Neural Network

Edited by: Yuzhou Wang

12/6/2020

Group 2 members:

Alston D'Souza Jiayi Liu Pengjin Wang Sachin Badole Yuzhou Wang

Basic Settings

This part has been eliminate from our project because RF is more efficient to solve the problems.

```
suppressMessages(library(tidyverse))
suppressMessages(library(data.table))
suppressMessages(library(factoextra))
suppressMessages(library(NbClust))

# Load Data
nndata <- data.frame(readRDS("../files for project/cleaned_data"))
```

```
# Construct Seasonal Variables
nndata$start_s <- ifelse(nndata$startdate_month %in% c(12, 1,
    2), "Winter", ifelse(nndata$startdate_month %in% c(3, 4,
    5), "Spring", ifelse(nndata$startdate_month %in% c(6, 7,
    8), "Summer", ifelse(nndata$startdate_month %in% c(9, 10,
    11), "Fall", NA))))

nndata$end_s <- ifelse(nndata$startdate_month %in% c(12, 1, 2),
    "Winter", ifelse(nndata$startdate_month %in% c(3, 4, 5),
        "Spring", ifelse(nndata$startdate_month %in% c(6, 7,
            8), "Summer", ifelse(nndata$startdate_month %in%
            c(9, 10, 11), "Fall", NA))))
```

```
# Categorize Variables
car_description <- c("maker", "interior", "exterior", "miles",
    "inspection", "doors", "trans", "model", "cyl", "warranty",
    "age", "age2")
listing_features <- c("text", "phone", "address", "store", "buyitnow",
    "photos", "addedinfo", "descriptionsize", "webpage", "caradphotos",
    "totallisted", "title", "html", "featured", "reserve", "auction",
    "primetime", "relist")
auction_time <- c("startdate_year", "startdate_month", "startdate_day",
    "startdate_hour", "startdate_minute", "startdate_second",
    "startdate_wday", "enddate_year", "enddate_month", "enddate_day",
```

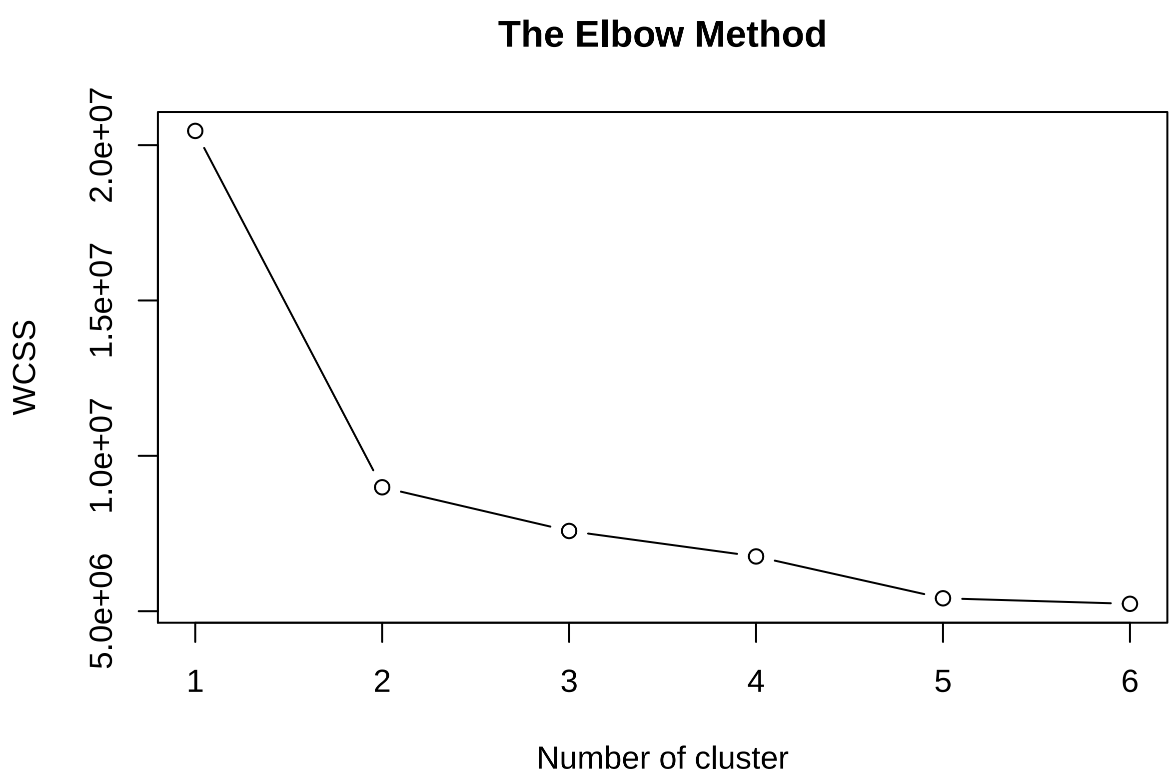
```r
    "enddate_hour", "enddate_minute", "enddate_second", "enddate_wday",
    "length", "week")
yz_customized_time <- c("startdate_year", "enddate_year", "startdate_wday",
    "enddate_wday", "length", "week")
auction_season <- c("start_s", "end_s")
car_quality <- c("ding_two", "ding_tiny", "ding_detectable",
    "ding_few", "scratch_two", "scratch_tiny", "scratch_detectable",
    "scratch_few", "dent_small", "dent_visible", "dent_two",
    "dent_tiny", "dent_detectable", "dent_few", "broken_two",
    "broken_tiny", "broken_detectable", "broken_few", "crack_wide",
    "crack_large", "crack_negligible", "crack_two", "crack_tiny",
    "crack_detectable", "crack_few", "crack_medium", "problem_one",
    "problem_two", "problem_tiny", "problem_detectable", "problem_few",
    "rust_two", "rust_tiny", "rust_detectable", "rust_few", "ding_bad",
    "ding_knowledge", "ding_pics", "dent_knowledge", "dent_pics",
    "crack_knowledge", "crack_pics", "problem_bad", "problem_knowledge",
    "problem_pics", "rust_knowledge", "rust_pics", "scratch_knowledge",
    "scratch_pics", "broken_bad", "broken_knowledge", "broken_pics",
    "ding_group", "scratch_group", "crack_group", "broken_group",
    "dent_group", "problem_group", "rust_group", "condition")
log_variables <- c("logmiles", "logtext", "logsize", "logstart",
    "logfdback", "logphotos", "logage", "loghtml")
seller_features <- c("software", "dealer", "negpct", "sellerborn",
    "sellerage", "pwrseller")
other <- c("numbids")

# Tweak this section for the variables that you want
# included:
types <- data.frame(vars = c("car_description", "listing_features",
    "auction_time", "yz_customized_time", "auction_season", "car_quality",
    "log_variables", "seller_features", "other"))

