Kaggle Playground Season 3 Episode 5 - Wine Quality Data Competition

This notebook is to test some modelstacking and AutoML routines in R

The Kaggle Playground competitions provide monthly opportunities to practice skills on realistic, real world type data sets.

The challenge is to predict wine quality ratings (between 3 and 8), based on wine chemistry data.

Credits to: https://bradleyboehmke.github.io/HOML/stacking.html (https://bradleyboehmke.github.io/HOML/stacking.html)

More on h20 here:

https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/score_tree_interval.html (https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/score_tree_interval.html)

https://docs.h2o.ai/h2o/latest-stable/h2o-docs/performance-and-prediction.html (https://docs.h2o.ai/h2o/latest-stable/h2o-docs/performance-and-prediction.html)

Load Packages

In [1]:

```
#load packages

package_names <- c("tidyverse", "caret", "UBL", "corrplot", "rsample", "recipe
s", "h2o")

check_install_load_packages <- function(package_names){
    for (i in package_names) {
           # if(!i %in% installed.packages()){
           #     install.packages(i, dependencies = c("Depends", "Suggests")))
           #     }
            if(!i %in% (.packages())){
                library(i, character.only = TRUE)
                }
    }
}

check_install_load_packages(package_names)
(.packages())
```

```
── Attaching packages ─────────────────────────────────────── tidyverse
1.3.2 ──
✓ ggplot2 3.4.0      ✓ purrr   1.0.1
✓ tibble  3.1.8      ✓ dplyr   1.0.10
✓ tidyr   1.2.1      ✓ stringr 1.5.0
✓ readr   2.1.3      ✓ forcats 0.5.2
── Conflicts ────────────────────────────────────────── tidyverse_confl
icts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
Loading required package: lattice


Attaching package: ‘caret’


The following object is masked from ‘package:purrr’:

    lift


The following object is masked from ‘package:httr’:

    progress


Loading required package: MBA

Loading required package: gstat

Loading required package: automap

Loading required package: sp

Loading required package: 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


corrplot 0.92 loaded


Attaching package: ‘recipes’


The following object is masked from ‘package:stringr’:

    fixed


The following object is masked from ‘package:stats’:

    step



------------------------------------------------------------------
---

Your next step is to start H2O:
    > h2o.init()

For H2O package documentation, ask for help:
    > ??h2o

After starting H2O, you can use the Web UI at http://localhost:5432
1
For more information visit https://docs.h2o.ai

------------------------------------------------------------------
---
```

```
Attaching package: ‘h2o’


The following objects are masked from ‘package:stats’:

    cor, sd, var


The following objects are masked from ‘package:base’:

    &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
    colnames<-, ifelse, is.character, is.factor, is.numeric, log,
    log10, log1p, log2, round, signif, trunc
```

'h2o' · 'recipes' · 'rsample' · 'corrplot' · 'UBL' · 'randomForest' · 'automap' · 'sp' · 'gstat' · 'MBA' · 'caret' · 'lattice' · 'forcats' · 'stringr' · 'dplyr' · 'purrr' · 'readr' · 'tidyr' · 'tibble' · 'ggplot2' · 'tidyverse' · 'stats' · 'graphics' · 'grDevices' · 'utils' · 'datasets' · 'bigrquery' · 'httr' · 'methods' · 'base'

Load Data

In [2]:

```
#import data
training <- read.csv(file = "/kaggle/input/playground-series-s3e5/train.csv")
testing <- read.csv(file = "/kaggle/input/playground-series-s3e5/test.csv")
head(training, 3)
dim(training)
table(training$quality)
dim(testing)
head(testing, 3)

#the outcome variable was first considered a factor but then realised to be better
suited to a regression model insead of
#classification, since the magnitude of the number has definite meaning
#training$quality <- factor(training$quality)
```

A data.frame: 3 × 13

	Id	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxid
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0	8.0	0.50	0.39	2.2	0.073	30
2	1	9.3	0.30	0.73	2.3	0.092	30
3	2	7.1	0.51	0.03	2.1	0.059	3

2056 · 13

3 4 5 6 7 8
12 55 839 778 333 39

1372 · 12

A data.frame: 3 × 12

	Id	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxid
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	2056	7.2	0.510	0.01	2	0.077	31
2	2057	7.2	0.755	0.15	2	0.102	14
3	2058	8.4	0.460	0.40	2	0.065	21

