Week 7 - Data Science II

Phileas Dazeley-Gaist

15/02/2022

```r
# Applied Data Science II - Week 7
# # ---------------------------------------------------------------

# Today we are going to talk about TREES!
#
#
# # ---------------------------------------------------------------
#
# Load your libraries!
#
# # ---------------------------------------------------------------

library(ISLR2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(nnet)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.
```

```
##
## Attaching package: 'mgcv'
```

```
## The following object is masked from 'package:nnet':
##
##     multinom
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
# do these not work? Then you'll have to install them!

# You  might need some new commands to install this!
# if you can just run install.packages(c("randomForest","xgboost","janitor")), try this:
#
# urlPackage <- "https://cran.r-project.org/src/contrib/Archive/randomForest/randomForest_4.6-
# install.packages(urlPackage, repos=NULL)
#
# if this doesn't work - you need to download the newest version of R
# and then update RStudio.
```

```
# # ---------------------------------------------------------------
#
# Lets build a Random Forest with our Spotify Data!
#
# # ---------------------------------------------------------------

# go ahead and grab that spotify data again.

spotify_data <- read_csv("w7 data/spotify_labels.csv")
```

```
## Rows: 13795 Columns: 15
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr  (2): label, artist_name
## dbl (13): danceability, energy, key, loudness, mode, speechiness, acousticne...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# clean it up with this procedure:

cleaned_spotify <- spotify_data %>%
        select(-artist_name,key) %>%
        mutate(mode = as.factor(mode),
               key = as.factor(key),
               time_signature = as.factor(time_signature),
               label = as.factor(label))


# let's build a test and training split of 80%/20%

indx <- createDataPartition(cleaned_spotify$label, p = 0.75, list = FALSE)
train <- cleaned_spotify[indx,]
test <- cleaned_spotify[-indx,]

# alrighty - go ahead and build yourself a tree!
# sit back, relax, and grab a cup of coffee...
random_forest_model <- train(label ~ .,
                             data = train,
                             method='rf',
                             metric='Accuracy',
                             trControl = trainControl(method = 'cv', number = 10))

# and now we wait!

random_forest_model
```

```
## Random Forest
##
## 10348 samples
##    13 predictor
##     4 classes: 'hiphop', 'indie', 'metal', 'pop'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9313, 9313, 9313, 9313, 9314, 9312, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
```

```
##    2    0.7347310  0.6404999
##   13    0.7413028  0.6502484
##   25    0.7359881  0.6432093
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 13.
```

```r
# let's assess the model accuracy!

random_forest_pred <- predict(random_forest_model, test)
confusionMatrix(random_forest_pred, test$label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction hiphop indie metal pop
##     hiphop    506    32     4  89
##     indie      49   792   106 188
##     metal       2   108   641  13
##     pop        88   152    10 667
##
## Overall Statistics
##
##                Accuracy : 0.756
##                  95% CI : (0.7413, 0.7703)
##     No Information Rate : 0.3145
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.67
##
##  Mcnemar's Test P-Value : 0.2062
##
## Statistics by Class:
##
##                      Class: hiphop Class: indie Class: metal Class: pop
## Sensitivity                 0.7845       0.7306       0.8423     0.6970
## Specificity                 0.9554       0.8548       0.9542     0.8996
## Pos Pred Value              0.8019       0.6978       0.8390     0.7274
## Neg Pred Value              0.9506       0.8737       0.9553     0.8854
## Prevalence                  0.1871       0.3145       0.2208     0.2776
## Detection Rate              0.1468       0.2298       0.1860     0.1935
## Detection Prevalence        0.1831       0.3293       0.2216     0.2660
## Balanced Accuracy           0.8699       0.7927       0.8983     0.7983
```

