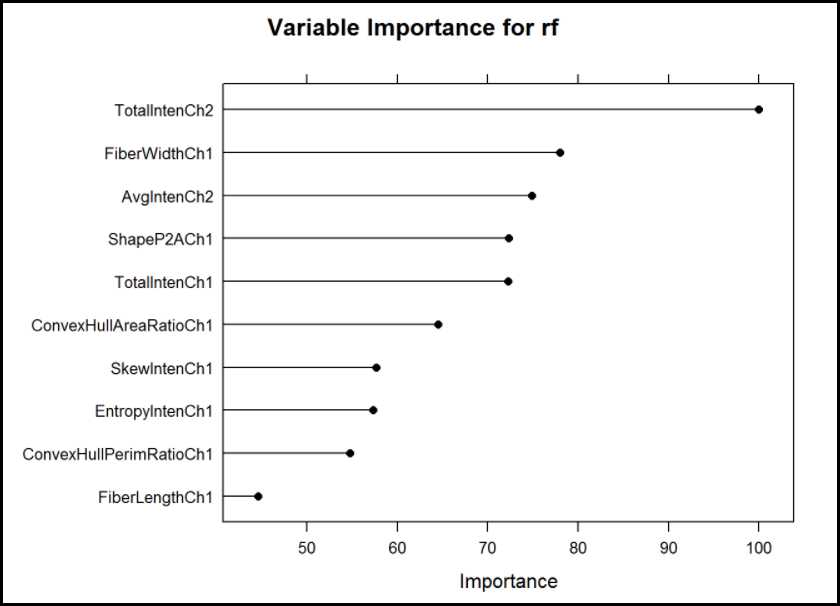
 

Figure . Variable importance for the Support Vector Machine (svmRadial) and Random Forest (rf) models.

# Model Tuning Plot

caret provides automatic variable tuning. A user-defined tuning scheme is also a possibility. The model with the best performance is automatically selected base on the tuning parameters/

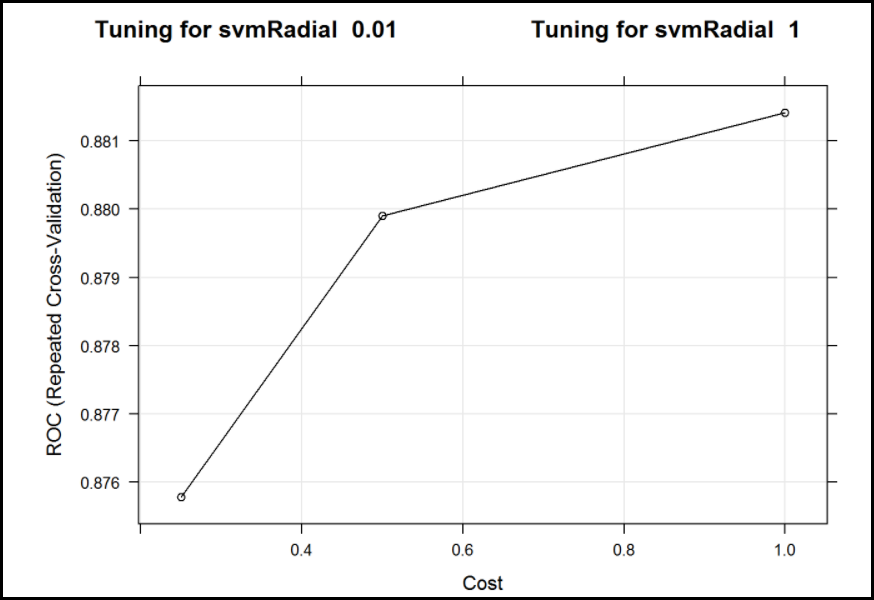
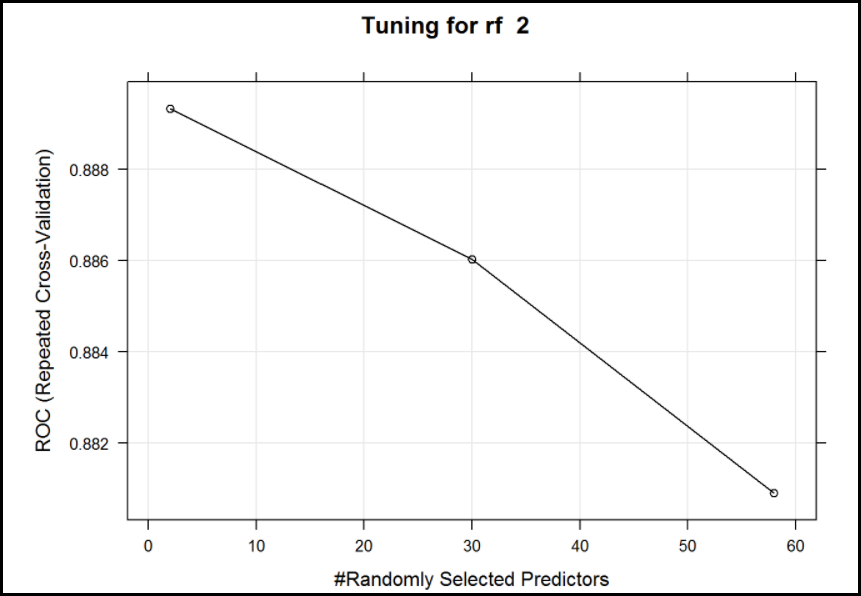
 

