# Allstate Data Report

Ayush Mishra

Liping Li

# CONTENTS

01

## Preparation

Data overview
Some Preprocessing

02

## Construction

h2o deep learning : : neural network
Xgboost : : gradient boost tree

03

## Compare Models

Comparison table

04

## Ensemble

Bagging
Ensemble of ensembles

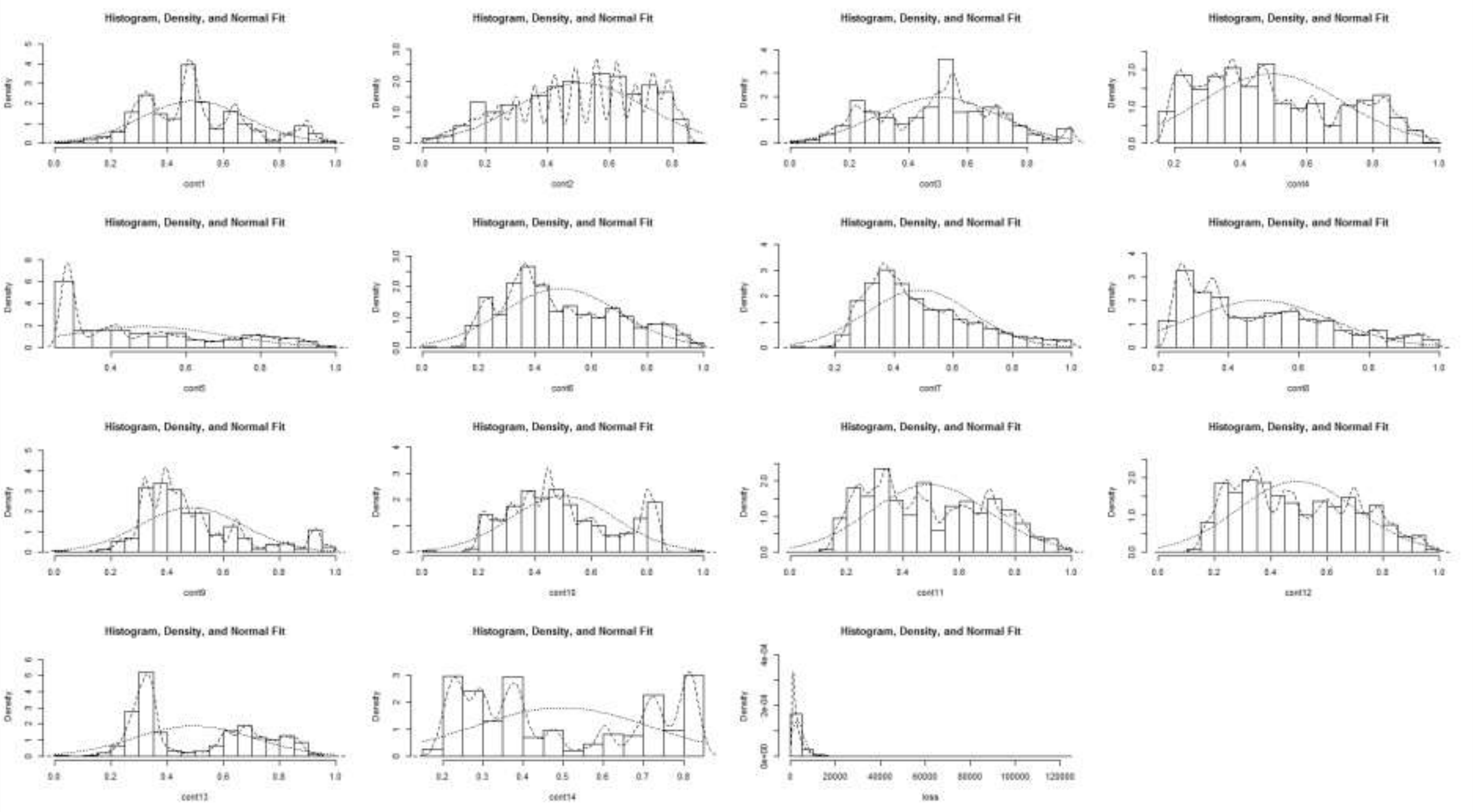
01

# Preparation

Data overview
Some preprocessing

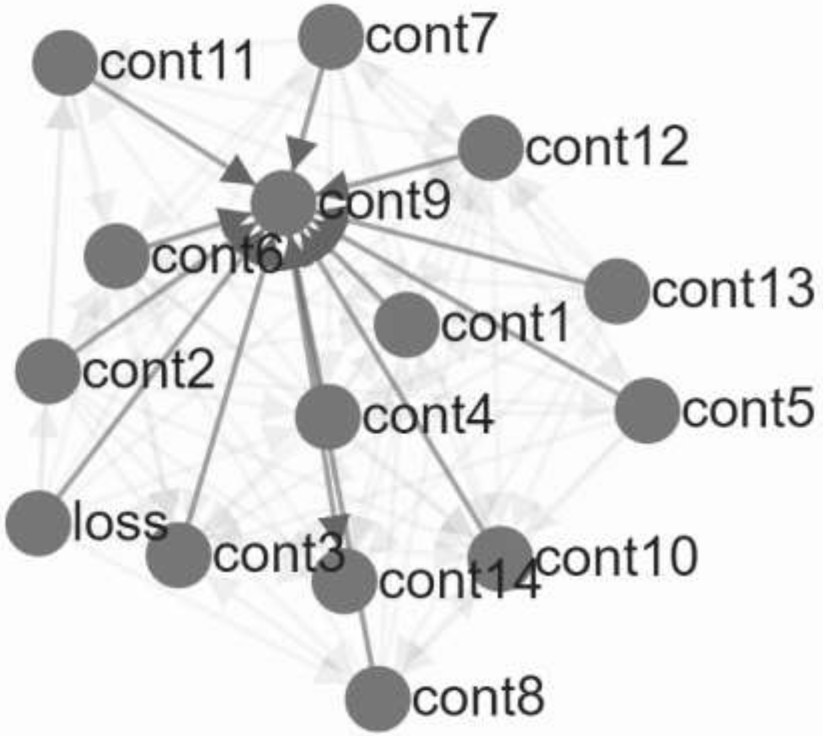
# Data Overview

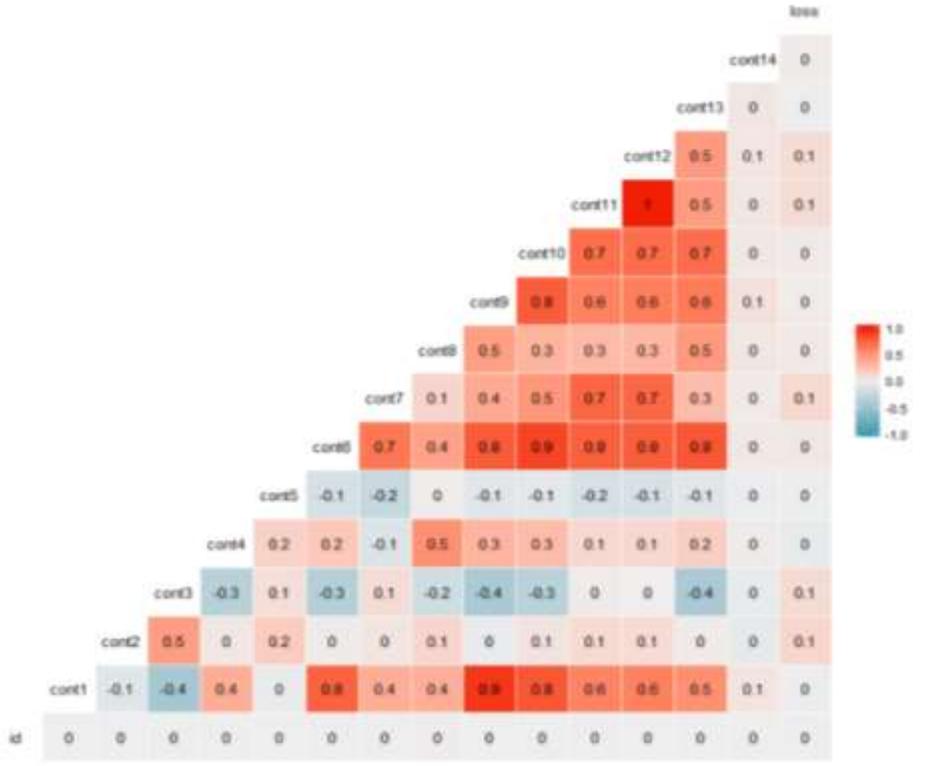
