# Improving Model Predictions via Stacking and Hyper-Parameters Tuning

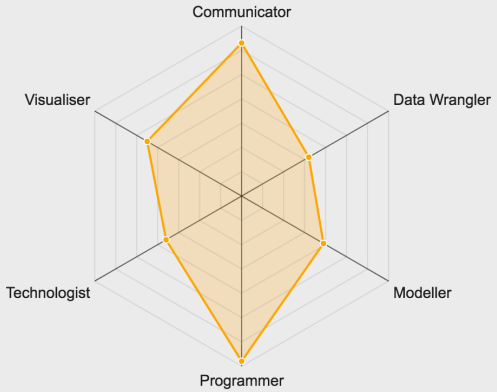

Jo-fai (Joe) Chow
Data Scientist
joe@h2o.ai
@matlabulus

# About Me

- 2005 - 2015
- Water Engineer
  - Consultant for Utilities
  - EngD Research
- 2015 - Present
- Data Scientist
  - Virgin Media
  - Domino Data Lab
  - H2O.ai

# Mango Data Science Radar

## You are a Programmer




When you are at the forefront of innovation, often the tools needed to solve a problem simply do not exist.

You may already be a master of multiple technical languages and enjoy adding languages to your skillset.

You combine carefully constructed analysis workflow, with robust pipelines to automate as much of the process as possible, but enjoy building applications from scratch.

You understand the value of planning, and know that thinking through an analysis is more efficient and less error prone than starting an analysis and seeing what sticks.

You always ensure that your customers are not misled by your work by providing thorough reporting, including lists of assumptions and descriptions of algorithms along with your findings.

You use software development approaches such as unit testing and version control to ensure that costly mistakes are not caused by your work.

Your cool head and rational approach serve as a great counterpoint to team members whose focus on the big picture can lead to key details being missed.

# About This Talk

- Predictive modelling
  - Kaggle as an example
- Improve predictions with simple tricks
- Use data science for social good 👍

# About Kaggle

- World's biggest predictive modelling competition platform
- 560k members
- Competition types:
  - Featured (prize)
  - Recruitment
  - Playground
  - 101

# Predicting Shelter Animal Outcomes

- X: Predictors
  - Name
  - Gender
  - Type (🐱 or 🐶)
  - Date & Time
  - Age
  - Breed
  - Colour

- Y: Outcomes (5 types)
  - Adoption
  - Died
  - Euthanasia
  - Return to Owner
  - Transfer

- Data
  - Training (27k samples)
  - Test (11k)

# Basic Feature Engineering

| X | Raw (Before) | Reformatted (After) |
|---|---|---|
| Name | Elsa, Steve, Lassie | [name_len]: 4, 5, 6 |
| Date & Time | 2014-02-12 18:22:00 | [year]: 2014<br>[month]: 2<br>[weekday]: 4<br>[hour]: 18 |
| Age | 1 year, 3 weeks, 2 days | [age_day]: 365, 21, 2 |
| Breed | German Shepherd, Pit Bull Mix | [is_mix]: 0, 1 |
| Colour | Brown Brindle/White | [simple_colour]: brown |