```r
# let's quickly compare that to the logistic model ....

multi_class_logit <- nnet::multinom(label ~ ., data = train)
```

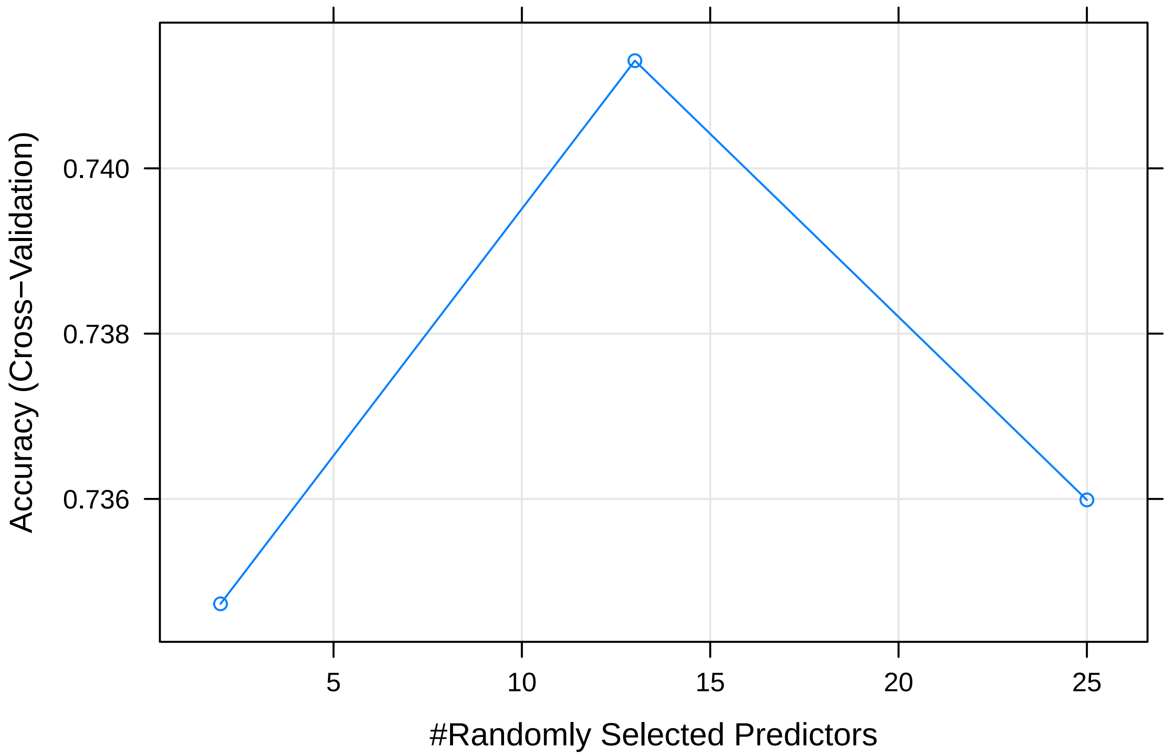
```
## # weights:  108 (78 variable)
## initial  value 14345.374049
## iter  10 value 13069.351070
## iter  20 value 9686.194357
## iter  30 value 8828.460211
## iter  40 value 8614.931670
## iter  50 value 8454.901993
## iter  60 value 8369.639079
## iter  70 value 8352.876210
## iter  80 value 8350.471812
## final  value 8350.134999
## converged
```

```r
logit_pred <- predict(multi_class_logit, test)
confusionMatrix(logit_pred, test$label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction hiphop indie metal pop
##     hiphop    442    54     1  81
##     indie      59   590    97 206
##     metal      10   186   649  26
##     pop       134   254    14 644
##
## Overall Statistics
##
##                Accuracy : 0.6745
##                  95% CI : (0.6586, 0.6901)
##     No Information Rate : 0.3145
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5612
##
##  Mcnemar's Test P-Value : 1.627e-10
##
## Statistics by Class:
##
##                      Class: hiphop Class: indie Class: metal Class: pop
## Sensitivity                 0.6853       0.5443       0.8528     0.6729
## Specificity                 0.9515       0.8468       0.9173     0.8386
## Pos Pred Value              0.7647       0.6197       0.7451     0.6157
## Neg Pred Value              0.9292       0.8020       0.9565     0.8696
## Prevalence                  0.1871       0.3145       0.2208     0.2776
## Detection Rate              0.1282       0.1712       0.1883     0.1868
## Detection Prevalence        0.1677       0.2762       0.2527     0.3035
## Balanced Accuracy           0.8184       0.6955       0.8851     0.7557
```

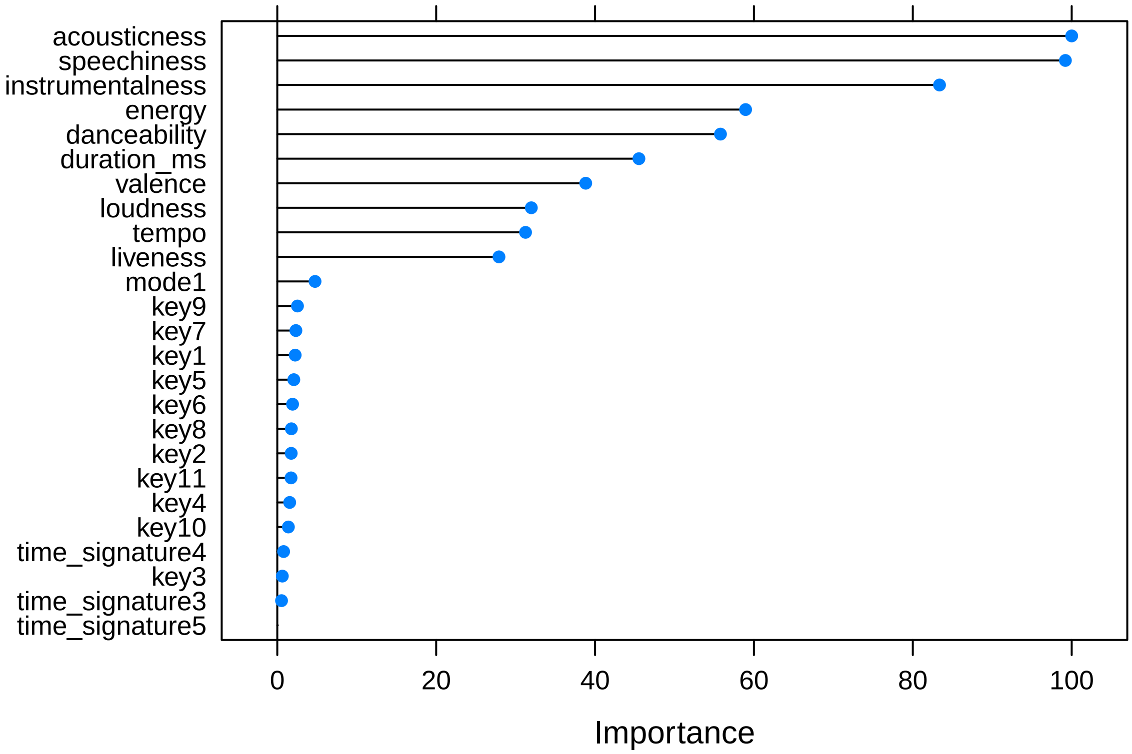
```r
# looks like randomForest is better out of the box! let's visualize what's going on.

plot(random_forest_model)
```




```r
# this gives you an idea of how your training accuracy changes over a randomly selected number
# maybe you want to know which features matter most?

plot(varImp(random_forest_model, scale = TRUE))
```




```
# ahhh, interesting! we see which variables mattered most here.

# # ---------------------------------------------------------------
#
# Stop! Go back to the presentation
#
# # ---------------------------------------------------------------
```

```