Figure . Tuning parameters for Support Vector Machine (svmRadial) and Random Forest (rf) methods.

# Performance Plots For Models

## Classification

The performance of each machine learning method is plotted as the mean (circle) and box-and-whiskers diagram. Receiver Operating Characteristic (ROC), Sensitivity and Specificity are shown. Classification models optimize ROC.

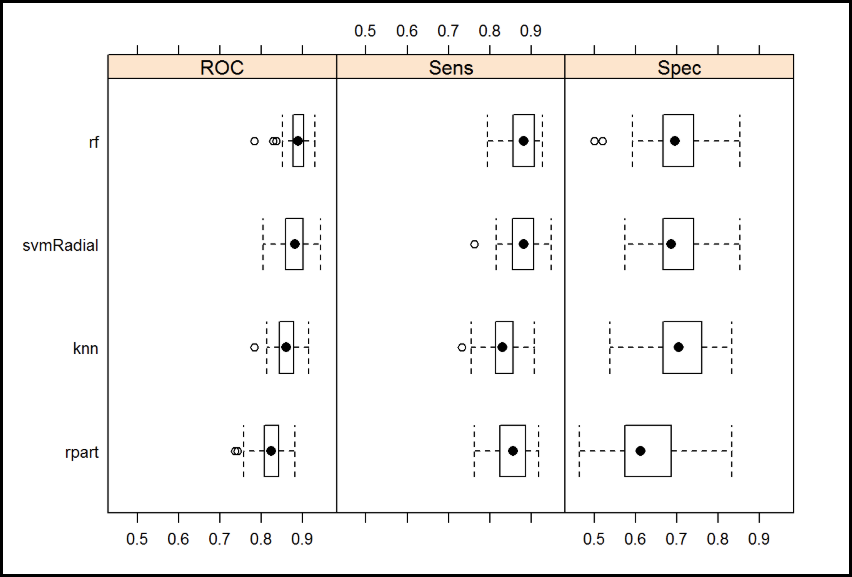


Figure . Model performance for the classification model is plotted for Random Forest (rf), Support Vector Machine (svmRadial), k-Nearest Neighbor (knn) and a CART decision tree (rpart) using (a) ROC (Receiver Operating Characteristic), (b) Sensitivity (Sens) and (c) Specificity (Spec). rpart shows the poorest performance while rf is the best-performing model.

## Confusion Matrix Statistics

A large number of confusion matrix statistics are calculated as shown in Figure 5. Confusion matrix statistics are calculated for each machine learning method. The statistics for Random Forest (rf) are shown.