- Some continuous variables are not normally distributed.



# Data Overview

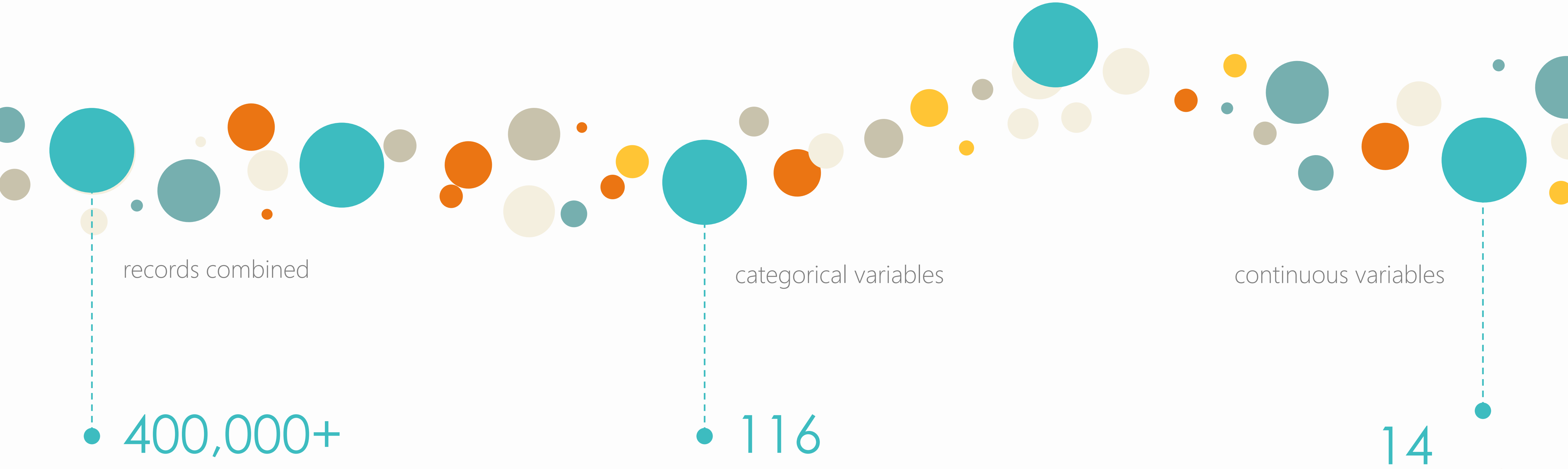
- There are many high correlations among variables.





# Data Overview

116 categorical variables combined with 14 continuous variables and over 400,000+ records together makes it impossible for common methods in R to compute.



records combined

400,000+

categorical variables

116

continuous variables

14

# Some Preprocessing

## Feature Selection

Variable Importance:

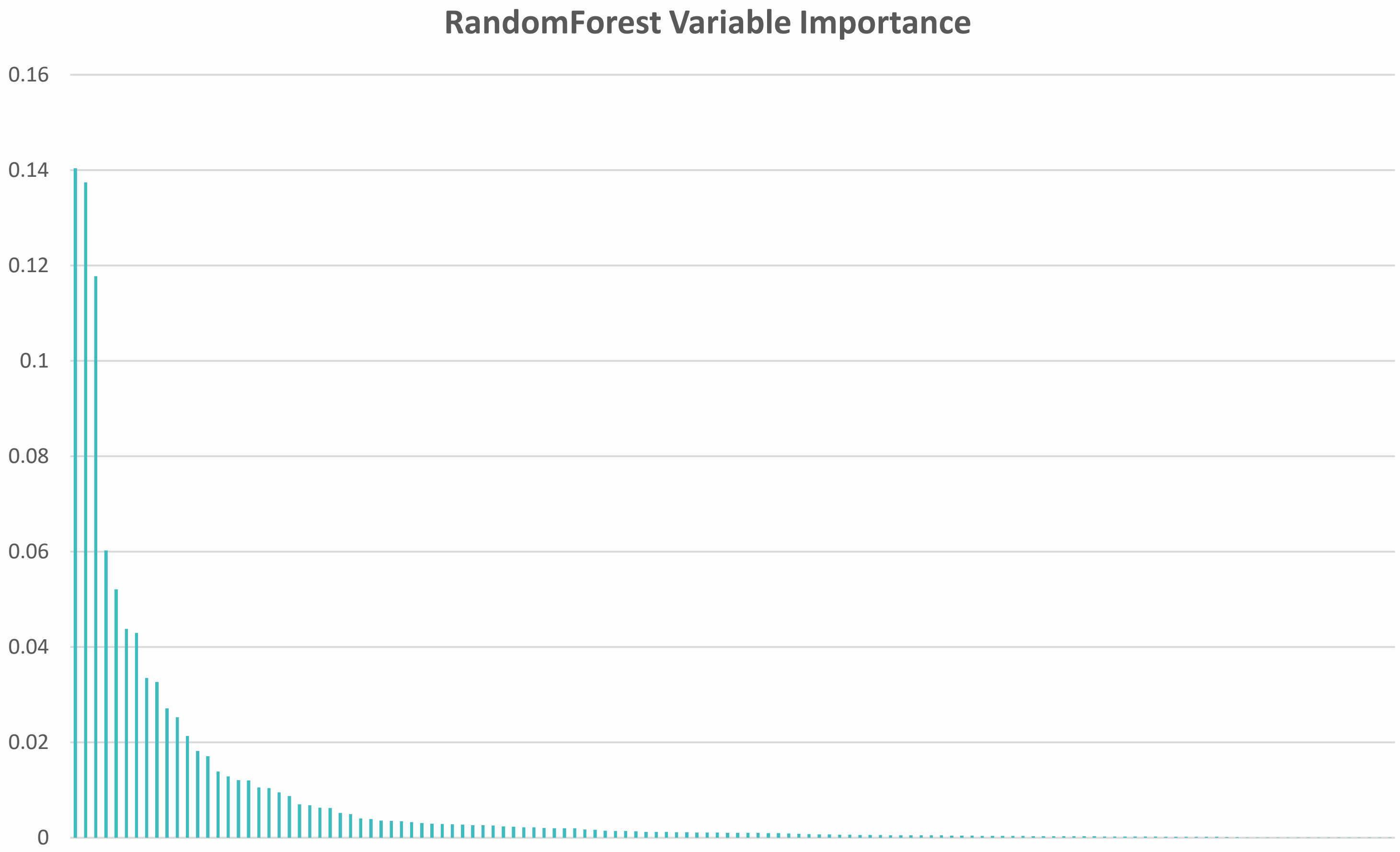
Random forest

Collinear Deduction:

Pearson correlation

&Chi Squared test of

independence.



# Some Preprocessing

- Categorical Variable Encoding

Sparse matrix

One-hot

Enum (embedded)

- Continuous Variable Transform

standardize, scale, etc(embedded)

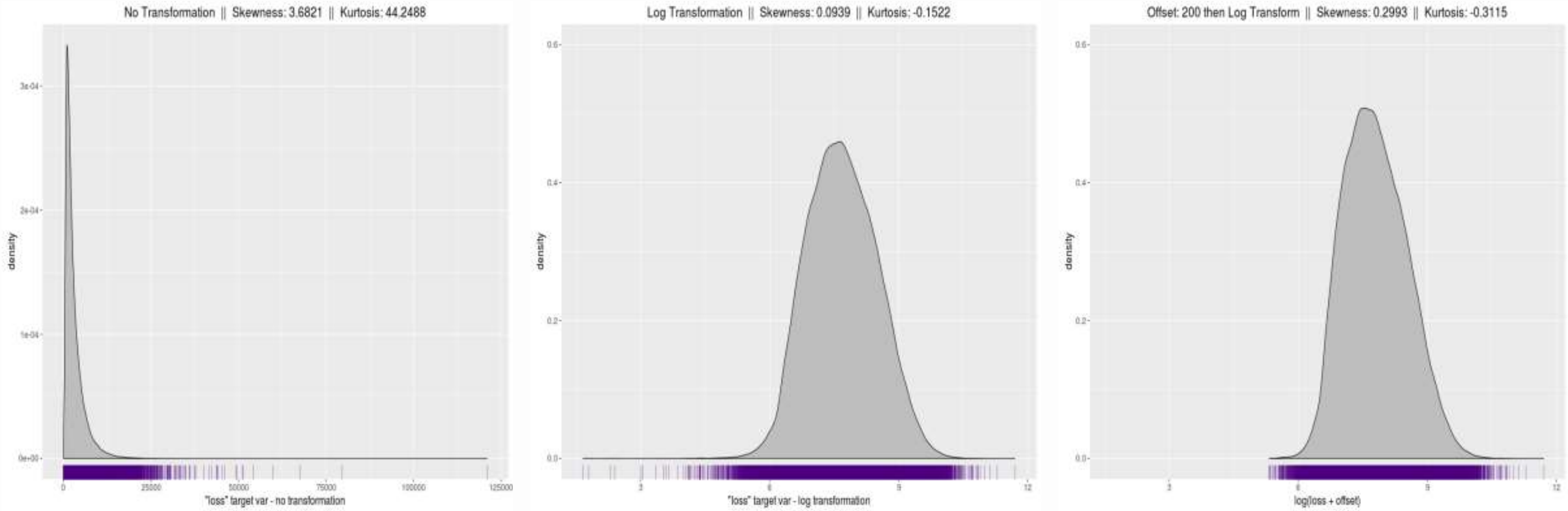
- Dependent Variable

Various distribution options(embedded)

Shift Introduction (Log Loss transformation)