# # ---------------------------------------------------------------
#
# Lets build an XGBoost model!
#
# # ---------------------------------------------------------------

# we're going to revisit the walmart data from last week!
# load it in from the Google Drive and prepare it as below:

walmart <- read_csv("w7 data/walmart.csv")
```

```
## Rows: 6435 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): date
## dbl (7): store, weekly_sales, holiday_flag, temperature, fuel_price, cpi, un...
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
walmart_cleaned <- walmart %>%
  mutate(store = as.factor(store),
         holiday_flag = as.factor(holiday_flag),
         year = as.factor(year(dmy(date))),
         month = as.factor(month(date))) %>%
  select(-c(date))


index <- createDataPartition(walmart_cleaned$weekly_sales, p = .8, list=FALSE)
training_data <- walmart_cleaned[ index,]
test_data  <- walmart_cleaned[-index,]


# now, let's build an xgboost model!
# XGBoost requires your data to be entirely numerical, so let's convert it!

X_prep_train <- training_data %>% dplyr::select(-weekly_sales)
X_prep_train
```

```
## # A tibble: 5,151 x 8
##    store holiday_flag temperature fuel_price   cpi unemployment year  month
##    <fct> <fct>              <dbl>      <dbl> <dbl>        <dbl> <fct> <fct>
##  1 1     0                   42.3       2.57  211.         8.11 2010  2
##  2 1     1                   38.5       2.55  211.         8.11 2010  2
##  3 1     0                   39.9       2.51  211.         8.11 2010  2
##  4 1     0                   46.6       2.56  211.         8.11 2010  2
##  5 1     0                   46.5       2.62  211.         8.11 2010  3
##  6 1     0                   57.8       2.67  211.         8.11 2010  3
##  7 1     0                   54.6       2.72  211.         8.11 2010  3
##  8 1     0                   51.4       2.73  211.         8.11 2010  3
##  9 1     0                   62.3       2.72  211.         7.81 2010  4
## 10 1     0                   65.9       2.77  211.         7.81 2010  4
## # ... with 5,141 more rows
```

```
X_prep_test <- test_data %>% dplyr::select(-weekly_sales)
X_train = model.matrix(~.-1, data = X_prep_train)
y_train = training_data$weekly_sales
X_test = model.matrix(~.-1, data = X_prep_test)
y_test = test_data$weekly_sales


# here we go!

xgboost_model <- train(
```

```
  x = X_train,
  y = y_train,
  method = "xgbTree",
  trControl = trainControl(method = 'cv', number = 10),
  verbosity = 0
)

