Machine Learning

don’t wait, dive right in

Jeff Jenkins

jeff.jenkins@dataxploits.com

Opportunity

Having coffee with a friend, hearing about customer attrition challenge

Wondered if attrition could be predicted, by customer, using Machine Learning

Found relevant data set on UCI Machine Learning Repository – Telco attrition

Getting Ready

- Evaluate / Prepare data
  - Setting up the question/hypothesis
- Select ML algorithm
  - Supervised (answer included in data) or Unsupervised? (answer not known for training)
  - Continuous response variable? Maybe start with regression (linear, polynomial), random forest
  - Categorical dependent variable – logistic regression, randomforest, svm, clustering

Build Your R Script

Set up any needed libraries:

```
library(caret); library(randomForest)
library(“ggplot2"); library("ggdendro")
```

Point working directory to your data

```
setwd("C:/Users/jeff/Documents/R/machinelearning/churn")
```

Build Your R Script

Read in data as data frame:

```
d <- read.csv("churn_train.csv",
              stringsAsFactors = FALSE)
```

Check variables for type/class (mis-matches contribute to errors), Nas

Prep data – scale, center, pca, et al

Train / Test subsets

Set aside part of the data for testing not to be used for training

Sample R code…probably lifted from Coursera

```
index <- sample(1:nrow(data),
            round(0.75*nrow(data)))
train <- data[index,]
test <- data[-index,]
```

Feature Selection

- Select features (columns) to use for training
  - Start with one feature
  - Or use all features

Maybe start with features that have strongest correlation to the response variable

Train a Model

logistic regression :

```
train.glm <- train(churn_result ~
         international_plan + total_day_charge +
        number_customer_service_calls +
        total_day_minutes,
        data=d, method = "glm")
```

Or **method = "rf"** or **method = "nnet"**

Evaluating training/testing output

- Training results in a classifier saved in your working space. In our examples:
  - train.glm, train.rf, train.nn
- Print the output statistics: train.glm
  - Or str(train.glm) for greater detail
- Output stats will show some variety between different learning algorithms but an Accuracy score will be shown

> train.glm #logistic regression

> Generalized Linear Model 3334 samples

20 predictor 2 classes: ' False', ' True'

No pre-processing Resampling: Bootstrapped (25 reps)

Accuracy	Kappa Accuracy	SD	Kappa SD
0.8547504	0.1686278	0.007549	0.031156

> train.rf

Random Forest 3334 samples 20 predictor 2 classes: ' False', ' True'
No pre-processing Resampling: Bootstrapped (25 reps)...

		Kappa		Kappa
mtry	Accuracy	Accuracy	SD	SD
2	**0.8600**	0.3928	0.0077	0.0245
3	0.8489	0.3705	0.0091	0.0226
4	0.8484	0.3691	0.0090	0.0213

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 mtry = 2.

> train.nn

3334 samples 20 predictor 2 classes: ' False', ' True'

No pre-processing Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3334, 3334,...

Resampling results across tuning parameters:

size	decay	Accuracy	Kappa Accuracy	SD	Kappa SD
1	0.0000	**0.8541**	0.0100	0.0077	0.0525
1	0.0001	0.8550	0.0217	0.0071	0.0761
1	0.1000	0.8541	0.0087	0.0074	0.0302
3	0.0000	0.8540	0.0237	0.0087	0.0827
3	0.0001	0.8551	0.0348	0.0087	0.0982
3	0.1000	0.8608	0.1239	0.0136	0.1792
5	0.0000	0.8569	0.0668	0.0107	0.1286
5	0.0001	0.8544	0.0169	0.0091	0.0602
5	0.1000	**0.8765**	0.3778	0.0105	0.0744

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

The final values used for the model were size = 5 and decay = 0.1.