# Some Preprocessing

## Shift Introduction (Log Loss transformation)



02

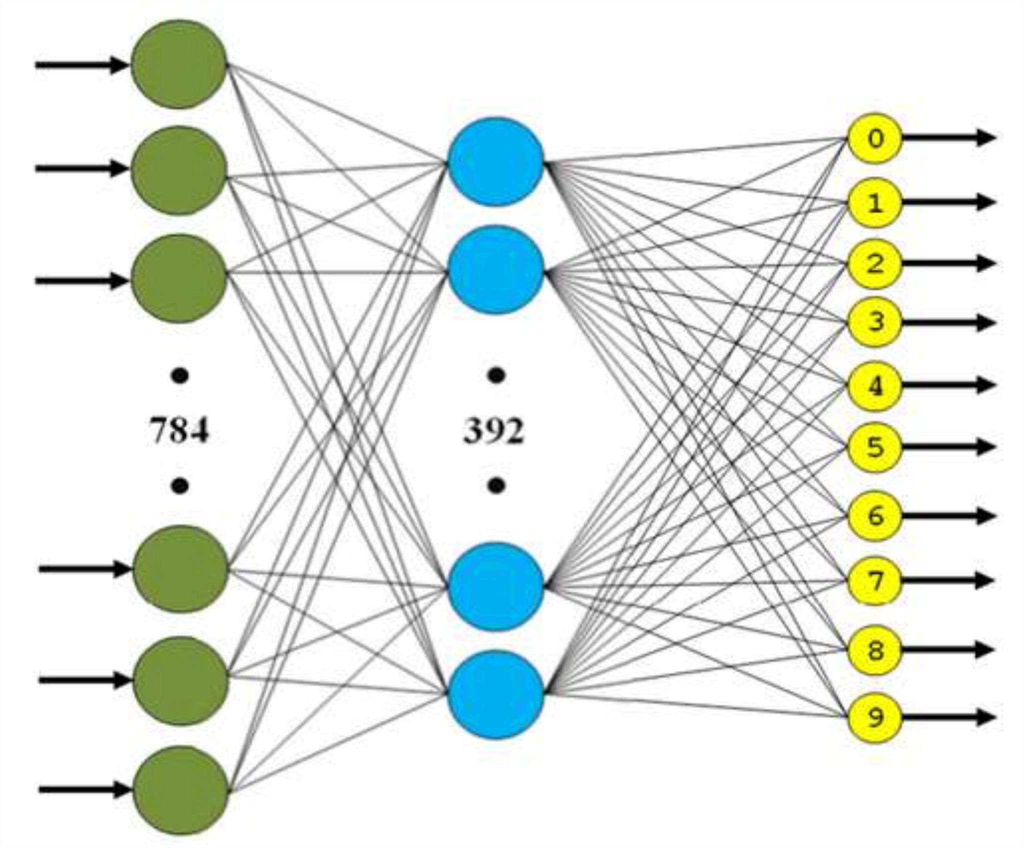
# Construction

h2o deep learning : : neural network
Xgboost : : gradient boost tree

# h2o deep learning : : neural network

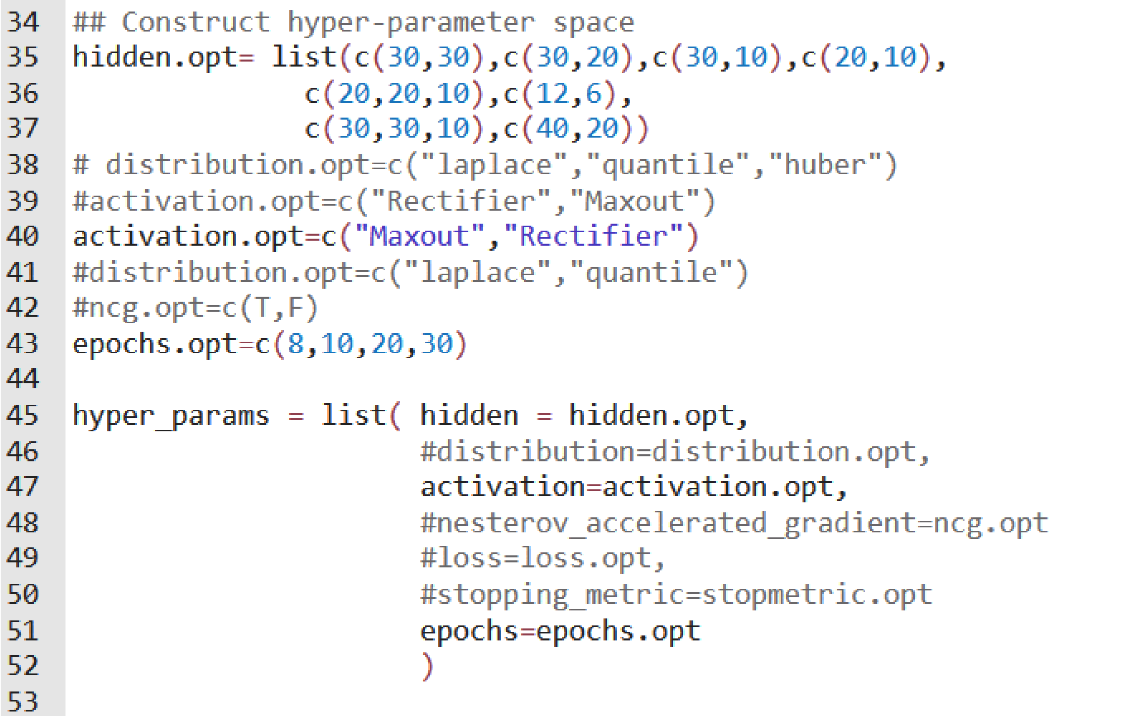
## Introduction

When the input layer receives an input it passes on a modified version of the input to the next layer. In a deep network, there are many layers between the input and output (and the layers are not made of neurons but it can help to think of it that way), allowing the algorithm to use multiple processing layers, composed of multiple linear and non-linear transformations.