# let's peek...

xgboost_model
```

```
## eXtreme Gradient Boosting
##
## 5151 samples
##   63 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4636, 4637, 4636, 4636, 4636, 4636, ...
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  RMSE      Rsquared
##   0.3  1          0.6               0.50        50      318881.3  0.8050927
##   0.3  1          0.6               0.50       100      223265.9  0.8995407
##   0.3  1          0.6               0.50       150      176634.0  0.9236105
##   0.3  1          0.6               0.75        50      318378.6  0.8160278
##   0.3  1          0.6               0.75       100      224411.9  0.8993174
##   0.3  1          0.6               0.75       150      178470.9  0.9235830
##   0.3  1          0.6               1.00        50      319874.1  0.8123191
##   0.3  1          0.6               1.00       100      225121.4  0.8983366
##   0.3  1          0.6               1.00       150      180615.4  0.9230187
##   0.3  1          0.8               0.50        50      316989.8  0.8122990
##   0.3  1          0.8               0.50       100      222860.7  0.8990883
##   0.3  1          0.8               0.50       150      175779.6  0.9234883
##   0.3  1          0.8               0.75        50      318188.5  0.8176886
##   0.3  1          0.8               0.75       100      223845.6  0.8985721
##   0.3  1          0.8               0.75       150      178488.1  0.9227743
##   0.3  1          0.8               1.00        50      319440.1  0.8115130
##   0.3  1          0.8               1.00       100      225031.7  0.8985178
##   0.3  1          0.8               1.00       150      180918.8  0.9227883
##   0.3  2          0.6               0.50        50      222087.2  0.9018215
##   0.3  2          0.6               0.50       100      147143.6  0.9398826
##   0.3  2          0.6               0.50       150      127754.2  0.9494978
##   0.3  2          0.6               0.75        50      220518.1  0.9069208
##   0.3  2          0.6               0.75       100      147035.2  0.9416809
##   0.3  2          0.6               0.75       150      126426.6  0.9510126
##   0.3  2          0.6               1.00        50      221070.4  0.9036285
```

```
##    0.3  2           0.6               1.00       100       149903.2  0.9394423
##    0.3  2           0.6               1.00       150       128056.4  0.9506266
##    0.3  2           0.8               0.50        50       219019.2  0.9065215
##    0.3  2           0.8               0.50       100       144217.0  0.9431617
##    0.3  2           0.8               0.50       150       124642.7  0.9518310
##    0.3  2           0.8               0.75        50       219868.3  0.9042241
##    0.3  2           0.8               0.75       100       145563.2  0.9429692
##    0.3  2           0.8               0.75       150       123503.6  0.9534820
##    0.3  2           0.8               1.00        50       220167.5  0.9047931
##    0.3  2           0.8               1.00       100       146256.8  0.9430762
##    0.3  2           0.8               1.00       150       125540.9  0.9523687
##    0.3  3           0.6               0.50        50       171429.9  0.9296519
##    0.3  3           0.6               0.50       100       126171.6  0.9506833
##    0.3  3           0.6               0.50       150       119052.3  0.9545909
##    0.3  3           0.6               0.75        50       170522.4  0.9329632
##    0.3  3           0.6               0.75       100       123754.0  0.9530765
##    0.3  3           0.6               0.75       150       115265.9  0.9576158
##    0.3  3           0.6               1.00        50       170714.5  0.9334600
##    0.3  3           0.6               1.00       100       123935.1  0.9534526
##    0.3  3           0.6               1.00       150       114744.8  0.9582383
##    0.3  3           0.8               0.50        50       168865.6  0.9314196
##    0.3  3           0.8               0.50       100       124627.0  0.9517360
##    0.3  3           0.8               0.50       150       117356.0  0.9558311
##    0.3  3           0.8               0.75        50       168167.8  0.9341664
##    0.3  3           0.8               0.75       100       124191.3  0.9523129
##    0.3  3           0.8               0.75       150       116234.1  0.9566464
##    0.3  3           0.8               1.00        50       168459.8  0.9351550
##    0.3  3           0.8               1.00       100       123093.0  0.9539122
##    0.3  3           0.8               1.00       150       114505.1  0.9582478
##    0.4  1           0.6               0.50        50       270681.2  0.8615349
##    0.4  1           0.6               0.50       100       182085.3  0.9218931
##    0.4  1           0.6               0.50       150       152484.3  0.9323471
##    0.4  1           0.6               0.75        50       271661.3  0.8584131
##    0.4  1           0.6               0.75       100       183369.4  0.9221703
##    0.4  1           0.6               0.75       150       153930.0  0.9321059
##    0.4  1           0.6               1.00        50       271777.6  0.8602084
##    0.4  1           0.6               