Setup h20

In [3]:

```
h2o.init()

# Split the data
ames <- AmesHousing::make_ames()
set.seed(123)  # for reproducibility
split <- initial_split(training, strata = "quality")
train <- training(split)
validate <- testing(split)

# Make sure we have consistent categorical levels
blueprint <- recipe(quality ~ ., data = train) %>%
  step_other(all_nominal(), threshold = 0.005)

# Create training & test sets for h2o
train_h2o <- prep(blueprint, training = train, retain = TRUE) %>%
  juice() %>%
  as.h2o()
validate_h2o <- prep(blueprint, training = train) %>%
  bake(new_data = validate) %>%
  as.h2o()

# Get response and feature names
Y <- "quality"
remove <- c("quality", "Id")
X <- setdiff(names(train), remove)
```

```
H2O is not running yet, starting it now...

Note:  In case of errors look at the following log files:
    /tmp/Rtmp6GVrDe/filee2ae636c0/h2o_UnknownUser_started_from_r.ou
t
    /tmp/Rtmp6GVrDe/filee76036f45/h2o_UnknownUser_started_from_r.er
r


Starting H2O JVM and connecting: .... Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         2 seconds 964 milliseconds
    H2O cluster timezone:       Etc/UTC
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.38.0.1
    H2O cluster version age:    6 months and 9 days !!!
    H2O cluster name:           H2O_started_from_R_root_pdq307
    H2O cluster total nodes:    1
    H2O cluster total memory:   7.50 GB
    H2O cluster total cores:    4
    H2O cluster allowed cores:  4
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    R Version:                  R version 4.0.5 (2021-03-31)


Warning message in h2o.clusterInfo():
“
Your H2O cluster version is too old (6 months and 9 days)!
Please download and install the latest version from http://h2o.ai/d
ownload/”


  |=================================================================
======| 100%
  |=================================================================
======| 100%
```

In [4]:

```
# Train & cross-validate a GLM model
best_glm <- h2o.glm(
  x = X, y = Y, training_frame = train_h2o, alpha = 0.1,
  remove_collinear_columns = TRUE, nfolds = 10, fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE, seed = 123
)

# Train & cross-validate a RF model
best_rf <- h2o.randomForest(
  x = X, y = Y, training_frame = train_h2o, ntrees = 1000,
  max_depth = 30, min_rows = 1, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123, stopping_rounds = 50, stopping_metric = "RMSE",
  stopping_tolerance = 0, score_tree_interval = 50
)

# Train & cross-validate a GBM model
best_gbm <- h2o.gbm(
  x = X, y = Y, training_frame = train_h2o, ntrees = 5000, learn_rate = 0.01,
  max_depth = 7, min_rows = 5, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123, stopping_rounds = 50, stopping_metric = "RMSE",
  stopping_tolerance = 0, score_tree_interval = 50
)

# Train & cross-validate an XGBoost model
best_xgb <- h2o.xgboost(
  x = X, y = Y, training_frame = train_h2o, ntrees = 5000, learn_rate = 0.05,
  max_depth = 3, min_rows = 3, sample_rate = 0.8,
  nfolds = 10, fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE, seed = 123, stopping_rounds = 50,
  stopping_metric = "RMSE", stopping_tolerance = 0,
  score_tree_interval = 50
)

#save models to working dir
saveRDS(best_glm, "/kaggle/working/GLM.rds")
saveRDS(best_rf, "/kaggle/working/RF.rds")
saveRDS(best_gbm, "/kaggle/working/GBM.rds")
saveRDS(best_xgb, "/kaggle/working/XGB.rds")
```

```
  |======================================================================
======| 100%
  |======================================================================
======| 100%
  |======================================================================
======| 100%
  |======================================================================
======| 100%
```

In [5]:

```
#load models
#best_glm <- loadRDS("/kaggle/working/GLM.rds")
#best_rf <- loadRDS("/kaggle/working/RF.rds")
#best_gbm <- loadRDS("/kaggle/working/GBM.rds")
#best_xgb <- loadRDS("/kaggle/working/XGB.rds")
```

In [6]:

```
# Train a stacked tree ensemble
ensemble_tree <- h2o.stackedEnsemble(
  x = X, y = Y, training_frame = train_h2o, model_id = "my_tree_ensemble",
  base_models = list(best_glm, best_rf, best_gbm, best_xgb),
  metalearner_algorithm = "drf"
)
#save models
saveRDS(ensemble_tree, "/kaggle/working/ensTree.rds")
```

```
Warning message in .h2o.processResponseWarnings(res):
“Dropping unused columns: [Id].
”
```

```
  |======================================================================
======| 100%
```