vars <- NULL
temp <- NULL
for (i in 1:nrow(types)) {
    temp <- data.frame(variable = eval(parse(text = types$vars[i])),
        category = types$vars[i])

    vars <- data.frame(rbind(vars, temp))
}
vars
##                variable           category
## 1                 maker    car_description
## 2              interior    car_description
## 3              exterior    car_description
## 4                 miles    car_description
## 5            inspection    car_description
## 6                 doors    car_description
## 7                 trans    car_description
## 8                 model    car_description
## 9                   cyl    car_description
## 10             warranty    car_description
## 11                  age    car_description
```

```
## 12               age2    car_description
## 13               text   listing_features
## 14              phone   listing_features
## 15            address   listing_features
## 16              store   listing_features
## 17           buyitnow   listing_features
## 18             photos   listing_features
## 19          addedinfo   listing_features
## 20    descriptionsize   listing_features
## 21            webpage   listing_features
## 22         caradphotos   listing_features
## 23        totallisted   listing_features
## 24              title   listing_features
## 25               html   listing_features
## 26           featured   listing_features
## 27            reserve   listing_features
## 28            auction   listing_features
## 29          primetime   listing_features
## 30             relist   listing_features
## 31     startdate_year       auction_time
## 32    startdate_month       auction_time
## 33      startdate_day       auction_time
## 34     startdate_hour       auction_time
## 35   startdate_minute       auction_time
## 36   startdate_second       auction_time
## 37     startdate_wday       auction_time
## 38       enddate_year       auction_time
## 39      enddate_month       auction_time
## 40        enddate_day       auction_time
## 41       enddate_hour       auction_time
## 42     enddate_minute       auction_time
## 43     enddate_second       auction_time
## 44       enddate_wday       auction_time
## 45             length       auction_time
## 46               week       auction_time
## 47     startdate_year yz_customized_time
## 48       enddate_year yz_customized_time
## 49     startdate_wday yz_customized_time
## 50       enddate_wday yz_customized_time
## 51             length yz_customized_time
## 52               week yz_customized_time
## 53            start_s     auction_season
## 54              end_s     auction_season
## 55           ding_two        car_quality
## 56          ding_tiny        car_quality
## 57    ding_detectable        car_quality
## 58           ding_few        car_quality
## 59        scratch_two        car_quality
## 60       scratch_tiny        car_quality
## 61 scratch_detectable        car_quality
## 62        scratch_few        car_quality
## 63         dent_small        car_quality
## 64       dent_visible        car_quality
```

```