1.00       100       185247.3  0.9225700
##    0.4  1           0.6               1.00       150       155695.8  0.9313455
##    0.4  1           0.8               0.50        50       270577.5  0.8607442
##    0.4  1           0.8               0.50       100       182113.6  0.9207918
##    0.4  1           0.8               0.50       150       152180.7  0.9325891
##    0.4  1           0.8               0.75        50       272094.8  0.8582196
##    0.4  1           0.8               0.75       100       183093.2  0.9215436
##    0.4  1           0.8               0.75       150       153777.2  0.9320624
##    0.4  1           0.8               1.00        50       272722.3  0.8596474
##    0.4  1           0.8               1.00       100       185106.3  0.9222648
##    0.4  1           0.8               1.00       150       155522.3  0.9314878
##    0.4  2           0.6               0.50        50       178385.4  0.9260528
```

```
##   0.4  2          0.6                0.50       100       128953.8  0.9489140
##   0.4  2          0.6                0.50       150       119642.0  0.9541412
##   0.4  2          0.6                0.75        50       178359.1  0.9286606
##   0.4  2          0.6                0.75       100       131141.3  0.9477631
##   0.4  2          0.6                0.75       150       119294.0  0.9547186
##   0.4  2          0.6                1.00        50       177883.5  0.9316482
##   0.4  2          0.6                1.00       100       129523.4  0.9498700
##   0.4  2          0.6                1.00       150       117768.4  0.9563566
##   0.4  2          0.8                0.50        50       177180.1  0.9268143
##   0.4  2          0.8                0.50       100       128173.7  0.9493682
##   0.4  2          0.8                0.50       150       118576.0  0.9550723
##   0.4  2          0.8                0.75        50       178210.0  0.9288764
##   0.4  2          0.8                0.75       100       127022.4  0.9513305
##   0.4  2          0.8                0.75       150       119233.1  0.9547033
##   0.4  2          0.8                1.00        50       178241.0  0.9298022
##   0.4  2          0.8                1.00       100       129590.1  0.9496206
##   0.4  2          0.8                1.00       150       118026.9  0.9560186
##   0.4  3          0.6                0.50        50       146067.5  0.9387714
##   0.4  3          0.6                0.50       100       121872.8  0.9525913
##   0.4  3          0.6                0.50       150       117602.3  0.9556247
##   0.4  3          0.6                0.75        50       147933.5  0.9372930
##   0.4  3          0.6                0.75       100       120994.4  0.9531608
##   0.4  3          0.6                0.75       150       115444.1  0.9572067
##   0.4  3          0.6                1.00        50       144822.7  0.9412783
##   0.4  3          0.6                1.00       100       118524.5  0.9554319
##   0.4  3          0.6                1.00       150       113830.4  0.9584668
##   0.4  3          0.8                0.50        50       141722.1  0.9426858
##   0.4  3          0.8                0.50       100       117862.6  0.9554089
##   0.4  3          0.8                0.50       150       115657.3  0.9569253
##   0.4  3          0.8                0.75        50       141750.8  0.9441617
##   0.4  3          0.8                0.75       100       118264.5  0.9553926
##   0.4  3          0.8                0.75       150       113208.4  0.9587947
##   0.4  3          0.8                1.00        50       139535.5  0.9465129
##   0.4  3          0.8                1.00       100       117685.6  0.9560690
##   0.4  3          0.8                1.00       150       112278.1  0.9595462
##   MAE
##   270804.75
##   176949.13
##   127755.69
##   270130.38
##   177692.85
##   129785.92
##   271430.61
##   178014.21
##   131392.26
##   268979.06
##   176649.66
##   127581.81
```

```
## 270400.08
## 176909.39
## 129271.81
## 271137.17
## 177926.60
## 131633.62
## 176483.61
##  99560.09
##  77991.30
## 175715.59
##  99914.26
##  77836.71
## 175342.65
## 101018.27
##  78728.17
## 174593.73
##  97786.48
##  76685.83
## 175241.08
##  99177.25
##  76643.98
## 174967.43
##  99527.14
##  77092.62
## 125038.91
##  76934.98
##  70395.63
## 124430.13
##  75671.69
##  67660.16
## 124944.15
##  75702.23
##  67559.42
## 123503.83
##  76376.28
##  69333.82
## 123754.83
##  75121.99
##  67624.94
## 123676.73
##  75261.82
##  67140.21
## 223582.46
## 134028.00
## 100436.13
## 224507.68
## 134550.36
## 101460.16
```

```
## 224650.85
## 135967.57
## 102865.97
## 223890.46
## 134059.91
##  99597.16
## 224859.53
## 134935.58
## 101409.92
## 225304.14
## 135783.55
## 102750.28
## 132881.92
##  80984.73
##  73030.28
## 132223.71
##  81477.78
##  71410.92
## 132513.75
##  80842.47
##  70720.62
## 132197.12
##  79939.41
##  71368.37
## 132417.37
##  79388.50
##  71168.81
## 132148.66
##  80859.38
##  70111.03
##  96324.10
##  71660.24
##  69322.66
##  96689.23
##  70071.46
##  67016.16
##  95505.72
##  69828.37
##  66731.32
##  93584.62
##  70193.16
##  69577.81
##  93901.30
##  68703.60
##  65933.74
##  92837.02
##  68763.25
##  65525.75
```