In [7]:

```
#load ensemble model
#ensemble_tree <- loadRDS("/kaggle/working/ensTree.rds")
```

In [8]:

```
# Get results from base learners
get_rmse <- function(model) {
  results <- h2o.performance(model, newdata = validate_h2o)
  results@metrics$RMSE
}
print("RMSE of base learners: best_glm, best_rf, best_gbm, best_xgb")
list(best_glm, best_rf, best_gbm, best_xgb) %>%
  purrr::map_dbl(get_rmse)

# Stacked results
perf <- h2o.performance(ensemble_tree, newdata = validate_h2o)
print("RMSE of stacked results: ")
perf@metrics$RMSE
```

```
[1] "RMSE of base learners: best_glm, best_rf, best_gbm, best_xgb"
```

0.71695224212299 · 0.711428310015168 · 0.755981160713386 · 0.793438524989041

```
[1] "RMSE of stacked results: "
```

0.715120253272267

Discussion of Results

The RF has the lowest RMSE of the base learners. Disappointingly, based on RMSE, the stacked model isn't better that the top two base learners.

Validation

In [9]:

```
set.seed(100)
print("Results on validation set prediction for ensemble model")

pred_ensemble <- predict(object = ensemble_tree, newdata = validate_h2o)
pred_ensemble_df <- as.data.frame(pred_ensemble)



outcome_ensemble <- factor(round(pred_ensemble_df$predict), levels = 3:8)
val <- as.data.frame(validate_h2o)
true <- factor(val$quality, levels = 3:8)

result_ensemble <- confusionMatrix(true, outcome_ensemble)
result_ensemble$table

set.seed(100)
print("Results on validation set prediction for top base learner model")

pred_rf <- predict(object = best_rf, newdata = validate_h2o)
pred_rf_df <- as.data.frame(pred_rf)

outcome_rf <- factor(round(pred_rf_df$predict), levels = 3:8)

result_rf <- confusionMatrix(true, outcome_rf)
result_rf$table
```

```
[1] "Results on validation set prediction for ensemble model"
  |======================================================================
======| 100%


          Reference
Prediction   3   4   5   6   7   8
         3   0   0   1   1   0   0
         4   0   0   8   3   0   0
         5   0   0 146  63   5   0
         6   0   0  55 128  12   0
         7   0   0  12  51  20   0
         8   0   0   0   5   5   0


[1] "Results on validation set prediction for top base learner mode
l"
  |======================================================================
======| 100%


          Reference
Prediction   3   4   5   6   7   8
         3   0   0   1   1   0   0
         4   0   0   8   3   0   0
         5   0   0 152  61   1   0
         6   0   0  53 132  10   0
         7   0   0  10  56  17   0
         8   0   0   1   6   3   0
```

Compare to auto_ml candidate search

In [10]:

```
#search for best models
# Use AutoML to find a list of candidate models (i.e., leaderboard)
auto_ml <- h2o.automl(
  x = X, y = Y, training_frame = train_h2o, nfolds = 5,
  max_runtime_secs = 60 * 15, max_models = 50,
  keep_cross_validation_predictions = TRUE, sort_metric = "RMSE", seed = 123,
  stopping_rounds = 50, stopping_metric = "RMSE", stopping_tolerance = 0
)

# Assess the leader board; the following truncates the results to show the top
# and bottom 15 models. You can get the top model with auto_ml@leader
auto_ml@leaderboard %>%
  as.data.frame() %>%
  dplyr::select(model_id, rmse) %>%
  dplyr::slice(1:25)
```

```
  |
|   0%
  |=================================================================
======| 100%
```