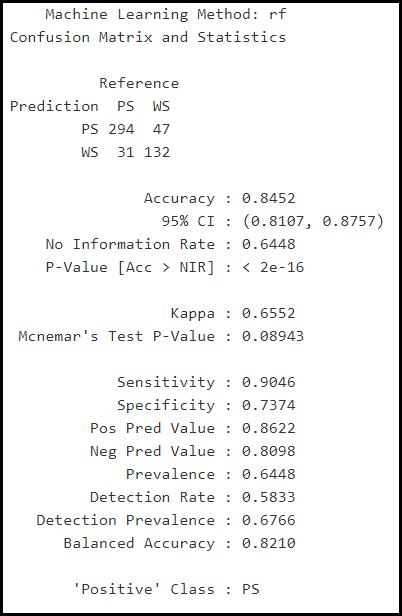


Figure . Confusion matrix statistics are calculated for each machine learning method. The statistics for Random Forest (rf) are shown.

## Regression

The performance of each machine learning method is plotted as the mean (circle) and box-and-whiskers diagram. Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-Squared are shown in Figure 6. Model performance for the regression model is plotted for Random Forest (rf), Support Vector Machine (svmRadial) and Extreme Gradient Boosting (xgbLinear) using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-Squared. svmRadial shows the poorest performance while xbgLinear is the best-performing model.. Regression models optimize RMSE.

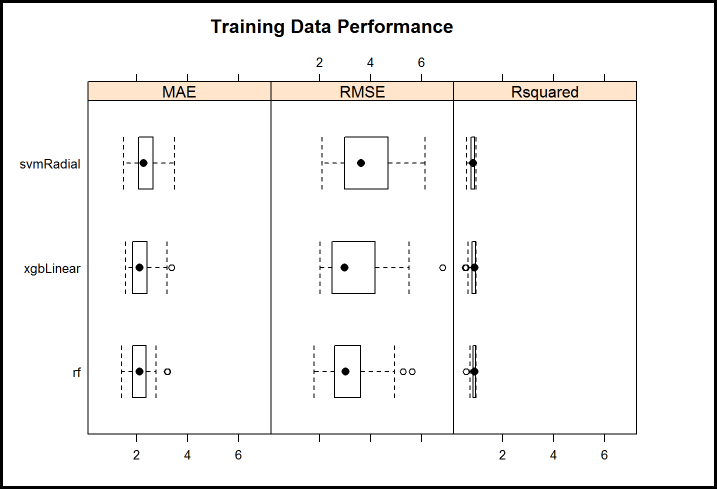


Figure . Model performance for the regression model is plotted for Random Forest (rf), Support Vector Machine (svmRadial) and Extreme Gradient Boosting (xgbLinear) using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-Squared. svmRadial shows the poorest performance while xbgLinear is the best-performing model.

A text summary of model performance is also produced as shown in Figure 7

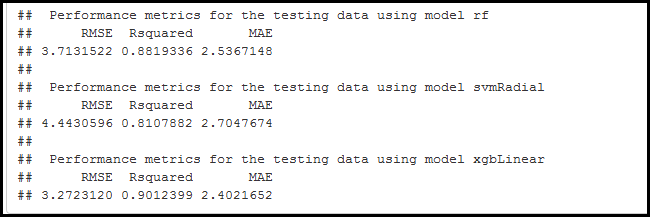


Figure . Model performance for regression models are produced.

# Statistical Comparison Of Models

A statistical comparison of the models is given in Figure 8. Each model is compared pairwise to all other models using the proper performance metrics for the classification or regression models.