```
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 150, max_depth = 3, eta
##  = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 1.
```

```
# let's assess the model fit with RMSE...

predicted = predict(xgboost_model, X_test)
residuals = y_test - predicted
(RMSE = sqrt(mean(residuals^2)))
```

```
## [1] 135753.8
```

```
# and we can even manually calculate an R2!

y_test_mean = mean(y_test)
# Calculate total sum of squares
tss =  sum((y_test - y_test_mean)^2 )
# Calculate residual sum of squares
rss =  sum(residuals^2)
# Calculate R-squared
(rsq  =  1 - (rss/tss))
```

```
## [1] 0.9470198
```

```
# let's compare this to what we did last week...

gams_model <- gam(weekly_sales ~ store + holiday_flag +
                        s(temperature) +
                        s(fuel_price) + s(cpi) + s(unemployment) +
                        year + month, data = training_data)

predictions_gam <- predict(gams_model, test_data)
residuals_gam <- y_test - predictions_gam
RMSE(predictions_gam, test_data$weekly_sales)
```

```
## [1] 162828
```

```
# and our model r2...

rss_gam =  sum(residuals_gam^2)
1 - (rss_gam/tss)
```

```
## [1] 0.9237802
```

```r
# # ----------------------------------------------------------------
#
# Stop! Go back to the presentation!
#
# # ----------------------------------------------------------------
```

```r
# # ----------------------------------------------------------------
#
# Your turn!
#
# # ----------------------------------------------------------------

# One thing tends to unite us COA weirdos: we're all fascinated by mushrooms!
# Upside: yummy! Downside: they can kill you.
# Your job is to build a model that can differentiate between poisonous and edible ones. :)
# Open up the mushrooms.csv file on the Google Drive.
# You must use the following code to generate your test/training split. After that, it's up to
# how you build the model. Best accuracy wins!
# If you need definitions of the dataset, check it out here:
# http://archive.ics.uci.edu/ml/datasets/Mushroom


mushrooms <- read.csv("w7 data/mushrooms.csv")

# pre-cleaning this for you :)
mushrooms_cleaned <- mushrooms %>%
  clean_names() %>%
  na.omit() %>%
  mutate_if(is.character, as.factor) %>%
  select(-c(bruises, gill_attachment, veil_type))

head(mushrooms)
```

```
##   class cap.shape cap.surface cap.color bruises odor gill.attachment
## 1     p         x           s         n       t    p               f
## 2     e         x           s         y       t    a               f
## 3     e         b           s         w       t    l               f
## 4     p         x           y         w       t    p               f
## 5     e         x           s         g       f    n               f
## 6     e         x           y         y       t    a               f
##   gill.spacing gill.size gill.color stalk.shape stalk.root
## 1            c         n          k           e          e
## 2            c         b          k           e          c
## 3            c         b          n           e          c
## 4            c         n          n           e          e
```

```
## 5            w         b          k           t          e
## 6            c         b          n           e          c
##   stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
## 1                        s                        s                      w
## 2                        s                        s                      w
## 3                        s                        s                      w
## 4                        s                        s                      w
## 5                        s                        s                      w
## 6                        s                        s                      w
##   stalk.color.below.ring veil.type veil.color ring.number ring.type
## 1                      w         p          w           o         p
## 2                      w         p          w           o         p
## 3                      w         p          w           o         p
## 4                      w         p          w           o         p
## 5                      w         p          w           o         e
## 6                      w         p          w           o         p
##   spore.print.color population habitat
## 1                 k          s       u
## 2                 n          n       g
## 3                 n          n       m
## 4                 k          s       u
## 5                 n          a       g
## 6                 k          n       g
```

```r
set.seed(12345)
mushroom_index <- createDataPartition(mushrooms_cleaned$class, p = .7, list=FALSE)
mushroom_training <- mushrooms_cleaned[ mushroom_index,]
mushroom_testing  <- mushrooms_cleaned[-mushroom_index,]