# h2o deep learning : : neural network

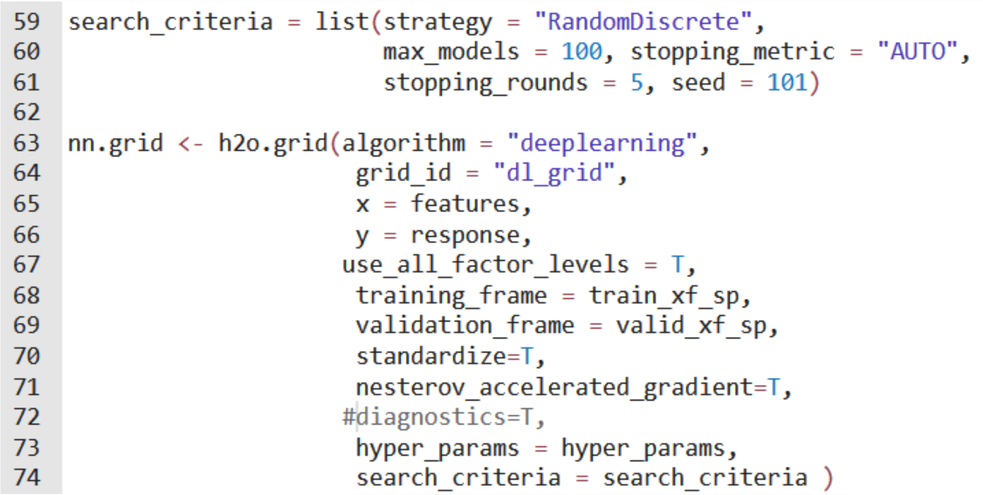
- A fast search tool: h2o.grid



```r
## Construct hyper-parameter space
hidden.opt= list(c(30,30),c(30,20),c(30,10),c(20,10),
              c(20,20,10),c(12,6),
              c(30,30,10),c(40,20))
# distribution.opt=c("laplace","quantile","huber")
#activation.opt=c("Rectifier","Maxout")
activation.opt=c("Maxout","Rectifier")
#distribution.opt=c("laplace","quantile")
#ncg.opt=c(T,F)
epochs.opt=c(8,10,20,30)

hyper_params = list( hidden = hidden.opt,
                     #distribution=distribution.opt,
                     activation=activation.opt,
                     #nesterov_accelerated_gradient=ncg.opt
                     #loss=loss.opt,
                     #stopping_metric=stopmetric.opt
                     epochs=epochs.opt
                     )

```

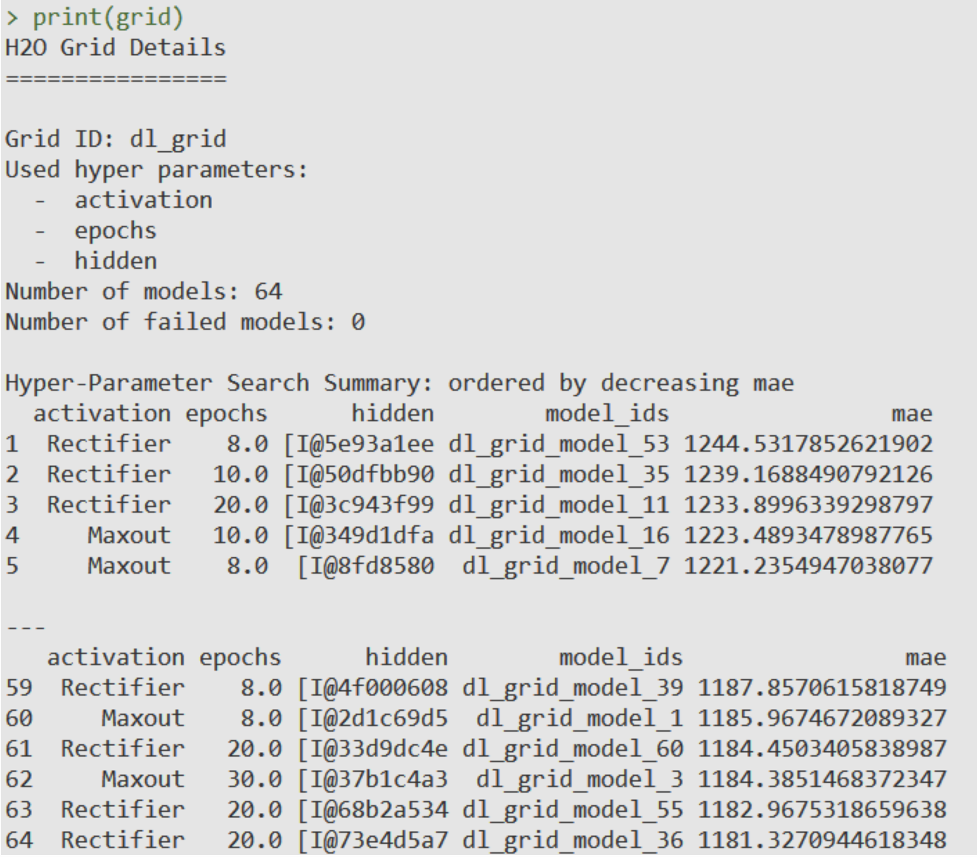


```r
search_criteria = list(strategy = "RandomDiscrete",
                       max_models = 100, stopping_metric = "AUTO",
                       stopping_rounds = 5, seed = 101)

nn.grid <- h2o.grid(algorithm = "deeplearning",
                     grid_id = "dl_grid",
                     x = features,
                     y = response,
                    use_all_factor_levels = T,
                     training_frame = train_xf_sp,
                     validation_frame = valid_xf_sp,
                     standardize=T,
                     nesterov_accelerated_gradient=T,
                    #diagnostics=T,
                     hyper_params = hyper_params,
                     search_criteria = search_criteria )
```

# h2o deep learning : : neural network

- A fast search tool: h2o.grid



```
> print(grid)
H2O Grid Details
================

Grid ID: dl_grid
Used hyper parameters:
  -  activation
  -  epochs
  -  hidden
Number of models: 64
Number of failed models: 0

Hyper-Parameter Search Summary: ordered by decreasing mae
  activation epochs      hidden         model_ids              mae
1  Rectifier    8.0 [I@5e93a1ee dl_grid_model_53 1244.5317852621902
2  Rectifier   10.0 [I@50dfbb90 dl_grid_model_35 1239.1688490792126
3  Rectifier   20.0 [I@3c943f99 dl_grid_model_11 1233.8996339298797
4     Maxout   10.0 [I@349d1dfa dl_grid_model_16 1223.4893478987765
5     Maxout    8.0  [I@8fd8580  dl_grid_model_7 1221.2354947038077

---
   activation epochs      hidden         model_ids              mae
59  Rectifier    8.0 [I@4f000608 dl_grid_model_39 1187.8570615818749
60     Maxout    8.0 [I@2d1c69d5  dl_grid_model_1 1185.9674672089327
61  Rectifier   20.0 [I@33d9dc4e dl_grid_model_60 1184.4503405838987
62     Maxout   30.0 [I@37b1c4a3  dl_grid_model_3 1184.3851468372347
63  Rectifier   20.0 [I@68b2a534 dl_grid_model_55 1182.9675318659638
64  Rectifier   20.0 [I@73e4d5a7 dl_grid_model_36 1181.3270944618348
```

These models can be sorted by validation index, like “mae”, “rmse” and so on. But I consider both validation MAE and train MAE.