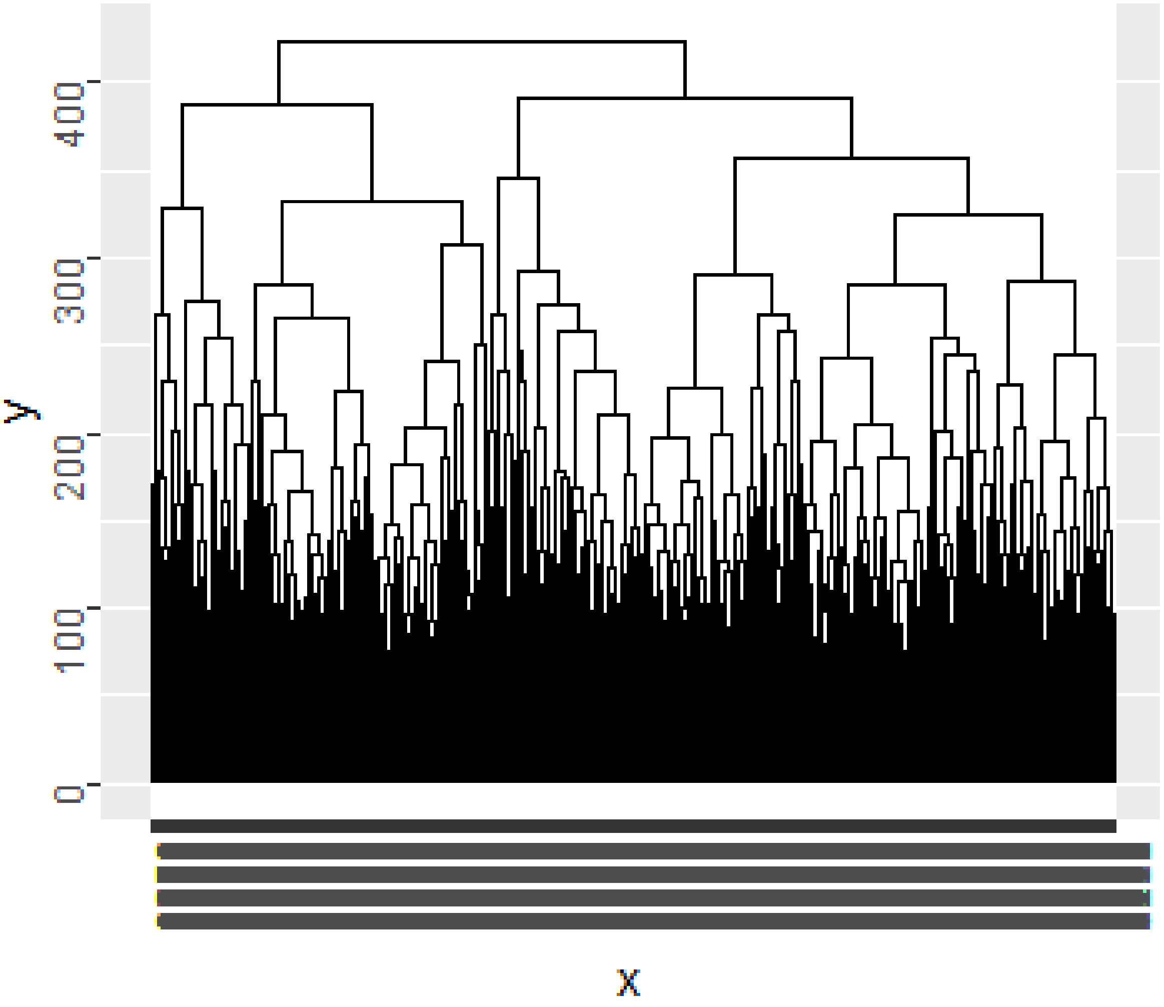
Testing/Evaluating model

- Training on a **test data set** is a measurement of Accuracy on independent data

- **Confusion Matrix** – true positive (precision), false negatives (recall)