## 65           dent_two         car_quality
## 66          dent_tiny         car_quality
## 67    dent_detectable         car_quality
## 68           dent_few         car_quality
## 69         broken_two         car_quality
## 70        broken_tiny         car_quality
## 71  broken_detectable         car_quality
## 72         broken_few         car_quality
## 73         crack_wide         car_quality
## 74        crack_large         car_quality
## 75   crack_negligible         car_quality
## 76          crack_two         car_quality
## 77         crack_tiny         car_quality
## 78   crack_detectable         car_quality
## 79          crack_few         car_quality
## 80       crack_medium         car_quality
## 81        problem_one         car_quality
## 82        problem_two         car_quality
## 83       problem_tiny         car_quality
## 84 problem_detectable         car_quality
## 85        problem_few         car_quality
## 86           rust_two         car_quality
## 87          rust_tiny         car_quality
## 88    rust_detectable         car_quality
## 89           rust_few         car_quality
## 90           ding_bad         car_quality
## 91     ding_knowledge         car_quality
## 92          ding_pics         car_quality
## 93     dent_knowledge         car_quality
## 94          dent_pics         car_quality
## 95    crack_knowledge         car_quality
## 96         crack_pics         car_quality
## 97        problem_bad         car_quality
## 98  problem_knowledge         car_quality
## 99       problem_pics         car_quality
## 100    rust_knowledge         car_quality
## 101         rust_pics         car_quality
## 102 scratch_knowledge         car_quality
## 103      scratch_pics         car_quality
## 104        broken_bad         car_quality
## 105  broken_knowledge         car_quality
## 106       broken_pics         car_quality
## 107        ding_group         car_quality
## 108     scratch_group         car_quality
## 109       crack_group         car_quality
## 110      broken_group         car_quality
## 111        dent_group         car_quality
## 112     problem_group         car_quality
## 113        rust_group         car_quality
## 114         condition         car_quality
## 115          logmiles       log_variables
## 116           logtext       log_variables
## 117           logsize       log_variables
```

```
## 118          logstart     log_variables
## 119         logfdback     log_variables
## 120         logphotos     log_variables
## 121            logage     log_variables
## 122           loghtml     log_variables
## 123          software   seller_features
## 124            dealer   seller_features
## 125            negpct   seller_features
## 126        sellerborn   seller_features
## 127         sellerage   seller_features
## 128         pwrseller   seller_features
## 129           numbids             other

## Prepare for KMeans
km.data <- nndata[, unique(c("sell", "biddy1", vars$variable[which(vars$category %in%
    c("car_description", "listing_features", "yz_customized_time",
        "auction_season", "car_quality", "seller_features", "other"))]))]
km.data$maker <- factor(km.data$maker, level = unique(km.data$maker),
    labels = 1:length(unique(km.data$maker)), exclude = NULL)
km.data$model <- factor(km.data$model, level = unique(km.data$model),
    labels = 1:length(unique(km.data$model)), exclude = NULL)
km.data$interior <- factor(km.data$interior, level = unique(km.data$interior),
    labels = 1:length(unique(km.data$interior)), exclude = NULL)
km.data$exterior <- factor(km.data$exterior, level = unique(km.data$exterior),
    labels = 1:length(unique(km.data$exterior)), exclude = NULL)
# km.data$location <-
# factor(km.data$location,level=unique(km.data$location),labels=1:length(unique(km.data$location)),excl
# = NULL)
km.data$software <- factor(km.data$software, level = unique(km.data$software),
