H2O AutoML Binary Classification Demo

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code. To execute a code chunk, click *Run* (play) button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

If you're viewing the Rmd file (code only), but you'd like to see the code *and* output rendered as an HTML document, an online HTML of this file is available here (http://htmlpreview.github.io/?https://github.com/h2oai/h2o-tutorials/blob/master/h2o-world-2017/automl/R/automl_binary_classification_product_backorders.html).

Start H2O

Load the **h2o** R library and initialize a local H2O cluster.

```
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## 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:54321
## For more information visit http://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.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T//RtmpuIjLBM/h2o_me_started_from_r.out
##     /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T//RtmpuIjLBM/h2o_me_started_from_r.err
##
##
## Starting H2O JVM and connecting: .. Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         2 seconds 125 milliseconds
##     H2O cluster version:        3.16.0.2
##     H2O cluster version age:    3 days
##     H2O cluster name:           H2O_started_from_R_me_fid134
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.56 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:         XGBoost, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.3.2 (2016-10-31)
```

```
h2o.no_progress()  # Turn off progress bars for notebook readability
```

Load Data

For the AutoML binary classification demo, we use a subset of the Product Backorders (https://www.kaggle.com/tiredgeek/predict-bo-trial/data) dataset. The goal here is to predict whether or not a product will be put on backorder status, given a number of product metrics such as current inventory, transit time, demand forecasts and prior sales.

```
# Use local data file or download from GitHub
docker_data_path <- "/home/h2o/data/automl/product_backorders.csv"
if (file.exists(docker_data_path)) {
  data_path <- docker_data_path
} else {
  data_path <- "https://github.com/h2oai/h2o-tutorials/raw/master/h2o-world-2017/automl/data/pro
duct_backorders.csv"
}

# Load data into H2O
df <- h2o.importFile(data_path)
```

For classification, the response should be encoded as categorical (aka. “factor” or “enum”). Let’s take a look.

```
h2o.describe(df)
```

```
##                Label Type Missing Zeros PosInf NegInf     Min     Max
## 1                sku  int       0     0      0      0 1111620 3284775
## 2      national_inv  int       0  1858      0      0   -1440  730722
## 3         lead_time  int    1078   121      0      0       0      52
## 4     in_transit_qty  int       0 15432      0      0       0  170920
## 5   forecast_3_month  int       0 12118      0      0       0  479808
## 6   forecast_6_month  int       0 11136      0      0       0  967776
## 7   forecast_9_month  int       0 10604      0      0       0 1418208
## 8      sales_1_month  int       0 10278      0      0       0  186451
## 9      sales_3_month  int       0  8022      0      0       0  550609
## 10     sales_6_month  int       0  6864      0      0       0 1136154
## 11     sales_9_month  int       0  6231      0      0       0 1759152
## 12          min_bank  int       0  9909      0      0       0   85584
## 13   potential_issue enum       0 19032      0      0       0       1
## 14   pieces_past_due  int       0 18601      0      0       0   13824
## 15  perf_6_month_avg real       0   474      0      0     -99       1
## 16 perf_12_month_avg real       0   401      0      0     -99       1
## 17       local_bo_qty  int       0 18585      0      0       0    1440
## 18          deck_risk enum       0 14842      0      0       0       1
## 19      oe_constraint enum       0 19048      0      0       0       1
## 20          ppap_risk enum       0 16728      0      0       0       1
## 21      stop_auto_buy enum       0   657      0      0       0       1
## 22           rev_stop enum       0 19044      0      0       0       1
## 23 went_on_backorder enum       0 16787      0      0       0       1
##              Mean        Sigma Cardinality
## 1   2.059553e+06 6.633376e+05          NA
## 2   3.763670e+02 7.002072e+03          NA
## 3   7.706036e+00 6.778665e+00          NA
## 4   4.827235e+01 1.465999e+03          NA
## 5   1.829108e+02 4.304866e+03          NA
## 6   3.447398e+02 8.406062e+03          NA
## 7   4.977924e+02 1.218057e+04          NA
## 8   5.611888e+01 1.544218e+03          NA
## 9   1.685345e+02 4.581340e+03          NA
## 10  3.335322e+02 9.294566e+03          NA
## 11  5.042554e+02 1.418415e+04          NA
## 12  4.884071e+01 9.687739e+02          NA
## 13  1.102189e-03 3.318180e-02           2
## 14  2.311500e+00 1.102411e+02          NA
## 15 -6.519834e+00 2.597514e+01          NA
## 16 -6.053935e+00 2.518450e+01          NA
## 17  8.917756e-01 2.303335e+01          NA
## 18  2.210151e-01 4.149415e-01           2
## 19  2.624259e-04 1.619786e-02           2
## 20  1.220280e-01 3.273268e-01           2
## 21  9.655172e-01 1.824704e-01           2
## 22  4.723666e-04 2.172943e-02           2
## 23  1.189314e-01 3.237163e-01           2
```