- **Optimizing model** – sooo much to say

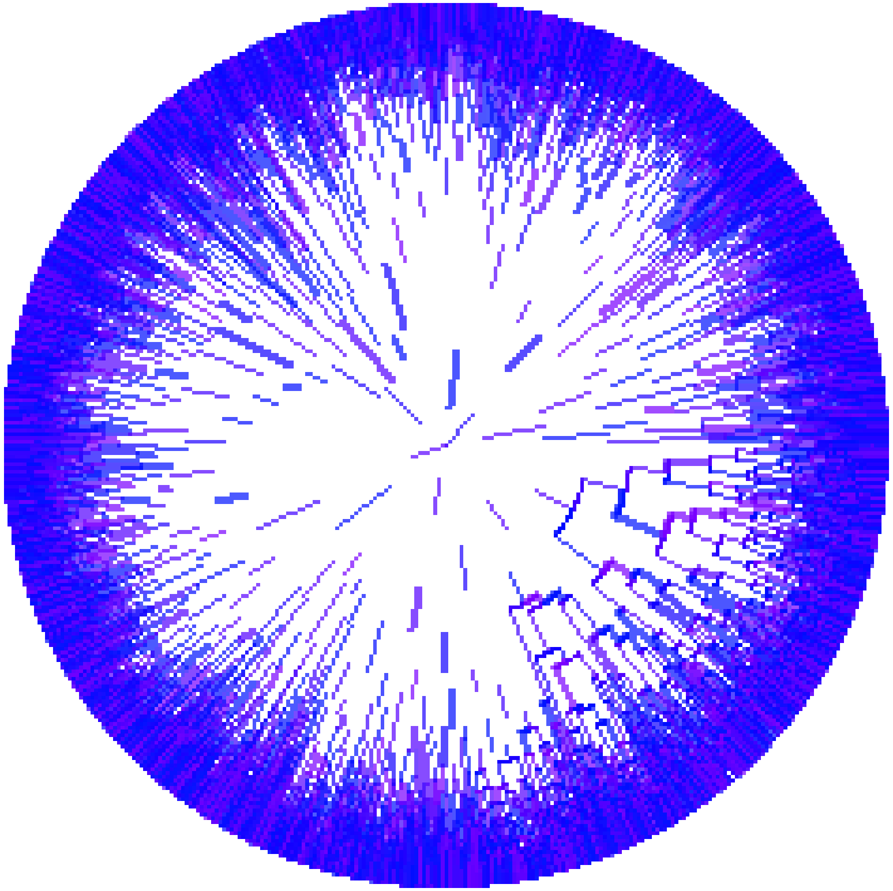
Dendrogram of clustered data



Radial dendrogram

A different look at the branching of categories

Low granularity but indicates groupings



Summary

- Quick & dirty can be a great jumping off point!
- Strong, usable output
  - >=86% Predictive
  - 3 classifiers gave consistent results within a narrow band – reinforcing the accuracy of the output.

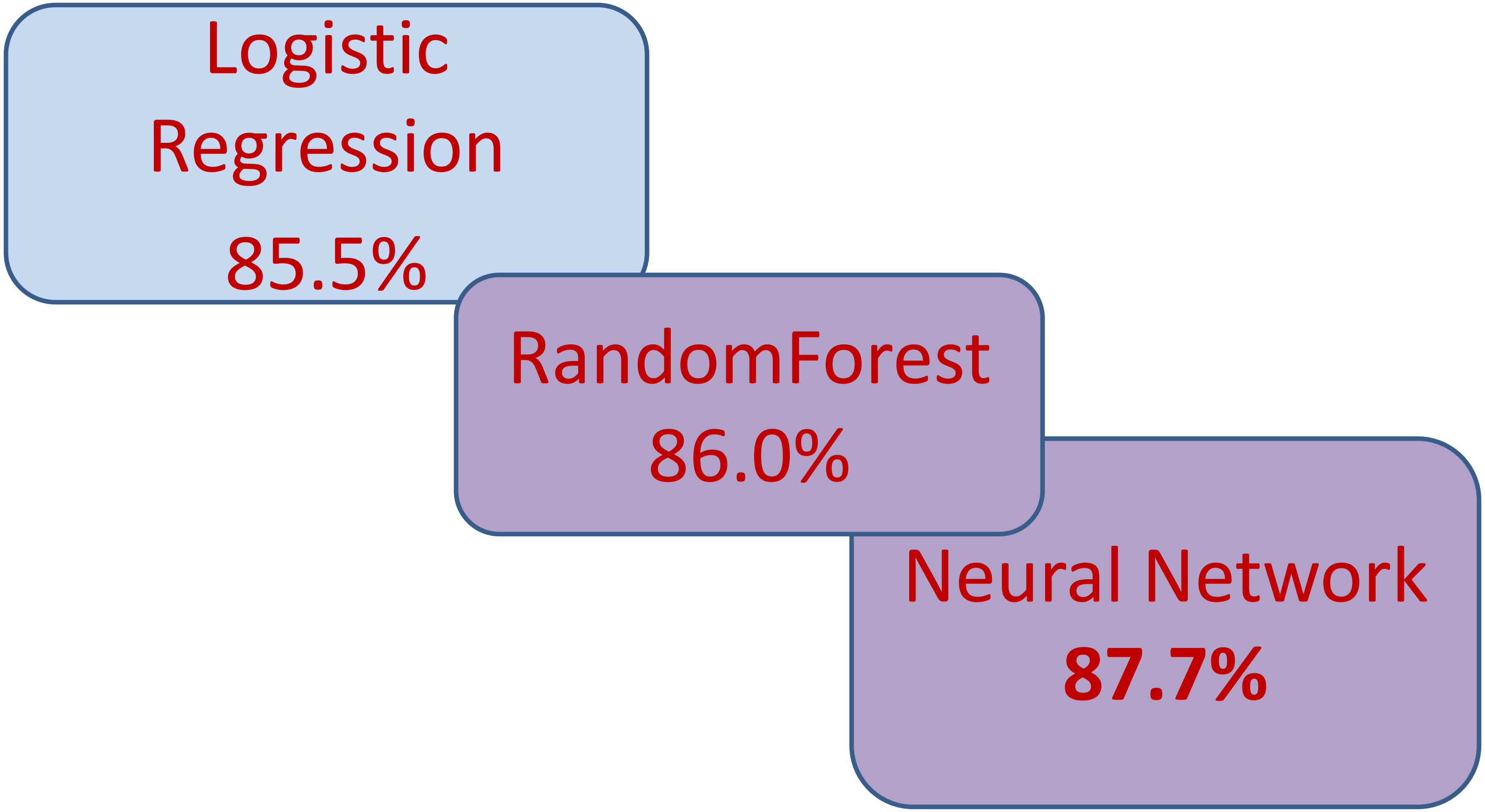
Questions?

jeff.jenkins@dataxploits.com

DataXploits.com links to blog

stone-village.com data blog

Accuracy by model



Logistic Regression

85.5%

RandomForest

86.0%

Neural Network

87.7%