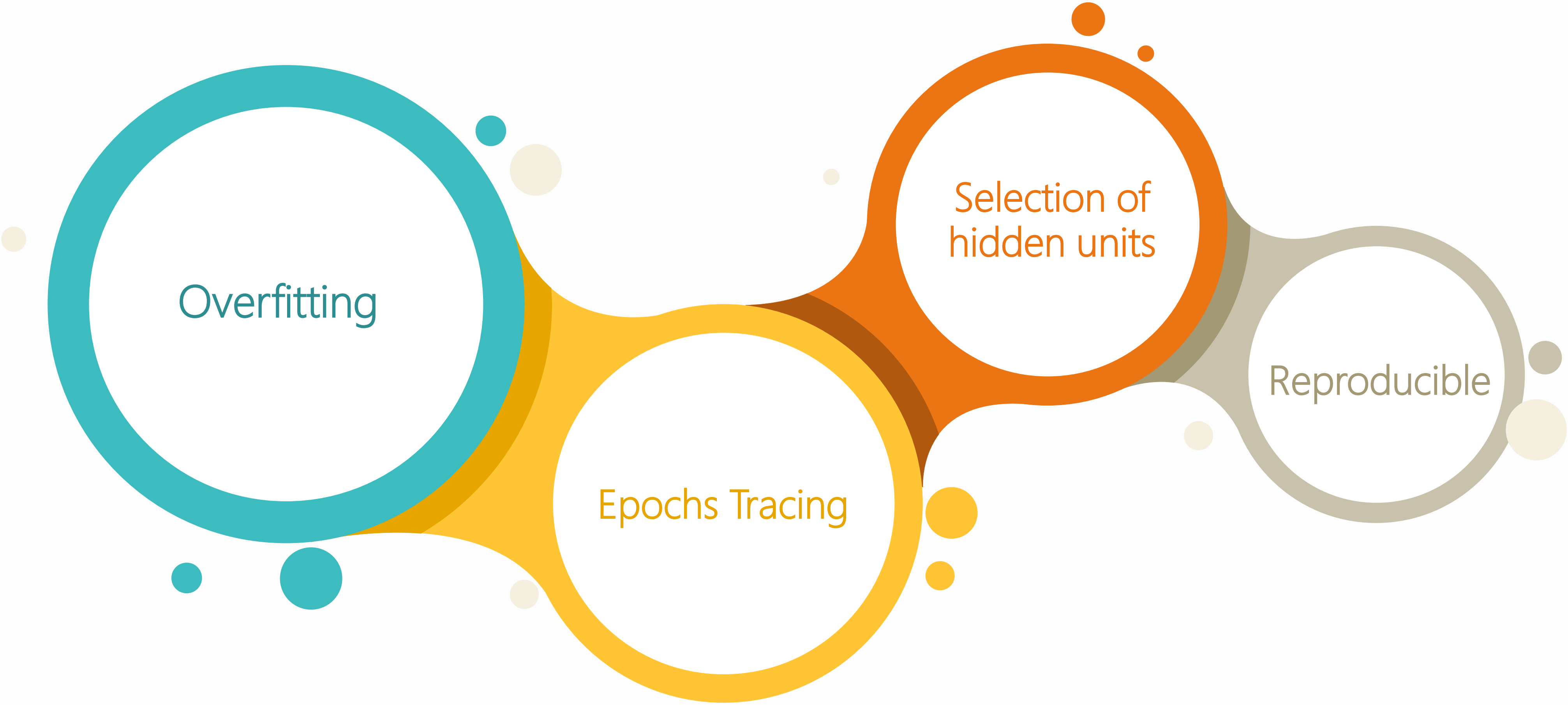
# h2o deep learning : : neural network

## Other parameters

| | |
|---|---|
| **use_all_factor_levels** | Use all the categorical variable levels, default encoding as enum |
| **standardize** | Standardize continuous variable. Though I feel continuous variable already be standardized, this procedure still has slight impact |
| **activation option** | Rectifier seems to be faster.<br>Maxout seems to work better with less hidden nodes setting. |
| **epochs** | How many times the dataset should be iterated. |
| **distribution option** | Laplace is double exponential distribution. Money issues often follow exponential distribution.<br>Quantile distribution is to cut groups in terms of quantiles. In general, both Laplace and quantile distribution get better result in deep learning models compared to Gaussian. |
| **Nesterov accelerated gradient** | Adjust momentum automatically |