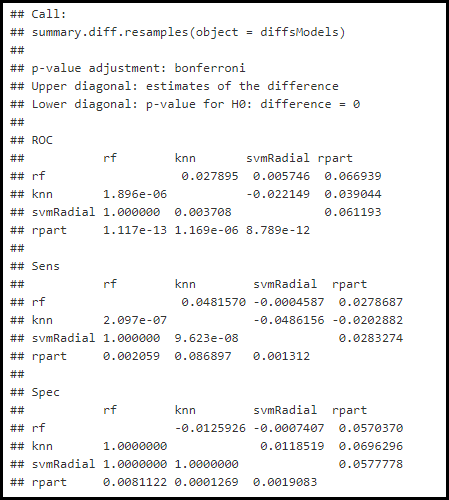


Figure . Statistical comparison between the models. The lower diagonal shows the p-value when comparing each model. For example, the Specificity (Spec) for rf, shows no statistical difference between the knn and svmRadial models (p = 1). However there is a statistical difference between rf and rpart (p = 0.008). This difference is due to the fact that the rpart model performs worse than the rf model.

# R Environment Is Saved

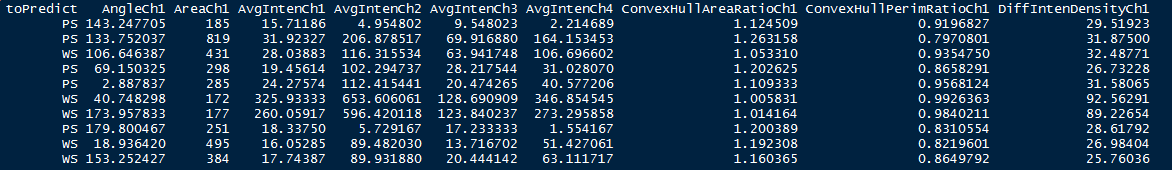
The R environment (RData file) is automatically saved. Model evaluation can take several hours. The results can be examined again by loading the R environment without having to execute the code.

# Datasets In R For Initial Testing

## Classification

The segmentation data uses information from high content imaging microscopy to determine if a cell is "well-segmented". If an individual cell is well-segmented (coded as WS), that means that the data from that cell is acceptable. PS indicates that data from this cell is poorly-segmented and should not be used for further analysis. Figure 9 shows a microscopic image with examples of well-segmented and poorly-segmented cells.

Note that the column name for what we are trying to predict must be named "toPredict".



Note that the the data has very different ranges and should be normalized. The method implemented is to set the mean to 0 and the standard deviation to 1; this is done in the ModelFit() function.

Information on this dataset can be found at: <https://topepo.github.io/caret/data-sets.html> and <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-340>

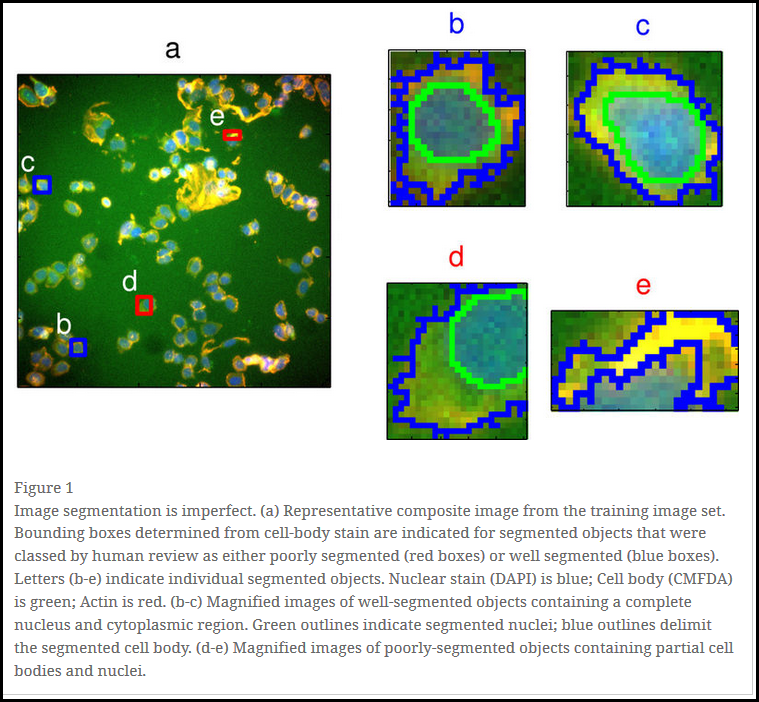


Figure . Microscopy showing well segmented (blue box) and poorly segmented (red box) cells. From <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-340>

## Regression

The objective is to predict the asking price for a house in Boston, the price is medv as shown below.