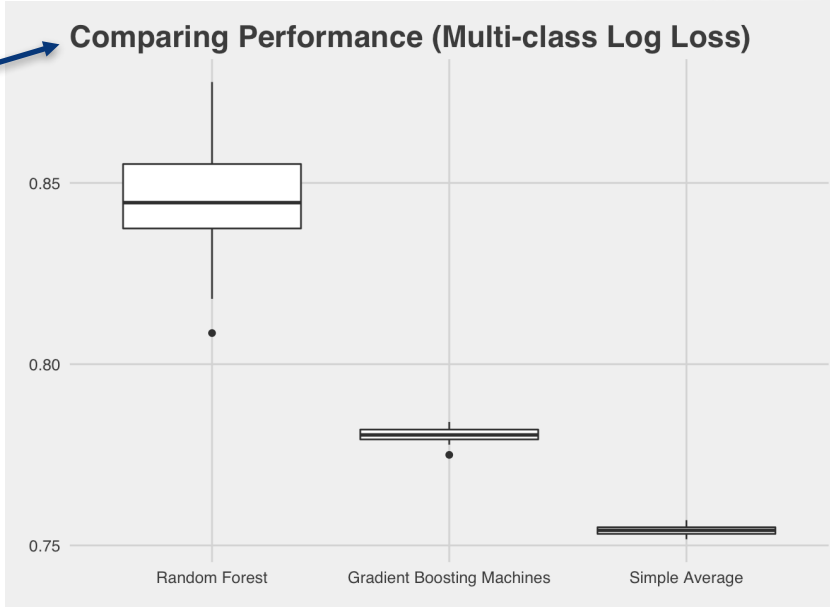
# Common Machine Learning Techniques

- Ensembles
  - Bagging/boosting of decision trees
  - Reduces variance and increase accuracy
  - Popular R Packages (used in next example)
    - “randomForest”
    - “xgboost”

- There are a lot more machine learning packages in R:
  - “caret”, “caretEnsemble”
  - “h2o”, “h2oEnsemble”
  - “mlr”

# Simple Trick – Model Averaging

- Stratified sampling
  - 80% for training
  - 20% for validation
- Evaluation metric
  - Multi-class Log Loss
  - Lower the better
  - 0 = Perfect
- 50 runs
  - different random seed



# More Advanced Methods

- Model Stacking
  - Uses a second-level metalearner to learn the optimal combination of base learners
  - R Packages:
    - “SuperLearner”
    - “subsemble”
    - “h2oEnsemble”
    - “caretEnsemble”
- Hyper-parameters Tuning
  - Improves the performance of individual machine learning algorithms
  - Grid search
    - Full / Random
  - R Packages:
    - “caret”
    - “h2o”

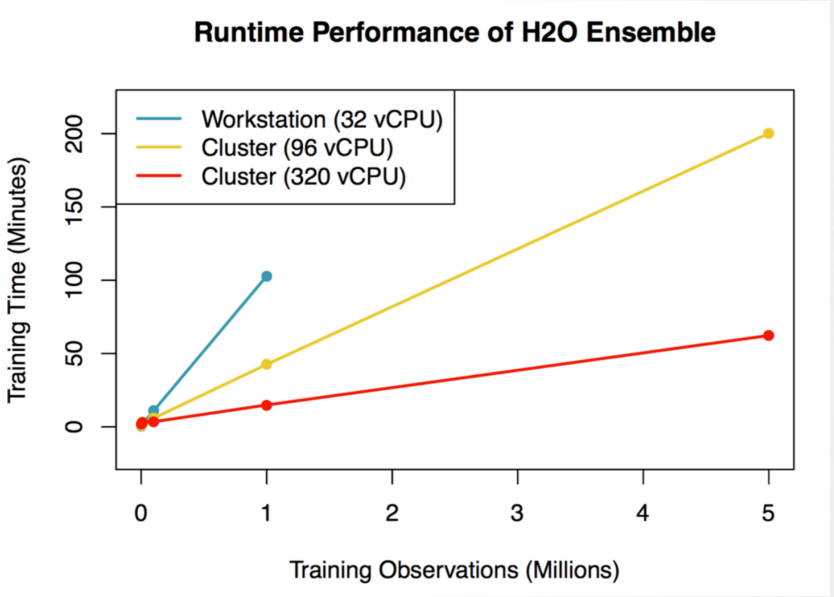
For more info, see
https://github.com/h2oai/h2o-meetups/tree/master/2016_05_20_MLconf_Seattle_Scalable_Ensembles
https://github.com/h2oai/h2o-3/blob/master/h2o-docs/src/product/tutorials/gbm/gbmTuning.Rmd

# Trade-Off of Advanced Methods

- Strength
  - Model tuning + stacking won nearly all Kaggle competitions.
  - Multi-algorithm ensemble may better approximate the true predictive function than any single algorithm.
- Weakness
  - Increased training and prediction times.
  - Increased model complexity.
  - Requires large machines or clusters for big data.