We will notice that the response column, `"went_on_backorder"`, is already encoded as "enum", so there's nothing we need to do here. If it were encoded as a 0/1 "int", then we'd have to convert the column as follows: `df[,y] <- as.factor[,y]`

Next, let's identify the response & predictor columns by saving them as `x` and `y`. The `"sku"` column is a unique identifier so we'll want to remove that from the set of our predictors.

```
y <- "went_on_backorder"
x <- setdiff(names(df), c(y, "sku"))
```

Run AutoML

Run AutoML, stopping after 10 models. The `max_models` argument specifies the number of individual (or "base") models, and does not include the two ensemble models that are trained at the end.

```
aml <- h2o.automl(y = y, x = x,
                  training_frame = df,
                  max_models = 10,
                  seed = 1)
```

Leaderboard

Next, we will view the AutoML Leaderboard. Since we did not specify a `leaderboard_frame` in the `h2o.automl()` function for scoring and ranking the models, the AutoML leaderboard uses cross-validation metrics to rank the models.

A default performance metric for each machine learning task (binary classification, multiclass classification, regression) is specified internally and the leaderboard will be sorted by that metric. In the case of binary classification, the default ranking metric is Area Under the ROC Curve (AUC). In the future, the user will be able to specify any of the H2O metrics so that different metrics can be used to generate rankings on the leaderboard.

The leader model is stored at `aml@leader` and the leaderboard is stored at `aml@leaderboard`.

```
lb <- aml@leaderboard
```

Now we will view a snapshot of the top models. Here we should see the two Stacked Ensembles at or near the top of the leaderboard. Stacked Ensembles can almost always outperform a single model.

```
print(lb)
```

```
##                                               model_id      auc  logloss
## 1    StackedEnsemble_AllModels_0_AutoML_20171203_212856 0.947237 0.184152
## 2             GBM_grid_0_AutoML_20171203_212856_model_3 0.946676 0.175636
## 3 StackedEnsemble_BestOfFamily_0_AutoML_20171203_212856 0.946293 0.184704
## 4             GBM_grid_0_AutoML_20171203_212856_model_2 0.945450 0.178495
## 5             GBM_grid_0_AutoML_20171203_212856_model_4 0.944599 0.179424
## 6             GBM_grid_0_AutoML_20171203_212856_model_1 0.943037 0.181671
##
## [12 rows x 3 columns]
```

To view the entire leaderboard, specify the `n` argument of the `print.H2OFrame()` function as the total number of rows:

```
print(lb, n = nrow(lb))
```

```
##                                                model_id      auc  logloss
## 1     StackedEnsemble_AllModels_0_AutoML_20171203_212856 0.947237 0.184152
## 2              GBM_grid_0_AutoML_20171203_212856_model_3 0.946676 0.175636
## 3  StackedEnsemble_BestOfFamily_0_AutoML_20171203_212856 0.946293 0.184704
## 4              GBM_grid_0_AutoML_20171203_212856_model_2 0.945450 0.178495
## 5              GBM_grid_0_AutoML_20171203_212856_model_4 0.944599 0.179424
## 6              GBM_grid_0_AutoML_20171203_212856_model_1 0.943037 0.181671
## 7              GBM_grid_0_AutoML_20171203_212856_model_0 0.941071 0.185334
## 8                           XRT_0_AutoML_20171203_212856 0.929800 0.214572
## 9              GBM_grid_0_AutoML_20171203_212856_model_5 0.924166 0.340390
## 10                          DRF_0_AutoML_20171203_212856 0.921531 0.227156
## 11             GLM_grid_0_AutoML_20171203_212856_model_0 0.735041 0.339752
## 12                 DeepLearning_0_AutoML_20171203_212856 0.581629 0.698937
##
## [12 rows x 3 columns]
```

Ensemble Exploration

To understand how the ensemble works, let's take a peek inside the Stacked Ensemble "All Models" model. The "All Models" ensemble is an ensemble of all of the individual models in the AutoML run. This is often the top performing model on the leaderboard.

```
# Get model ids for all models in the AutoML Leaderboard
model_ids <- as.data.frame(aml@leaderboard$model_id)[,1]
# Get the "All Models" Stacked Ensemble model
se <- h2o.getModel(grep("StackedEnsemble_AllModels", model_ids, value = TRUE)[1])
# Get the Stacked Ensemble metalearner model
metalearner <- h2o.getModel(se@model$metalearner$name)
```