# h2o deep learning : : neural network

- Limitation

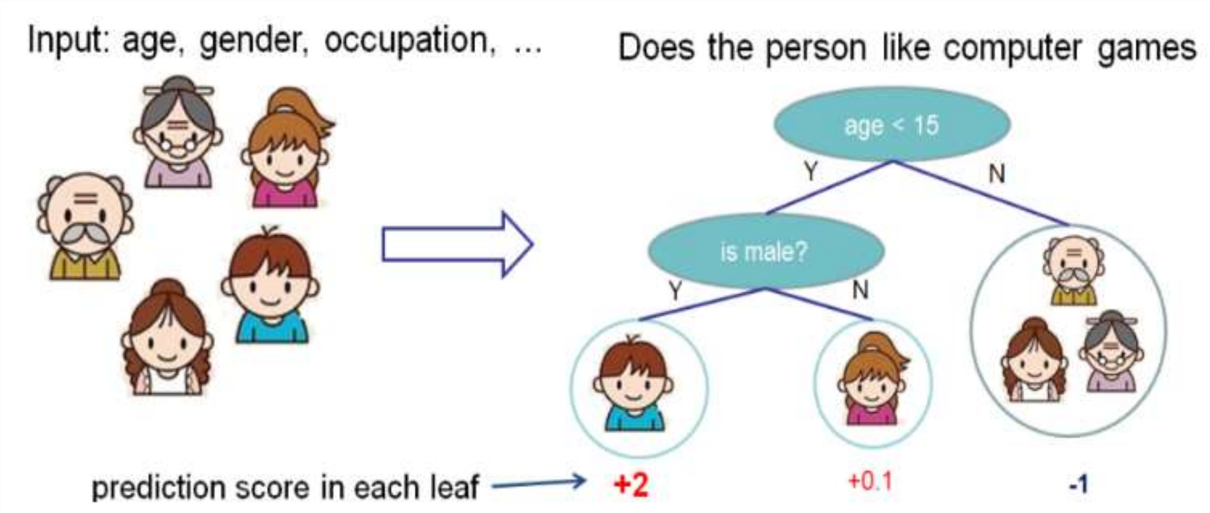


# Xgboost : : gradient boost tree

## Introduction

Uses Greedy Function Approximation, as proposed by Friedman.

Tree based ensemble which uses Classification and Regression Trees (CART)


Input: age, gender, occupation, ...
Does the person like computer games
age < 15
Y
N
is male?
Y
N
prediction score in each leaf
+2
+0.1
-1

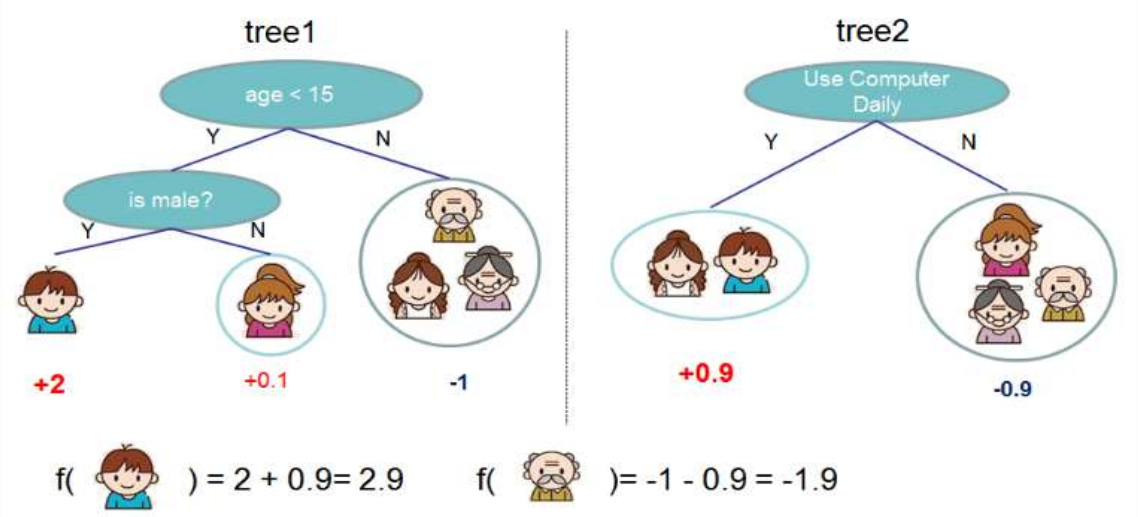
- d-tree?
- Prediction Score of leaf
- Single tree enough?

# Xgboost : : gradient boost tree

## Introduction

Usually a single tree isn't enough for practical purposes.

Tree ensemble that has the capability to sum prediction results of multiple trees:



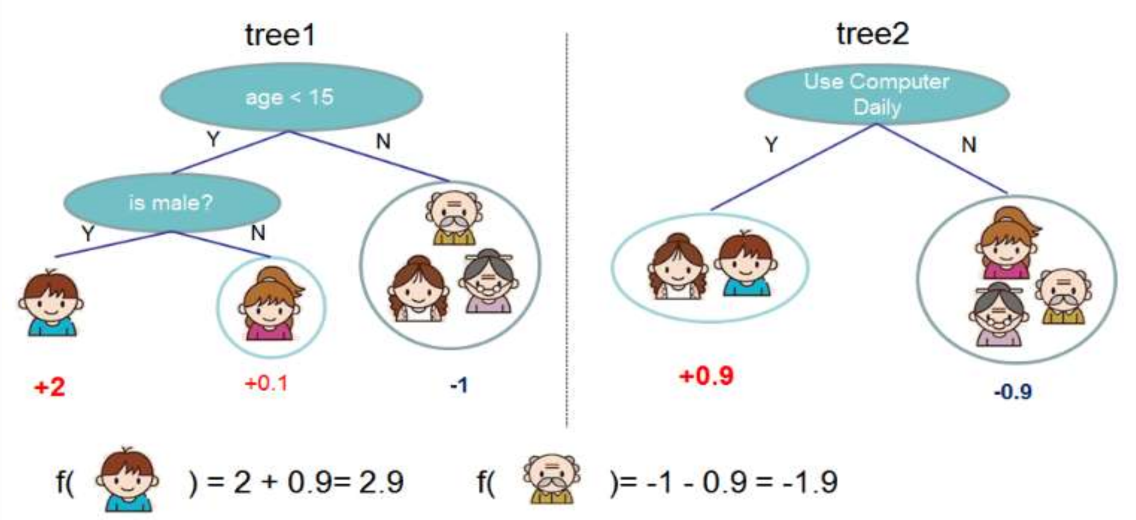
Prediction score of each individual leaf, however it occurs i.e., either with an individual score or in a combined form, is summed up.

# Xgboost : : gradient boost tree

## Introduction

Usually a single tree isn’t enough for practical purposes.

Tree ensemble that has the capability to sum prediction results of multiple trees:



Prediction score of each individual leaf, however it occurs i.e., either with an individual score or in a combined form, is summed up.

# Xgboost : : gradient boost tree

The important fact, two trees complement each other, in other words, enhance a positive result and reduce negative scores even further.