# R + H2O = Scalable Machine Learning

- H2O is an open-source, distributed machine learning library written in Java with APIs in R, Python and more.
- ”h2oEnsemble” is the scalable implementation of the Super Learner algorithm for H2O.



# H2O Random Grid Search Example

Define search range and criteria

```
# Define a list of hyper-parameters
h_param_gbm ← list(
  max_depth       = seq(5, 10, 1),
  sample_rate     = seq(0.5, 0.8, 0.1),
  col_sample_rate = seq(0.5, 0.8, 0.1)
  )

# Search Criteria
search_criteria = list(
  strategy         = "RandomDiscrete", # Use random grid search
  max_runtime_secs = 600               # limit the runtime to 10 minutes
)
```

```
# Grid Search
grid ← h2o.grid(
  hyper_params    = h_param_gbm,
  search_criteria = search_criteria,
  algorithm       = "gbm",           # which H2O algorithm to run

  # Standard model parameters
  training_frame   = hex_train,
  validation_frame = hex_valid,
  x                = features,
  y                = "OutcomeType",
```

```
H2O Grid Details
================

Grid ID: gbm_grid
Used hyper parameters:
  -  sample_rate
  -  max_depth
  -  col_sample_rate
Number of models: 21
Number of failed models: 0
```

Best models

```
Hyper-Parameter Search Summary: ordered by increasing logloss
  sample_rate max_depth col_sample_rate         model_ids           logloss
1         0.8         7              0.6  gbm_grid_model_2 0.752684243384921
2         0.8         6              0.5 gbm_grid_model_10  0.75318954895977
3         0.8         8              0.6  gbm_grid_model_4 0.753499220269728
4         0.7         6              0.6  gbm_grid_model_8 0.755403405990038
5         0.7         5              0.5 gbm_grid_model_14 0.756978014232093
---
   sample_rate max_depth col_sample_rate         model_ids           logloss
16         0.8        10              0.8 gbm_grid_model_12 0.765548067940223
17         0.6         9              0.7  gbm_grid_model_5 0.765760036358475
18         0.6        10              0.7 gbm_grid_model_16 0.769422559292015
19         0.5        10              0.8  gbm_grid_model_0 0.770202667369843
20         0.7        10              0.8 gbm_grid_model_17 0.773242423940347
21         0.6        10              0.5 gbm_grid_model_18 0.776840032616653
```

# H2O Model Stacking Example

Using h2o.stack(…) to combine multiple models

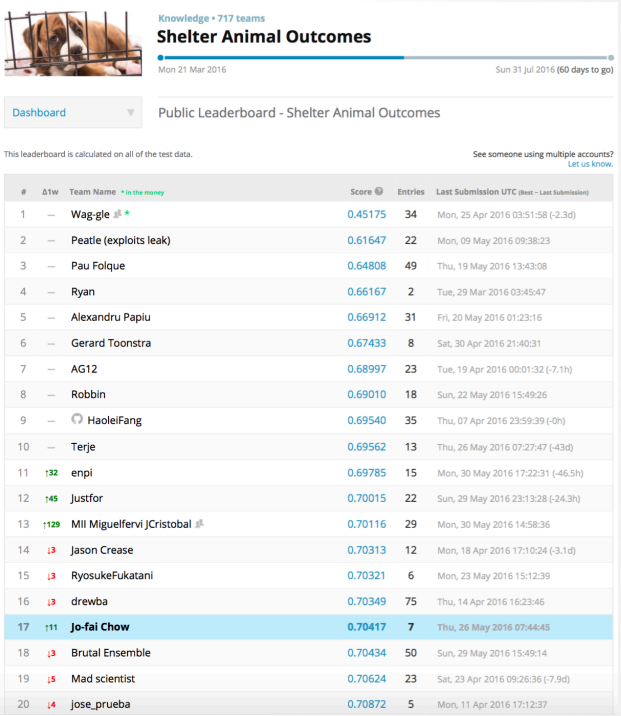
```
# H2O GBM
model_gbm <- h2o.gbm(training_frame = hex_train,
                     x = features, y = target,
                     nfolds = 5,
                     fold_assignment = "Modulo",
                     keep_cross_validation_predictions = TRUE)

# H2O Random Forest
model_drf <- h2o.randomForest(training_frame = hex_train,
                              x = features, y = target,
                              nfolds = 5,
                              fold_assignment = "Modulo",
                              keep_cross_validation_predictions = TRUE)

# Define list of models and metalearner
models <- list(model_gbm, model_drf)
metalearner <- "h2o.glm.wrapper"

# Run h2o.stack for metalearning
stack <- h2o.stack(models = models,
                   response_frame = hex_train[, target],
                   metalearner = metalearner,
                   keep_levelone_data = TRUE)
```