    labels = 1:length(unique(km.data$software)), exclude = NULL)
km.data$caradphotos <- factor(km.data$caradphotos, level = unique(km.data$caradphotos),
    labels = 1:length(unique(km.data$caradphotos)), exclude = NULL)
km.data$start_s <- factor(km.data$start_s, level = unique(km.data$start_s),
    labels = 1:length(unique(km.data$start_s)), exclude = NULL)
km.data$end_s <- factor(km.data$end_s, level = unique(km.data$end_s),
    labels = 1:length(unique(km.data$end_s)), exclude = NULL)
km.data$startdate_year <- factor(km.data$startdate_year, level = unique(km.data$startdate_year),
    labels = 1:length(unique(km.data$startdate_year)), exclude = NULL)
km.data$enddate_year <- factor(km.data$enddate_year, level = unique(km.data$enddate_year),
    labels = 1:length(unique(km.data$enddate_year)), exclude = NULL)
km.data$reserve <- factor(km.data$reserve)
km.data$buyitnow <- factor(km.data$buyitnow)
km.data$store <- factor(km.data$store)
# km.data <- data.frame(sapply(km.data,as.numeric))
km.data <- na.omit(km.data)

x <- km.data[, which(!names(km.data) %in% c("sell", "biddy1"))]

factor <- lapply(km.data, is.factor)
km.num.data <- data.frame(lapply(km.data, as.numeric)) %>% drop_na()
km.num.data[, which(factor == 1)] <- km.num.data[, which(factor ==
    1)] - 1
## minus one because as.numeric indexing from 1 instead of 0
```

```r
##
rangeStandardize <- function(x) {
    (x - min(x))/diff(range(x))
}
x <- x %>% mutate_if(is.numeric, rangeStandardize)

x <- data.frame(lapply(x, as.numeric))

wcss = vector()
for (i in 1:6) wcss[i] = sum(kmeans(x, i)$withinss, na.rm = TRUE)
plot(1:6, wcss, type = "b", main = paste("The Elbow Method"),
    xlab = "Number of cluster", ylab = "WCSS")
```




```r
# Visualize k-means km = kmeans(x, 2, iter.max =300, nstart
# =200) y_kmeans = km$cluster

# km.num.data$cluster <- y_kmeans

# sum <- km.num.data %>% group_by(cluster) %>%
# summarise_all(.vars=c(),mean,na.omit=TRUE) data.frame(sum)

# km.num.data %>% mutate(cluster = km$cluster) %>%
# ggplot(aes(numbids, sell, color = factor(cluster), label =
# cluster)) + geom_text()
```

```
# km.num.data %>% select(cluster, miles, inspection,
# warranty, age, sell) %>% group_by(cluster) %>%
# summarise_all(.vars=c(),mean,na.omit=TRUE)



# km.num.data[c(listing_features,'sell','cluster')] %>%
# group_by(cluster) %>%
# summarise_all(.vars=c(),mean,na.omit=TRUE)
```

Neural Network

First we fit models for 'sell' (==1 for sold, ==0 for unsold)

```
library(h2o)
h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 days 12 hours
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    2 months and 2 days
##     H2O cluster name:           H2O_started_from_R_ellenyz_wun812
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   0.82 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.2 (2020-06-22)
```

```
km.num.data$start_s <- factor(km.num.data$start_s)
exclude <- c("end_s", "software", "age2", "biddy1", "sellerborn",
    "doors", "week", "maker", "dealer")
km.num.data <- km.num.data[, which(!names(km.num.data) %in% exclude)]
splits <- h2o.splitFrame(as.h2o(km.num.data), c(0.6, 0.2), seed = 1234)
```

```
##   |                                                                              |
```

```
train <- h2o.assign(splits[[1]], "train.hex")  # 60%
valid <- h2o.assign(splits[[2]], "valid.hex")  # 20%
test <- h2o.assign(splits[[3]], "test.hex")  # 20%