The BostonHousing data is as follows

We are trying to predict medv (median value of home in $1000's)

Data has very different ranges and should be normalized

crim - per capita crime rate by town

zn - proportion of residential land zoned for lots over 25,000 sq.ft

indus - proportion of non-retail business acres per town

chas - charles river dummy variable (1 if tract bounds river; else 0)

nox - nitric oxides concentration (parts per 10 million)

rm - average number of rooms per dwelling

age - proportion of owner-occupied units built prior to 1940

dis - weighted distances to five boston employment centres

rad - index of accessibility to radial highways

tax - full-value property-tax rate per $10,000

ptratio - pupil-teacher ratio by town

b - 1000(bk - 0.63)^2 where bk is the proportion of blacks by town

lstat - % lower status of the population

**medv - median value of owner-occupied homes in $1000's**

# Miscellaneous

## Caret functions

These are given as caret::function

## Main author-defined functions

### ModelFit()

This is the main function for evaluating the different machine learning methods.

ModelFit <- function(MLtype, dataIn, trControlParams = trn\_ctl, ...) {

fitData <- caret::train(toPredict ~. ,

data = dataIn ,

method = MLtype ,

metric = 'ROC' ,

preProcess = c('center', 'scale') ,

trControl = trControlParams

)

return(fitData)

}

ModelFit arguments:

MLtype - a character string with the models to be evaluated. Each algorithm is called in turn. For example **MLmethods <- c("rf", "knn", "svmRadial", "rpart").** MLmethods is passed to ModelFit() as MLtype.

dataIn - dataframe with one target to predict, which must be a two-level factor and binary (WS/PS, TRUE/FALSE, 0/1, toxic/non-toxic, etc.) for the classification model and a number for the regression model.

trControlParams = trn\_ctl()

trn\_ctl() is described below

ModelFit variables

metric - This selects the optimization type. ROC is optimized for the classification model and

RMSE is optimized for the regression model.

preProcess - center and scale sets variables to mean 0 and standard deviation 1

### trn\_ctl()

trn\_ctl <- caret::trainControl(method = "repeatedcv",

repeats = 5,

number = 10,

classProbs = TRUE, #for confusion matrix

summaryFunction = twoClassSummary # for confusion matrix

)

The function trn\_clt() calls caret::trainControl() which sets many options that control how training models are created.

There are quite a few options for resampling and model tuning.

See: <http://topepo.github.io/caret/model-training-and-tuning.html>

As of Jan 2018, the following methods exist for trainControl():

"boot", "cv", "LOOCV", "LGOCV", "repeatedcv", "timeslice", "none" and "oob".

The function specifies 5-fold cross-validation repeated 10 times.

### MissingValueImpute()

MissingValueImpute <- function(dataIn, classification\_or\_regression, percentDeleteThreshold = 10) {}

percentDeleteThreshold sets the threshold for row and column deletion. The default value is 10 which means that any row or column with > 10% missing values will be deleted. The threshold can be changed by the user. Two functions are called in this function, namely NaDeletePercentRows() and NaDeletePercentColumns().

The method is to first remove rows, remove columns and impute using using the k-Nearest Neighbor (kNN) method in the package DMwR. If there is not enough data for kNN imputation column median imputation is used from the package imputeTS. This occurs in the tryCatch loop as shown below.

impKnn = tryCatch({

DMwR::knnImputation(forImpute)

}, warning = function(cond) {

message('Warning in knn imputation')

message(cond)

}, error = function(cond) {

message('Too many missing values for k Nearest Neighbor imputation, using column median instead')

impKnn <- imputeTS::na.mean(forImpute, option = 'median')

}, finally = {

# nothing

} )

## Required packages and libraries

Packages for each machine learning method must be loaded. You may need to install quite a few packages, but nicely, caret will prompt you to install needed packages during code execution.

