Applied Machine Learning - Getting Started

Max Kuhn (RStudio)

Course Overview

> The session will step through the process of building, visualizing, testing and comparing models that are focused on prediction. The goal of the course is to provide a thorough workflow in R that can be used with many different regression or classification techniques. Case studies are used to illustrate functionality.
>
> *Basic familiarity with R is required.*

The *goal* is for you to be able to easily build predictive/machine learning models in R using a variety of packages and model types.

- "Models that are focused on prediction"... what does that mean?

- "Machine Learning"... so this is deep learning with massive data sets, right?

The course is broken up into sections for *regression* (predicting a numeric outcome) and *classification* (predicting a category).

Why R for Modeling?

- *R has cutting edge models*. Machine learning developers in some domains use R as their primary computing environment and their work often results in R packages.

- *It is easy to port or link to other applications*. R doesn't try to be everything to everyone. If you prefer models implemented in C, C++, `tensorflow`, `keras`, `python`, `stan`, or `Weka`, you can access these applications without leaving R.

- *R and R packages are built by people who* ***do*** *data analysis*.

- *The S language is very mature*.

- The machine learning environment in R is extremely rich.

Downsides to Modeling in R

- R is a data analysis language and is not C or Java. If a high performance deployment is required, R can be treated like a prototyping language.
- R is mostly memory-bound. There are plenty of exceptions to this though.

The main issue is one of *consistency of interface*. For example:

- There are two methods for specifying what terms are in a model[1]. Not all models have both.
- 99% of model functions automatically generate dummy variables.
- Sparse matrices can be used (unless the can't).

[1] There are now three but the last one is brand new and will be discussed later.

Syntax for Computing Predicted Class Probabilities

Function	Package	Code
`lda`	`MASS`	`predict(obj)`
`glm`	`stats`	`predict(obj, type = "response")`
`gbm`	`gbm`	`predict(obj, type = "response", n.trees)`
`mda`	`mda`	`predict(obj, type = "posterior")`
`rpart`	`rpart`	`predict(obj, type = "prob")`
`Weka`	`RWeka`	`predict(obj, type = "probability")`
`logitboost`	`LogitBoost`	`predict(obj, type = "raw", nIter)`

We'll see a solution for this later in the class.

Different Philosophies Used Here

There are two main philosophies to data analysis code that will be discussed in this worksop:

The more *traditional approach* uses high-level syntax and is perhaps the most untidy code that you will encounter.

`caret` is the primary package for untidy predictive modeling:

1. More traditional R coding style.
2. High-level "I'll do that for you" syntax.
3. More comprehensive (for now) and less modular.
4. Contains many optimizations and is easily parallelized.

The *tidy modeling* approach espouses the tenets of the tidyverse:

1. Reuse existing data structures.
2. Compose simple functions with the pipe.
3. Embrace functional programming.
4. Design for humans.

This approach is exemplified by packages such as `modelr`, `broom`, `recipes`, `rsample`, `yardstick`, and `tidyposterior`.

Example Data Set - House Prices

For regression problems, we will use the Ames IA housing data. There are 2,930 properties in the data.

The sale price was recorded along with 81 predictors, including:

- Location (e.g. neighborhood) and lot information.
- House components (garage, fireplace, pool, porch, etc.).
- General assessments such as overall quality and condition.
- Number of bedrooms, baths, and so on.

More details can be found in De Cock (2011, Journal of Statistics Education).

The raw data are at `http://bit.ly/2whgsQM` but we will use a processed version found in the `AmesHousing` package.

Example Data Set - House Prices

Example Data Set - Fuel Economy

The data that are used here are an extended version of the ubiquitous `mtcars` data set.

`fueleconomy.gov` was used to obtain fuel efficiency data on cars from 2015-2018.

Over this time range, duplicate ratings were eliminated; these occur when the same car is sold for several years in a row. As a result, there are 3294 cars that are listed in the data. The predictors include the automaker and addition information about the cars (e.g. intake valves per cycle, aspiration method, etc).

In our analysis, the data from 2015-2107 are used for training to see if we can predict the 609 cars that were new in 2018.

These data are supplied in the GitHub repo.

Example Data Set - Predicting Profession

OkCupid is an online data site that serves international users. Kim and Escobedo-Land (2015, Journal of Statistics Education) describe a data set where over 50,000 profiles from the San Fransisco area were made available by the company.

The data contains several types of fields:

- a number of open text essays related to interests and personal descriptions
- single choice type fields, such as profession, diet, gender, body type, etc.
- multiple choice data, including languages spoken, etc.
- ***no*** usernames or pictures were included.

We will try to predict whether someone has a profession in the STEM fields (science, technology, engineering, and math) using a random sample of the overall dataset.

Tidyverse Syntax

Many tidyverse functions have syntax unlike base R code. For example:

- vectors of variable names are eschewed in favor of *functional programming*. For example:

```r
contains("Sepal")

# instead of

c("Sepal.Width", "Sepal.Length")
```

- The *pipe* operator is preferred. For example

```r
merged <- inner_join(a, b)

# is equal to

merged <- a %>%
  inner_join(b)
```

- Functions are more *modular* than their traditional analogs (`dplyr`'s `filter` and `select` vs `base::subset`)