```

```r
classifier = h2o.deeplearning(y = "sell", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(64,
        32, 16), epochs = 100, variable_importances = T)
```

```
##   |                                                                              |
```

```r
# summary(classifier)
out <- head(as.data.frame(h2o.varimp(classifier)), 20)
out[, 2:4] <- round(out[, 2:4], 4)

out2 <- as.data.frame(h2o.varimp(classifier))
out2 <- out2[which(!out2$variable %in% car_quality), ]
out2[, 2:4] <- round(out2[, 2:4], 4)
out2 <- head(out2, 20)

nn_mse_643216 <- h2o.mse(classifier)

prob_pred = h2o.predict(classifier, newdata = as.h2o(test))
```

```
##   |                                                                              |
```

```r
y_pred = (prob_pred > 0.5)
y_pred = as.vector(y_pred)
# y_pred summary(y_pred)

check <- ifelse(y_pred == test$sell, 1, 0)
mean(check)
```

```
## [1] 0.3276762
```

```r
## repeat nn4 and try for other type of layers

classifier = h2o.deeplearning(y = "sell", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(32,
        32, 32), epochs = 10, variable_importances = T)
```

```
##   |                                                                              |
```

```r
out2 <- as.data.frame(h2o.varimp(classifier))
out2 <- out2[which(!out2$variable %in% car_quality), ]
out2[, 2:4] <- round(out2[, 2:4], 4)
out2 <- head(out2, 20)
nn_mse_323232 <- h2o.mse(classifier)

classifier = h2o.deeplearning(y = "sell", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(100),
    epochs = 10, variable_importances = T)
```

```
##   |                                                                              |
```

```
out2 <- as.data.frame(h2o.varimp(classifier))
out2 <- out2[which(!out2$variable %in% car_quality), ]
out2[, 2:4] <- round(out2[, 2:4], 4)
out2 <- head(out2, 20)
nn_mse_100 <- h2o.mse(classifier)

classifier = h2o.deeplearning(y = "sell", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(100,
        50), epochs = 10, variable_importances = T)
```

```
##   |                                                                            |
```

```
out2 <- as.data.frame(h2o.varimp(classifier))
out2 <- out2[which(!out2$variable %in% car_quality), ]
out2[, 2:4] <- round(out2[, 2:4], 4)
out2 <- head(out2, 20)
nn_mse_10050 <- h2o.mse(classifier)
```

Now selecting only variables being selected by the RF Model

```
var <- c("reserve", "caradphotos", "logstart", "buyitnow", "exterior",
    "software", "age", "photos", "interior", "logmiles", "age2",
    "enddate_wday", "startdate_wday", "logfdback", "miles", "logage",
    "logsize", "logphotos", "html", "loghtml")
nndata <- data.frame(readRDS("../files for project/cleaned_data"))
# nndata$sold <- 0 nndata$sold[which(nndata$numbids > 0)] <-
# 1
nndata <- nndata[c(var, "sell")]

# nndata$maker <-
# factor(nndata$maker,level=unique(nndata$maker),labels=1:length(unique(nndata$maker)),exclude
# = NULL) nndata$model<-
# factor(nndata$model,level=unique(nndata$model),labels=1:length(unique(nndata$model)),exclude
# = NULL)
nndata$interior <- factor(nndata$interior, level = unique(nndata$interior),
    labels = 1:length(unique(nndata$interior)), exclude = NULL)
nndata$exterior <- factor(nndata$exterior, level = unique(nndata$exterior),
    labels = 1:length(unique(nndata$exterior)), exclude = NULL)
# nndata$location <-
# factor(nndata$location,level=unique(nndata$location),labels=1:length(unique(nndata$location)),exclude
# = NULL)
nndata$software <- factor(nndata$software, level = unique(nndata$software),