Getting reasonable results




| # | Δ1w | Team Name * In the money | Score | Entries | Last Submission UTC (Best - Last Submission) |
|---|---|---|---|---|---|
| 1 | – | Wag-gle * | 0.45175 | 34 | Mon, 25 Apr 2016 03:51:58 (-2.3d) |
| 2 | – | Peatle (exploits leak) | 0.61647 | 22 | Mon, 09 May 2016 09:38:23 |
| 3 | – | Pau Folque | 0.64808 | 49 | Thu, 19 May 2016 13:43:08 |
| 4 | – | Ryan | 0.66167 | 2 | Tue, 29 Mar 2016 03:45:47 |
| 5 | – | Alexandru Papiu | 0.66912 | 31 | Fri, 20 May 2016 01:23:16 |
| 6 | – | Gerard Toonstra | 0.67433 | 8 | Sat, 30 Apr 2016 21:40:31 |
| 7 | – | AG12 | 0.68997 | 23 | Tue, 19 Apr 2016 00:01:32 (-7.1h) |
| 8 | – | Robbin | 0.69010 | 18 | Sun, 22 May 2016 15:49:26 |
| 9 | – | HaoleiFang | 0.69540 | 35 | Thu, 07 Apr 2016 23:59:39 (-0h) |
| 10 | – | Terje | 0.69562 | 13 | Thu, 26 May 2016 07:27:47 (-43d) |
| 11 | ↑32 | enpi | 0.69785 | 15 | Mon, 30 May 2016 17:22:31 (-46.5h) |
| 12 | ↑45 | Justfor | 0.70015 | 22 | Sun, 29 May 2016 23:13:28 (-24.3h) |
| 13 | ↑129 | MII Miguelfervi jCristobal | 0.70116 | 29 | Mon, 30 May 2016 14:58:36 |
| 14 | ↓3 | Jason Crease | 0.70313 | 12 | Mon, 18 Apr 2016 17:10:24 (-3.1d) |
| 15 | ↓3 | RyosukeFukatani | 0.70321 | 6 | Mon, 23 May 2016 15:12:39 |
| 16 | ↓3 | drewba | 0.70349 | 75 | Thu, 14 Apr 2016 16:23:46 |
| 17 | ↑11 | **Jo-fai Chow** | **0.70417** | **7** | Thu, 26 May 2016 07:44:45 |
| 18 | ↓3 | Brutal Ensemble | 0.70434 | 50 | Sun, 29 May 2016 15:49:14 |
| 19 | ↓5 | Mad scientist | 0.70624 | 23 | Sat, 23 Apr 2016 09:26:36 (-7.9d) |
| 20 | ↓4 | jose_prueba | 0.70872 | 5 | Mon, 11 Apr 2016 17:12:37 |