A data.frame: 21 × 2

model_id	rmse
<chr>	<dbl>
XRT_1_AutoML_1_20230328_213023	0.6992789
DRF_1_AutoML_1_20230328_213023	0.7007516
GBM_1_AutoML_1_20230328_213023	0.7113893
GLM_1_AutoML_1_20230328_213023	0.7142504
GBM_5_AutoML_1_20230328_213023	0.7291048
GBM_4_AutoML_1_20230328_213023	0.7395562
GBM_3_AutoML_1_20230328_213023	0.7424162
GBM_2_AutoML_1_20230328_213023	0.7433848
XGBoost_grid_1_AutoML_1_20230328_213023_model_9	0.7464224
XGBoost_grid_1_AutoML_1_20230328_213023_model_6	0.7489645
XGBoost_grid_1_AutoML_1_20230328_213023_model_2	0.7492352
XGBoost_grid_1_AutoML_1_20230328_213023_model_8	0.7534277
XGBoost_grid_1_AutoML_1_20230328_213023_model_1	0.7576452
XGBoost_grid_1_AutoML_1_20230328_213023_model_4	0.7791013
XGBoost_3_AutoML_1_20230328_213023	0.7791650
XGBoost_grid_1_AutoML_1_20230328_213023_model_7	0.7792803
XGBoost_2_AutoML_1_20230328_213023	0.7868100
XGBoost_1_AutoML_1_20230328_213023	0.8006827
XGBoost_grid_1_AutoML_1_20230328_213023_model_3	0.8093602
XGBoost_grid_1_AutoML_1_20230328_213023_model_5	0.8138360
DeepLearning_1_AutoML_1_20230328_213023	0.9347740

In [11]:

```
#save model list
leaderList <- auto_ml@leaderboard %>%
  as.data.frame() %>%
  dplyr::select(model_id, rmse) %>%
  dplyr::slice(1:25)

write.csv(leaderList, "/kaggle/working/leaderList.csv")

#extract models

#get leading model
model1 <- auto_ml@leader

#or other models
model_ids <- as.vector(auto_ml@leaderboard$model_id)

index <- 2
model2 <- h2o.getModel(model_ids[index])
index <- 3
model3 <- h2o.getModel(model_ids[index])
index <- 4
model4 <- h2o.getModel(model_ids[index])

# Train a stacked tree ensemble
ensemble_tree_AML <- h2o.stackedEnsemble(
  x = X, y = Y, training_frame = train_h2o, model_id = "my_AML_tree_ensemble",
  base_models = list(model1, model2, model3, model4),
  metalearner_algorithm = "drf"
)

# Get results from base learners
get_rmse <- function(model) {
  results <- h2o.performance(model, newdata = validate_h2o)
  results@metrics$RMSE
}
list(model1, model2, model3, model4) %>%
  purrr::map_dbl(get_rmse)

# Stacked results
h2o.performance(ensemble_tree_AML, newdata = validate_h2o)@metrics$RMSE
```

```
Warning message in .h2o.processResponseWarnings(res):
“Dropping unused columns: [Id].
”

  |=================================================================
======| 100%
```

0.71158241828779 · 0.710229621153381 · 0.733226376471326 · 0.715550546070982

0.738140382401868

Once again the ensemble does not outperform the base models. Using the best base model:

In [12]:

```
#Prediction and confusion matrix

set.seed(100)
print("Results on validation set prediction for best base model:")

pred_drf_AML <- predict(object = model2, newdata = validate_h2o)
pred_drf_df_AML <- as.data.frame(pred_drf_AML)


outcome_drf_AML <- factor(round(pred_drf_df_AML$predict), levels = 3:8)

result_drf_AML <- confusionMatrix(true, outcome_drf_AML)
result_drf_AML$table
```

```
[1] "Results on validation set prediction for best base model:"
  |======================================================================
======| 100%


          Reference
Prediction   3   4   5   6   7   8
         3   0   0   1   1   0   0
         4   0   0   7   4   0   0
         5   0   0 146  67   1   0
         6   0   0  47 138  10   0
         7   0   0  10  57  16   0
         8   0   0   1   6   3   0
```

Conclusion

The out of the box auto_ml came up with a model that beat the rf model that we came up with by a very narrow margin. The ensembles didn't help much.

In [13]:

```
#choose the best_rf model results for submission

test_h2o <- prep(blueprint, training = train) %>%
  bake(new_data = testing) %>%
  as.h2o()

set.seed(100)
print("Results on test set prediction for drf model")

pred_submit <- predict(object = model2, newdata = test_h2o)
pred_submit_df <- as.data.frame(pred_submit)


outcome_submit <- factor(round(pred_submit_df$predict), levels = 3:8)
head(outcome_submit)
length(outcome_submit)
dim(testing)[1]
```

```
  |======================================================================
======| 100%
[1] "Results on test set prediction for drf model"
  |======================================================================
======| 100%
```

5 · 6 · 6 · 6 · 6 · 6

▶ **Levels**:

1372

1372

In [14]:

```
write.csv(outcome_submit, "/kaggle/working/submission.csv")
```