    labels = 1:length(unique(nndata$software)), exclude = NULL)
# nndata <- nndata[which(nndata$software!=1 &
# nndata$software!=2),]

nndata <- data.frame(lapply(nndata, as.numeric))
nndata <- na.omit(nndata)

library(h2o)
h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 days 12 hours
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    2 months and 2 days
##     H2O cluster name:           H2O_started_from_R_ellenyz_wun812
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   0.81 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.2 (2020-06-22)
splits <- h2o.splitFrame(as.h2o(nndata), c(0.6, 0.2), seed = 1234)
##   |                                                                            |
train <- h2o.assign(splits[[1]], "train.hex")  # 60%
valid <- h2o.assign(splits[[2]], "valid.hex")  # 20%
test <- h2o.assign(splits[[3]], "test.hex")  # 20%
classifier2 = h2o.deeplearning(y = "sell", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(64,
        32, 16), epochs = 100, variable_importances = T)
##   |                                                                            |

# summary(classifier2)
# head(as.data.frame(h2o.varimp(classifier2)))

prob_pred = h2o.predict(classifier, newdata = test)
##   |                                                                            |
y_pred = (prob_pred > 0.5)
y_pred = as.vector(y_pred)
# y_pred summary(y_pred)

# data.frame(nn_mse_100,nn_mse_10050,nn_mse_323232,nn_mse_643216)
```

NN for Subset of Samples (sell==1)

From now we implement with sub-set with only sell==1, Implement NN for prediction on bid Remove some variabnle not making sense like the end season, the seller born…

```
factor <- lapply(km.data, is.factor)
nndata2 <- data.frame(lapply(km.data, as.numeric)) %>% drop_na()
nndata2[, which(factor == 1)] <- nndata2[, which(factor == 1)] -
    1  ## minus one because as.numeric indexing from 1 instead of 0
nndata2 <- nndata2[which(nndata2$sell == 1), ]
##
rangeStandardize <- function(x) {
```

```
    (x - min(x))/diff(range(x))
}
nndata2[, which(!names(nndata2) == "biddy1")] <- nndata2[, which(!names(nndata2) ==
    "biddy1")] %>% mutate_if(is.numeric, rangeStandardize)
nndata2 <- data.frame(lapply(nndata2, as.numeric))

library(h2o)
h2o.init(nthreads = -1)
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 days 12 hours
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    2 months and 2 days
##     H2O cluster name:           H2O_started_from_R_ellenyz_wun812
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   0.81 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.2 (2020-06-22)
nndata2$start_s <- factor(nndata2$start_s)
exclude <- c("end_s", "software", "age2", "sell", "sellerborn",
    "doors", "dealer", "week")
nndata2 <- nndata2[, which(!names(nndata2) %in% exclude)]
nndata2$biddy1 <- log(nndata2$biddy1)
splits <- h2o.splitFrame(as.h2o(nndata2), c(0.6, 0.2), seed = 1234)
##   |                                                                               |
train <- h2o.assign(splits[[1]], "train.hex")  # 60%
valid <- h2o.assign(splits[[2]], "valid.hex")  # 20%
test <- h2o.assign(splits[[3]], "test.hex")  # 20%
classifier = h2o.deeplearning(y = "biddy1", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(64,
        32, 16), epochs = 100, variable_importances = T, export_weights_and_biases = T)