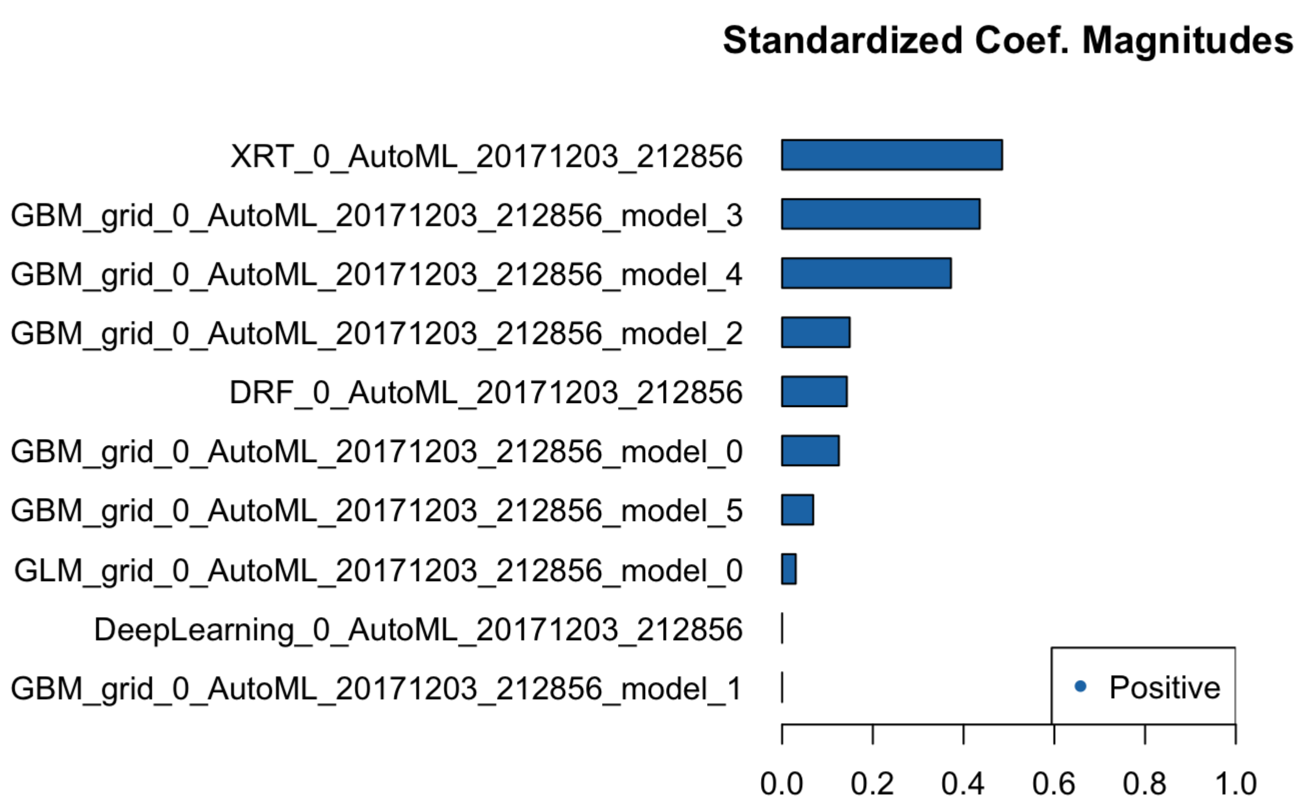
Examine the variable importance of the metalearner (combiner) algorithm in the ensemble. This shows us how much each base learner is contributing to the ensemble. The AutoML Stacked Ensembles use the default metalearner algorithm (GLM with non-negative weights), so the variable importance of the metalearner is actually the standardized coefficient magnitudes of the GLM.

```
h2o.varimp(metalearner)
```

```
## Standardized Coefficient Magnitudes: standardized coefficient magnitudes
##                                       names coefficients sign
## 1                XRT_0_AutoML_20171203_212856     0.484930  POS
## 2  GBM_grid_0_AutoML_20171203_212856_model_3     0.435628  POS
## 3  GBM_grid_0_AutoML_20171203_212856_model_4     0.372178  POS
## 4  GBM_grid_0_AutoML_20171203_212856_model_2     0.149037  POS
## 5                DRF_0_AutoML_20171203_212856     0.142688  POS
## 6  GBM_grid_0_AutoML_20171203_212856_model_0     0.125083  POS
## 7  GBM_grid_0_AutoML_20171203_212856_model_5     0.068494  POS
## 8  GLM_grid_0_AutoML_20171203_212856_model_0     0.030054  POS
## 9  GBM_grid_0_AutoML_20171203_212856_model_1     0.000000  POS
## 10     DeepLearning_0_AutoML_20171203_212856     0.000000  POS
```

We can also plot the base learner contributions to the ensemble.

```
h2o.varimp_plot(metalearner)
```



Save Leader Model

There are two ways to save the leader model – binary format and MOJO format. If you're taking your leader model to production, then we'd suggest the MOJO format since it's optimized for production use.

```
h2o.saveModel(aml@leader, path = "./product_backorders_model_bin")
```

```
## [1] "/Users/me/h2oai/code/h2o-tutorials/h2o-world-2017/automl/R/product_backorders_model_bin/
StackedEnsemble_AllModels_0_AutoML_20171203_212856"
```

```
h2o.download_mojo(aml@leader, path = "./")
```

```
## [1] "StackedEnsemble_AllModels_0_AutoML_20171203_212856.zip"
```