library(caret) # the program is constructed using this package

# Machine learning

library(randomForest) # random forest

library(xgboost) # xgbLinear

library(kernlab) # svmRadial

library(kknn) # k nearest neighbor

library(rpart) # CART (decision tree)

library(C50) # C5.0

# imputation

library(DMwR) # knn imputation

library(imputeTS) # median imputation

library(corrplot) # for examination of variable correlations

library(reshape) # for melt

library(dplyr) # for merge

library(mlbench) # for Boston Housing dataset which is used as the default dataset for regression

## Required files

deleteHighlyCorrelated\_<<DATE>>.R

deleteNearZeroVariance\_<<DATE>>.R

GetDefaultDataset\_<<DATE>>.R

MissingValueHandlerImpute\_<<DATE>>.R

**machine\_learning\_pipeline\_ <<DATE>> # MAIN PROGRAM, EXECUTE THIS**

NAs\_Delete\_Percent\_Cols\_<<DATE>>.R

NAs\_Delete\_Percent\_Rows\_<<DATE>>.R

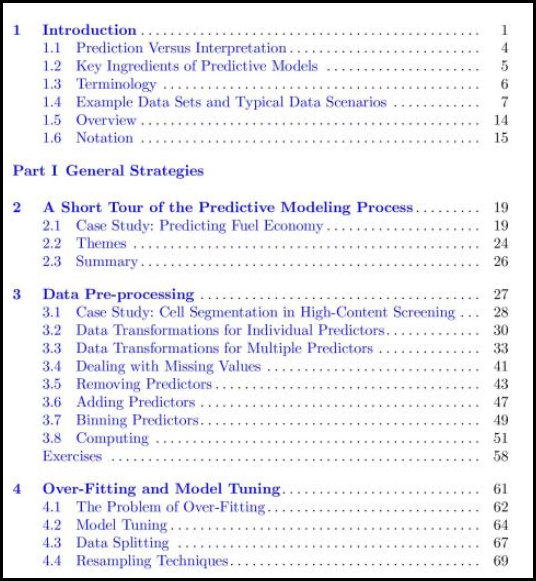
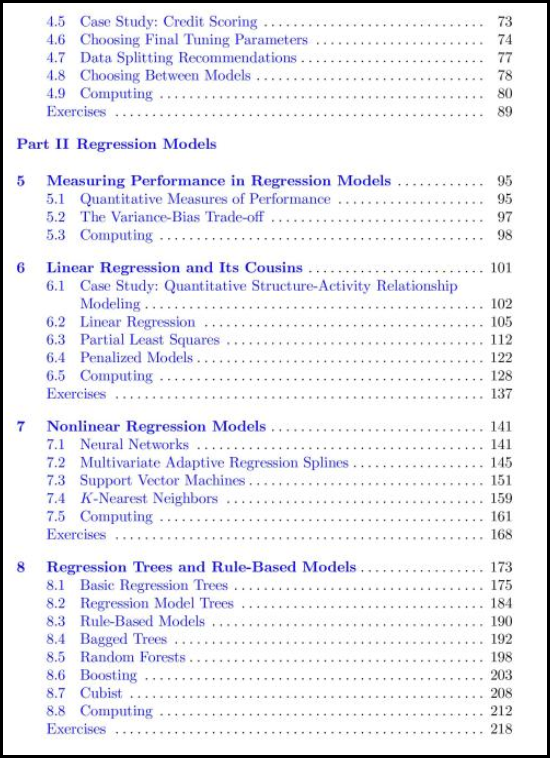
visualization.R