##   |                                                                               |
summary(classifier)
## Model Details:
## ==============
##
## H2ORegressionModel: deeplearning
## Model Key:  DeepLearning_model_R_1607361727750_194
## Status of Neuron Layers: predicting biddy1, regression, gaussian distribution, Quadratic loss, 8,449
##   layer units      type dropout       l1       l2 mean_rate rate_rms momentum
## 1     1    90     Input  0.00 %       NA       NA        NA       NA       NA
## 2     2    64 Rectifier  0.00 % 0.000000 0.000000  0.043850 0.126714 0.000000
## 3     3    32 Rectifier  0.00 % 0.000000 0.000000  0.001758 0.000938 0.000000
```

```
## 4     4    16 Rectifier  0.00 % 0.000000 0.000000  0.007783 0.014922 0.000000
## 5     5     1    Linear      NA 0.000000 0.000000  0.000457 0.000232 0.000000
##   mean_weight weight_rms mean_bias bias_rms
## 1          NA         NA        NA       NA
## 2    0.017746   0.153444  0.415626 0.114109
## 3   -0.031999   0.168234  0.957127 0.105576
## 4   -0.030104   0.225266  0.901264 0.194175
## 5    0.004602   0.249956 -0.147898 0.000000
##
## H2ORegressionMetrics: deeplearning
## ** Reported on training data. **
## ** Metrics reported on temporary training frame with 9946 samples **
##
## MSE:  0.3726276
## RMSE:  0.6104323
## MAE:  0.4417606
## RMSLE:  0.0729631
## Mean Residual Deviance :  0.3726276
##
##
## H2ORegressionMetrics: deeplearning
## ** Reported on validation data. **
## ** Metrics reported on full validation frame **
##
## MSE:  0.4589332
## RMSE:  0.6774461
## MAE:  0.4960762
## RMSLE:  0.07670435
## Mean Residual Deviance :  0.4589332
##
##
##
##
## Scoring History:
##              timestamp   duration training_speed    epochs iterations
## 1  2020-12-11 00:18:13  0.000 sec             NA   0.00000          0
## 2  2020-12-11 00:18:16  3.155 sec  33597 obs/sec   5.02002          1
## 3  2020-12-11 00:18:22  9.063 sec  34179 obs/sec  15.05690          3
## 4  2020-12-11 00:18:28 14.334 sec  35814 obs/sec  25.10637          5
## 5  2020-12-11 00:18:33 19.428 sec  36858 obs/sec  35.13878          7
## 6  2020-12-11 00:18:40 27.036 sec  37684 obs/sec  50.18449         10
## 7  2020-12-11 00:18:47 34.219 sec  38659 obs/sec  65.23135         13
## 8  2020-12-11 00:18:55 41.252 sec  39409 obs/sec  80.28162         16
## 9  2020-12-11 00:19:01 47.527 sec  40581 obs/sec  95.33335         19
## 10 2020-12-11 00:19:03 49.702 sec  40890 obs/sec 100.35211         20
## 11 2020-12-11 00:19:03 49.818 sec  40875 obs/sec 100.35211         20
##           samples training_rmse training_deviance training_mae training_r2
## 1        0.000000            NA                NA           NA          NA
## 2   100054.000000       0.65931           0.43469      0.48622     0.67883
## 3   300099.000000       0.61043           0.37263      0.44176     0.72468
## 4   500395.000000       0.58180           0.33849      0.41867     0.74990
## 5   700351.000000       0.54398           0.29592      0.39477     0.78136
## 6  1000227.000000       0.51629           0.26655      0.37941     0.80306
```

```