# now, let's build an xgboost model!
# XGBoost requires your data to be entirely numerical, so let's convert it!

X_prep_train <- mushroom_training %>% dplyr::select(-class)
X_prep_test <- mushroom_testing %>% dplyr::select(-class)
X_train = model.matrix(~.-1, data = X_prep_train)
y_train = mushroom_training$class
X_test = model.matrix(~.-1, data = X_prep_test)
y_test = mushroom_testing$class

# here we go!

xgboost_model <- train(
  x = X_train,
  y = y_train,
  method = "xgbTree",
  trControl = trainControl(method = 'cv', number = 10),
```

```r
  verbosity = 0
)

# let's peek...

xgboost_model
```

```
## eXtreme Gradient Boosting
##
## 5688 samples
##   94 predictor
##    2 classes: 'e', 'p'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5119, 5120, 5119, 5118, 5119, 5119, ...
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  Accuracy   Kappa
##   0.3  1          0.6               0.50        50      0.9933198  0.9866274
##   0.3  1          0.6               0.50       100      0.9980662  0.9961266
##   0.3  1          0.6               0.50       150      0.9989449  0.9978867
##   0.3  1          0.6               0.75        50      0.9945487  0.9890858
##   0.3  1          0.6               0.75       100      0.9980662  0.9961266
##   0.3  1          0.6               0.75       150      0.9989449  0.9978867
##   0.3  1          0.6               1.00        50      0.9940206  0.9880303
##   0.3  1          0.6               1.00       100      0.9980662  0.9961266
##   0.3  1          0.6               1.00       150      0.9989449  0.9978867
##   0.3  1          0.8               0.50        50      0.9945478  0.9890847
##   0.3  1          0.8               0.50       100      0.9980662  0.9961266
##   0.3  1          0.8               0.50       150      0.9989449  0.9978867
##   0.3  1          0.8               0.75        50      0.9947239  0.9894367
##   0.3  1          0.8               0.75       100      0.9980662  0.9961266
##   0.3  1          0.8               0.75       150      0.9989449  0.9978867
##   0.3  1          0.8               1.00        50      0.9936712  0.9873304
##   0.3  1          0.8               1.00       100      0.9980662  0.9961266
##   0.3  1          0.8               1.00       150      0.9989449  0.9978867
##   0.3  2          0.6               0.50        50      0.9991213  0.9982399
##   0.3  2          0.6               0.50       100      1.0000000  1.0000000
##   0.3  2          0.6               0.50       150      1.0000000  1.0000000
##   0.3  2          0.6               0.75        50      0.9992964  0.9985907
##   0.3  2          0.6               0.75       100      1.0000000  1.0000000
##   0.3  2          0.6               0.75       150      1.0000000  1.0000000
##   0.3  2          0.6               1.00        50      0.9992964  0.9985907
##   0.3  2          0.6               1.00       100      1.0000000  1.0000000
##   0.3  2          0.6               1.00       150      1.0000000  1.0000000
##   0.3  2          0.8               0.50        50      0.9998246  0.9996486
```

```
##    0.3  2          0.8           0.50        100     1.0000000  1.0000000
##    0.3  2          0.8           0.50        150     1.0000000  1.0000000
##    0.3  2          0.8           0.75         50     0.9996479  0.9992947
##    0.3  2          0.8           0.75        100     1.0000000  1.0000000
##    0.3  2          0.8           0.75        150     1.0000000  1.0000000
##    0.3  2          0.8           1.00         50     0.9991210  0.9982393
##    0.3  2          0.8           1.00        100     1.0000000  1.0000000
##    0.3  2          0.8           1.00        150     1.0000000  1.0000000
##    0.3  3          0.6           0.50         50     1.0000000  1.0000000
##    0.3  3          0.6           0.50        100     1.0000000  1.0000000
##    0.3  3          0.6           0.50        150     1.0000000  1.0000000
##    0.3  3          0.6           0.75         50     1.0000000  1.0000000
##    0.3  3          0.6           0.75        100     1.0000000  1.0000000
##    0.3  3          0.6           0.75        150     1.0000000  1.0000000
##    0.3  3          0.6           1.00         50     1.0000000  1.0000000
##    0.3  3          0.6           1.00        100     1.0000000  1.0000000
##    0.3  3          0.6           1.00        150     1.0000000  1.0000000
##    0.3  3          0.8           0.50         50     1.0000000  1.0000000
##    0.3  3          0.8           0.50        100     1.0000000  1.0000000
##    0.3  3          0.8           0.50        150     1.0000000  1.0000000
##    0.3  3          0.8           0.75         50     1.0000000  1.0000000
##    0.3  3          0.8           0.75        100     1.0000000  1.0000000
##    0.3  3          0.8           0.75        150     1.0000000  1.0000000
##    0.3  3          0.8           1.00         50     1.0000000  1.0000000
##    0.3  3          0.8           1.00        100     1.0000000  1.0000000
##    0.3  