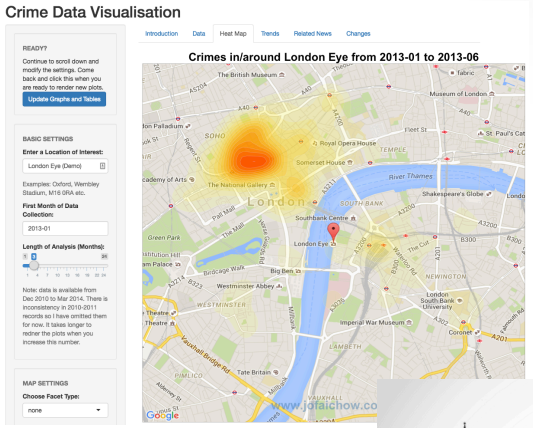
17 out of 717 teams (≈ top 2%)

# Conclusions

- Many R packages for predictive modelling.
- Use hyper-parameters tuning to improve individual models.
- Use model averaging / stacking to improve predictions.
- Trade-off between model performance and computational costs.
- Use R + H2O for scalable machine learning.
- H2O random grid search and stacking.
- Use data science for social good 👍

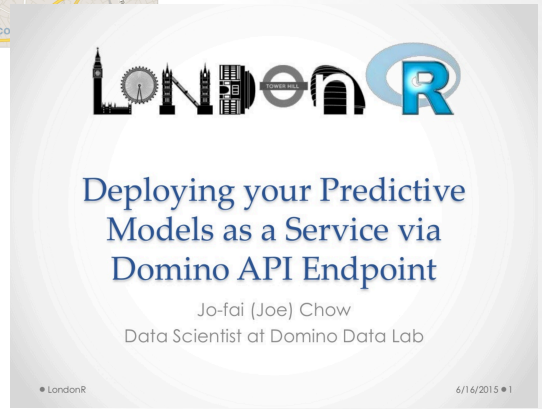
# Big Thank You!

- Mango Solutions
- RStudio
- Domino Data Lab
- H2O
  - Erin LeDell
  - Raymond Peck
  - Arno Candel




1st LondonR Talk
Crime Map Shiny App
bit.ly/londonr_crimemap

2nd LondonR Talk
Domino API Endpoint
bit.ly/1cYbZbF



# Any Questions?

- Contact
  - joe@h2o.ai
  - @matlabulous
  - github.com/woobe
- Slides & Code
  - github.com/h2oai/h2o-meetups
- H2O in London
  - Meetups / Office (soon)
  - www.h2o.ai/careers
- More H2O at Strata tomorrow
  - Innards of H2O (11:15)
  - Intro to Generalised Low-Rank Models (14:05)

# Extra Slide (Stratified Sampling)

## Raw Target (y)

```
smry_target_raw <- data.frame(table(y_train$OutcomeType))
smry_target_raw$Percentage <- smry_target_raw$Freq / length(y_train$OutcomeType)
smry_target_raw$Percentage <- round(smry_target_raw$Percentage, 4)
print(smry_target_raw)
```

```
##              Var1  Freq Percentage
## 1        Adoption 10769     0.4029
## 2            Died   197     0.0074
## 3      Euthanasia  1555     0.0582
## 4 Return_to_owner  4785     0.1790
## 5        Transfer  9422     0.3525
```

## Target by Fold

```
for (n_fold in 1:5) {
  cat("Summary of fold", n_fold, "\n")
  smry_target_fold <- data.frame(table(y_train[which(y_train$fold == n_fold),]$OutcomeType))
  smry_target_fold$Percentage <- smry_target_fold$Freq / length(y_train[which(y_train$fold == n_fold),]$OutcomeTy
pe)
  smry_target_fold$Percentage <- round(smry_target_fold$Percentage, 4)
  print(smry_target_fold)
  cat("\n")
}
```

```
## Summary of fold 1
##              Var1 Freq Percentage
## 1        Adoption 2154     0.4028
## 2            Died   40     0.0075
## 3      Euthanasia  311     0.0582
## 4 Return_to_owner  957     0.1790
## 5        Transfer 1885     0.3525
```