## 7  1300126.000000       0.49489            0.24491      0.36393     0.81905
## 8  1600093.000000       0.48891            0.23904      0.36028     0.82339
## 9  1900089.000000       0.48136            0.23171      0.35456     0.82880
## 10 2000118.000000       0.47949            0.22991      0.35461     0.83013
## 11 2000118.000000       0.61043            0.37263      0.44176     0.72468
##    validation_rmse validation_deviance validation_mae validation_r2
## 1               NA                  NA             NA            NA
## 2          0.68519             0.46949        0.50840       0.64098
## 3          0.67745             0.45893        0.49608       0.64905
## 4          0.68714             0.47217        0.49871       0.63893
## 5          0.68854             0.47409        0.50270       0.63746
## 6          0.69381             0.48138        0.51256       0.63189
## 7          0.71048             0.50477        0.52118       0.61400
## 8          0.72778             0.52967        0.52995       0.59496
## 9          0.73638             0.54226        0.53761       0.58533
## 10         0.73192             0.53571        0.53386       0.59034
## 11         0.67745             0.45893        0.49608       0.64905
##
## Variable Importances: (Extract with `h2o.varimp`)
## =================================================
##
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1      age            1.000000          1.000000   0.037764
## 2      cyl            0.689908          0.689908   0.026054
## 3    maker            0.627385          0.627385   0.023693
## 4    model            0.568913          0.568913   0.021485
## 5    title            0.426579          0.426579   0.016110
##
## ---
##               variable relative_importance scaled_importance percentage
## 85            dent_few            0.218671          0.218671   0.008258
## 86            ding_two            0.218594          0.218594   0.008255
## 87           crack_few            0.216215          0.216215   0.008165
## 88           dent_tiny            0.213258          0.213258   0.008054
## 89        scratch_pics            0.184964          0.184964   0.006985
## 90 start_s.missing(NA)            0.000000          0.000000   0.000000

out3 <- as.data.frame(h2o.varimp(classifier))
out3 <- out2[which(!out2$variable %in% car_quality), ]
out3[, 2:4] <- round(out2[, 2:4], 4)
out3 <- head(out2, 20)

nn_mse_bid <- h2o.mse(classifier)

# w1 <- as.data.frame(h2o.weights(classifier, matrix_id=1))
# w2 <- as.data.frame(h2o.weights(classifier, matrix_id=2))
# w3 <- as.data.frame(h2o.weights(classifier, matrix_id=3))
# w4 <- as.data.frame(h2o.weights(classifier, matrix_id=4))
```

```
classifier = h2o.deeplearning(y = "biddy1", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(100),
    epochs = 100, variable_importances = T, export_weights_and_biases = T)
```

```r
##   |                                                                          |
# summary(classifier)

nn_mse_bid100 <- h2o.mse(classifier)

classifier = h2o.deeplearning(y = "biddy1", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(100,
        50), epochs = 100, variable_importances = T, export_weights_and_biases = T)
##   |                                                                          |

nn_mse_bid10050 <- h2o.mse(classifier)

classifier = h2o.deeplearning(y = "biddy1", training_frame = train,
    validation_frame = valid, activation = "Rectifier", hidden = c(32,
        32, 32), epochs = 100, variable_importances = T, export_weights_and_biases = T)
##   |                                                                          |

nn_mse_bid_323232 <- h2o.mse(classifier)
data.frame(nn_mse_bid100, nn_mse_bid10050, nn_mse_bid_323232,
    nn_mse_bid)
##   nn_mse_bid100 nn_mse_bid10050 nn_mse_bid_323232 nn_mse_bid
## 1     0.3818587       0.4043673         0.3167722  0.3726276
```