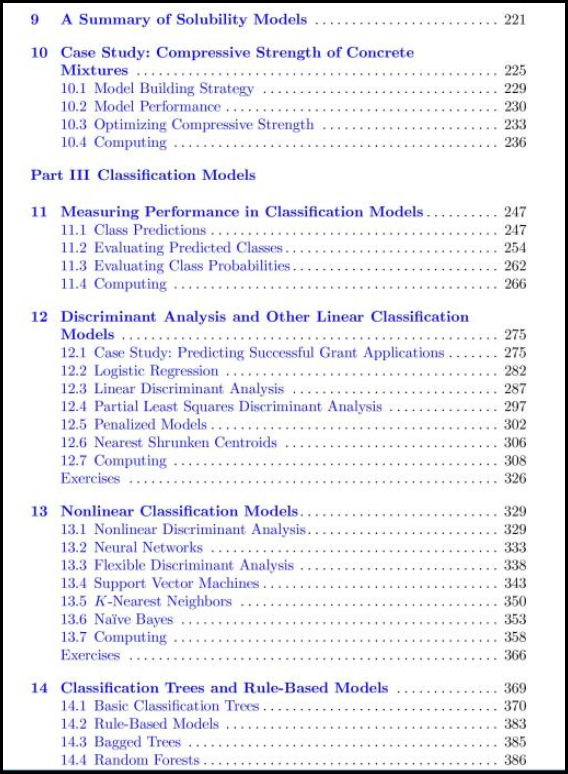
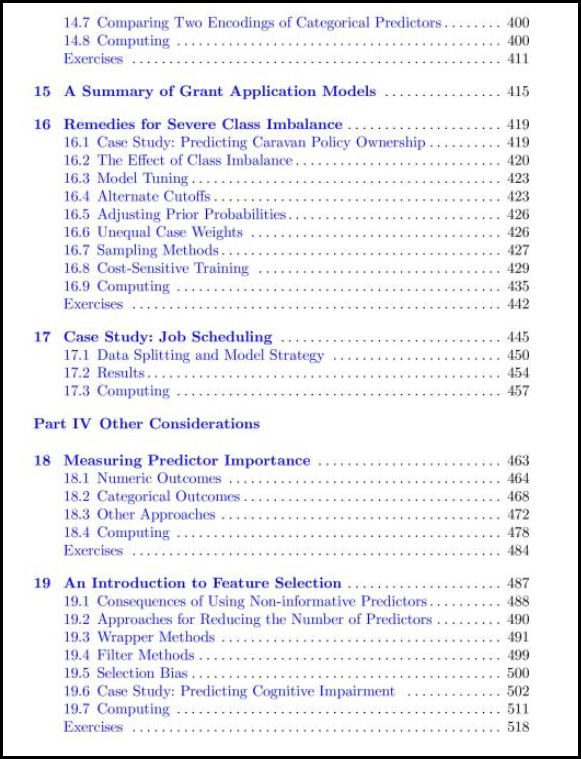
## Table of Contents of The Book "Applied Predictive Modeling"

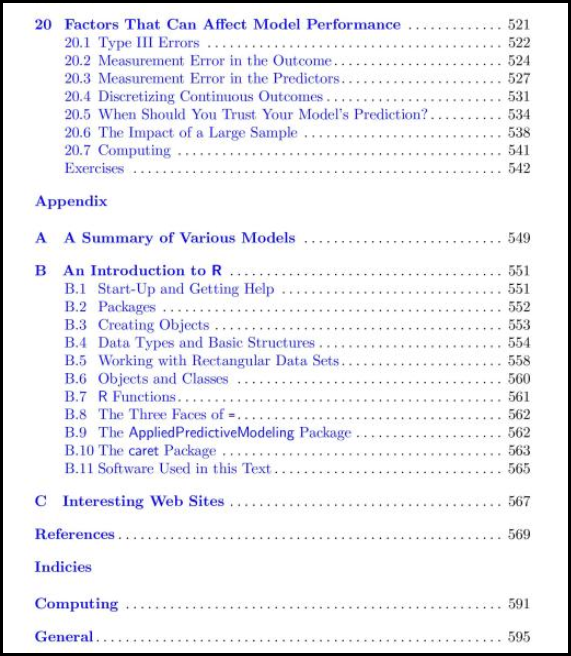
Max Kuhn is the author of the caret package and this book is highly recommended.

The Amazon.com URL is:

<https://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485>



1. <http://scienceblogs.com/mixingmemory/2007/07/01/the-basics-of-statistics-ii-st/> [↑](#footnote-ref-1)