3          0.8           1.00        150     1.0000000  1.0000000
##    0.4  1          0.6           0.50         50     0.9963075  0.9926062
##    0.4  1          0.6           0.50        100     0.9987695  0.9975353
##    0.4  1          0.6           0.50        150     0.9989449  0.9978867
##    0.4  1          0.6           0.75         50     0.9968350  0.9936622
##    0.4  1          0.6           0.75        100     0.9989449  0.9978867
##    0.4  1          0.6           0.75        150     0.9989449  0.9978867
##    0.4  1          0.6           1.00         50     0.9973619  0.9947171
##    0.4  1          0.6           1.00        100     0.9989449  0.9978867
##    0.4  1          0.6           1.00        150     0.9991206  0.9982388
##    0.4  1          0.8           0.50         50     0.9970111  0.9940143
##    0.4  1          0.8           0.50        100     0.9985934  0.9971828
##    0.4  1          0.8           0.50        150     0.9989449  0.9978867
##    0.4  1          0.8           0.75         50     0.9978904  0.9957747
##    0.4  1          0.8           0.75        100     0.9989449  0.9978867
##    0.4  1          0.8           0.75        150     0.9991210  0.9982393
##    0.4  1          0.8           1.00         50     0.9966589  0.9933099
##    0.4  1          0.8           1.00        100     0.9985934  0.9971828
##    0.4  1          0.8           1.00        150     0.9989449  0.9978867
##    0.4  2          0.6           0.50         50     1.0000000  1.0000000
##    0.4  2          0.6           0.50        100     1.0000000  1.0000000
##    0.4  2          0.6           0.50        150     1.0000000  1.0000000
##    0.4  2          0.6           0.75         50     1.0000000  1.0000000
```

```
##   0.4  2          0.6                0.75       100       1.0000000  1.0000000
##   0.4  2          0.6                0.75       150       1.0000000  1.0000000
##   0.4  2          0.6                1.00        50       1.0000000  1.0000000
##   0.4  2          0.6                1.00       100       1.0000000  1.0000000
##   0.4  2          0.6                1.00       150       1.0000000  1.0000000
##   0.4  2          0.8                0.50        50       1.0000000  1.0000000
##   0.4  2          0.8                0.50       100       1.0000000  1.0000000
##   0.4  2          0.8                0.50       150       1.0000000  1.0000000
##   0.4  2          0.8                0.75        50       1.0000000  1.0000000
##   0.4  2          0.8                0.75       100       1.0000000  1.0000000
##   0.4  2          0.8                0.75       150       1.0000000  1.0000000
##   0.4  2          0.8                1.00        50       1.0000000  1.0000000
##   0.4  2          0.8                1.00       100       1.0000000  1.0000000
##   0.4  2          0.8                1.00       150       1.0000000  1.0000000
##   0.4  3          0.6                0.50        50       1.0000000  1.0000000
##   0.4  3          0.6                0.50       100       1.0000000  1.0000000
##   0.4  3          0.6                0.50       150       1.0000000  1.0000000
##   0.4  3          0.6                0.75        50       1.0000000  1.0000000
##   0.4  3          0.6                0.75       100       1.0000000  1.0000000
##   0.4  3          0.6                0.75       150       1.0000000  1.0000000
##   0.4  3          0.6                1.00        50       1.0000000  1.0000000
##   0.4  3          0.6                1.00       100       1.0000000  1.0000000
##   0.4  3          0.6                1.00       150       1.0000000  1.0000000
##   0.4  3          0.8                0.50        50       1.0000000  1.0000000
##   0.4  3          0.8                0.50       100       1.0000000  1.0000000
##   0.4  3          0.8                0.50       150       1.0000000  1.0000000
##   0.4  3          0.8                0.75        50       1.0000000  1.0000000
##   0.4  3          0.8                0.75       100       1.0000000  1.0000000
##   0.4  3          0.8                0.75       150       1.0000000  1.0000000
##   0.4  3          0.8                1.00        50       1.0000000  1.0000000
##   0.4  3          0.8                1.00       100       1.0000000  1.0000000
##   0.4  3          0.8                1.00       150       1.0000000  1.0000000
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 50, max_depth = 2, eta
##  = 0.4, gamma = 0, colsample_bytree = 0.6, min_child_weight = 1 and subsample
##  = 0.5.
```

```r
# let's assess the model accuracy
xgboost_model_pred <- predict(xgboost_model, X_test)
confusionMatrix(xgboost_model_pred, y_test)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    e    p
##          e 1262    0
##          p    0 1174
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5181
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5181
##          Detection Rate : 0.5181
##    Detection Prevalence : 0.5181
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```