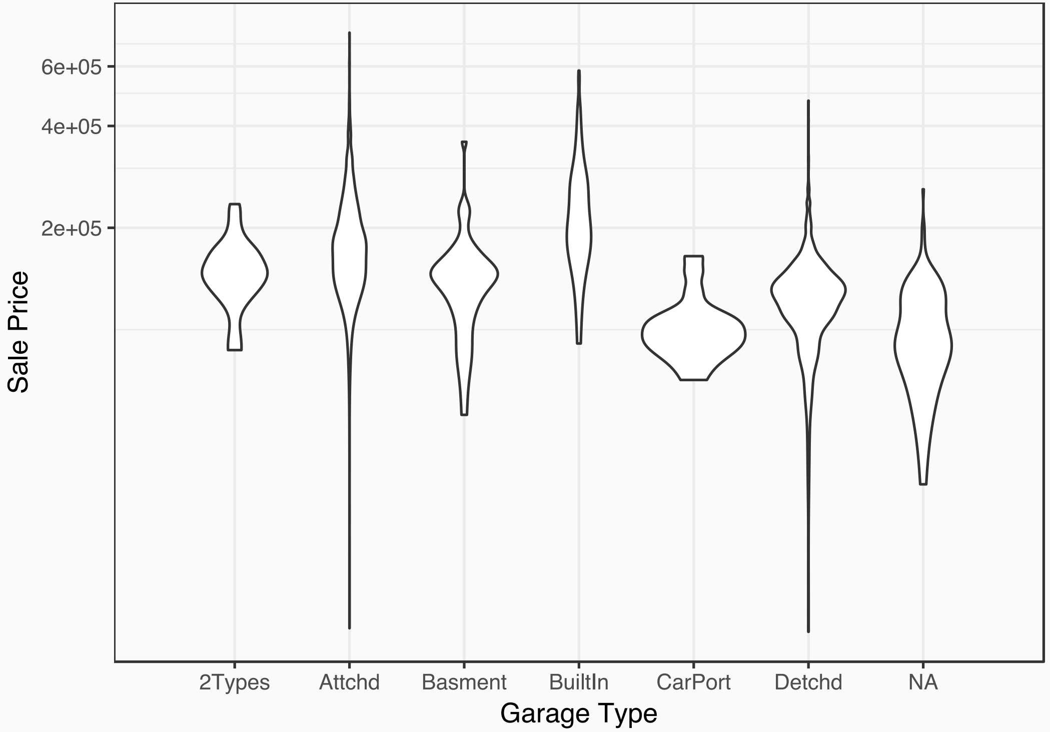
Some Example Data Manipulation Code

```r
library(tidyverse)

ames <-
  read_delim("http://bit.ly/2whgsQM", delim = "\t") %>%
  rename_at(vars(contains(' ')), funs(gsub(' ', '_', .))) %>%
  rename(Sale_Price = SalePrice) %>%
  filter(!is.na(Electrical)) %>%
  select(-Order, -PID, -Garage_Yr_Blt)

ames %>%
  group_by(Alley) %>%
  summarize(mean_price = mean(Sale_Price/1000),
            n = sum(!is.na(Sale_Price)))
```

```
## # A tibble: 3 x 3
##   Alley mean_price     n
##   <chr>      <dbl> <int>
## 1 Grvl         124   120
## 2 Pave         177    78
## 3 <NA>         183  2731
```

Example ggplot2 Code

```r
library(ggplot2)

ggplot(ames,
       aes(x = Garage_Type,
           y = Sale_Price)) +
  geom_violin() +
  coord_trans(y = "log10") +
  xlab("Garage Type") +
  ylab("Sale Price")
```



Examples of purrr::map*

```
library(purrr)

# summarize via purrr::map
by_alley <- split(ames, ames$Alley)
is_list(by_alley)
```

```
## [1] TRUE
```

```
map(by_alley, nrow)
```

```
## $Grvl
## [1] 120
##
## $Pave
## [1] 78
```

```
# or better yet:
map_int(by_alley, nrow)
```

```
## Grvl Pave
##  120   78
```

```
# works on non-list vectors too
ames %>%
  mutate(Sale_Price = Sale_Price %>%
           map_dbl(function(x) x / 1000)) %>%
  select(Sale_Price, Yr_Sold) %>%
  head(4)
```

```
## # A tibble: 4 x 2
##   Sale_Price Yr_Sold
##        <dbl>   <int>
## 1        215    2010
## 2        105    2010
## 3        172    2010
## 4        244    2010
```

Quick Data Investigation

To get warmed up, let's load the Ames data and do some basic investigations into the variables, such as exploratory visualizations or summary statistics. The idea is to get a feel for the data.

Let's take 10 minutes to work on your own or with someone next to you. Collaboration is highly encouraged!

To get the data:

```
library(AmesHousing)
ames <- make_ames()
```

Where We Go From Here

Part 2 Basic Principals

- Data Splitting, Models in R, Resampling, Tuning (`rsample`)

Part 3 Feature Engineering and Preprocessing

- Data treatments (`recipes`)

Part 4 Regression Modeling

- Measuring Performance, penalized regression, multivariate adaptive regression splines (MARS), ensembles (`yardstick` , `recipes` , `caret` , `earth` , `glmnet` , `tidyposterior` , `doParallel`)

Part 5 Classification Modeling

- Measuring Performance, trees, ensembles, naive Bayes (`yardstick` , `recipes` , `caret` , `rpart` , `klaR` , `tidyposterior`)

Resources

- `http://www.tidyverse.org/`
- R for Data Science
- Jenny's `purrr` tutorial or Happy R Users Purrr
- Programming with `dplyr` vignette
- Selva Prabhakaran's `ggplot2` tutorial
- `caret` package documentation
- CRAN Machine Learning Task View

About these slides.... they were created with Yihui's `xaringan` and the stylings are a slightly modified version of Patrick Schratz's Metropolis theme.