Intuitively we can see how this evolves as the number of trees increase.

The idea is to build as many tree models as possible and combine them together. Sounds a bit Random Forest? Doesn't it.

In fact, the objective function used in this model is exactly the same as that of Random Forests.
Difference? How do you train. Advantage of Supervised learning objective functions

# Xgboost : : gradient boost tree

## How we fared with initial models?

| | | | | |
|---|---|---|---|---|
| **Gradient boosted trees** | **XGboost** | **All** | **seed = 0,<br>colsample_bytree = 0.7,<br>subsample = 0.7,<br>eta = 0.075,<br>objective = 'reg:linear',<br>max_depth = 6,<br>num_parallel_tree = 1,<br>min_child_weight = 1,<br>base_score = 7** | **1149** |
| **Gradient boosted trees** | XGboost | Reduced with Chi Squared test. Down to almost half of the features | seed = 0,<br>colsample_bytree = 0.7,<br>subsample = 0.7,<br>eta = 0.075,<br>objective = 'reg:linear',<br>max_depth = 6,<br>num_parallel_tree = 1,<br>min_child_weight = 1,<br>base_score = 7 | 1183 |

# Xgboost : : gradient boost tree

## How we fared with initial models?

| | |
|---|---|
| **colsample_bytree** | Subsamples ratio of columns when constructing each tree |
| **subsample** | Subsample ratio of the training instance. Setting it to 0.5 means that xgboost randomly collected half of the data instances to grow trees and this prevents overfitting. We use it as = 0.8, since we already subsample the columns to half the size. |
| **eta** | Control the learning rate: scale the contribution of each tree by a factor of 0 < eta < 1 when it is added to the current approximation. Used to prevent overfitting by making the boosting process more conservative |
| **objective** | Specify the learning task and the corresponding learning objective. “reg:linear” - linear regression |
| **max_depth** | maximum depth of a tree |
| **alpha** | L1 regularization term on weights. (there is no L1 reg on bias because it is not important) |
| **gamma** | minimum loss reduction required to make a further partition on a leaf node of the tree. |
| **min_child_weight** | minimum sum of instance weight(hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. |
| **base_score** | the initial prediction score of all instances, global bias. |

# Xgboost : : gradient boost tree

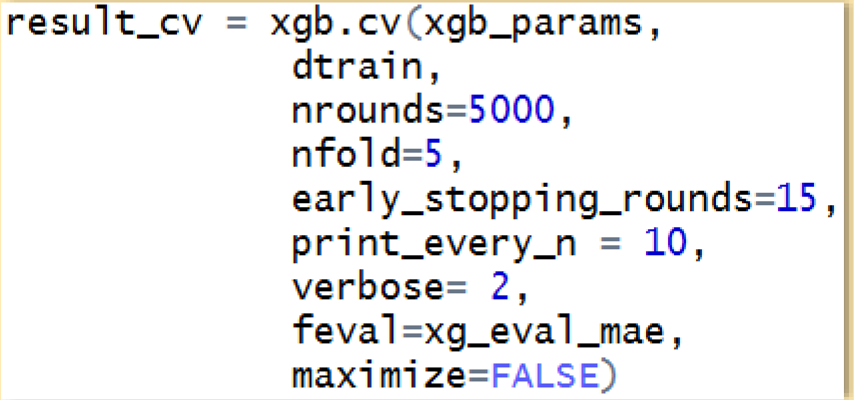
Parameter optimization with shift: how we fared?

| | | | | |
|---|---|---|---|---|
| Gradient boosted trees | XGboost | All<br>(shift = 200) | seed = 0,<br>colsample_bytree = 0.5,<br>subsample = 0.8,<br>eta = 0.01,<br>objective = 'reg:linear',<br>max_depth = 12,<br>alpha = 1,<br>gamma = 2,<br>min_child_weight = 1,<br>base_score = 7.76 | 1133 |
| Gradient boosted trees | XGboost | Chi Sq reduced dataset.<br>(shift = 200) | seed = 0,<br>colsample_bytree = 0.5,<br>subsample = 0.8,<br>eta = 0.01,<br>objective = 'reg:linear',<br>max_depth = 12,<br>alpha = 1,<br>gamma = 2,<br>min_child_weight = 1,<br>base_score = 7.76 | 1150 |
| Gradient boosted trees | XGboost | Random Forest importance reduced dataset (shift = 200) | seed = 0,<br>colsample_bytree = 0.5,<br>subsample = 0.8,<br>eta = 0.01,<br>objective = 'reg:linear',<br>max_depth = 12,<br>alpha = 1,<br>gamma = 2,<br>min_child_weight = 1,<br>base_score = 7.76 | 1135 |

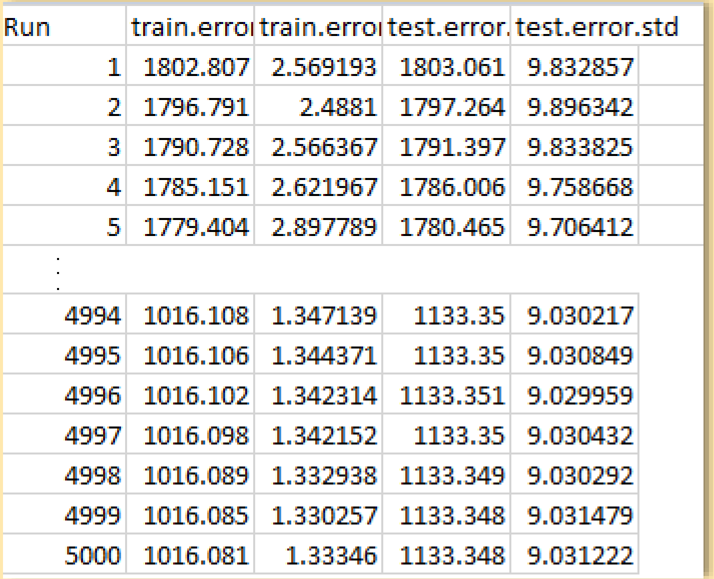
# Xgboost : : gradient boost tree

We could see here the difference parameter optimization brings about in terms of model performance

We used around 5000 rounds of cross validation to determine the best performing models



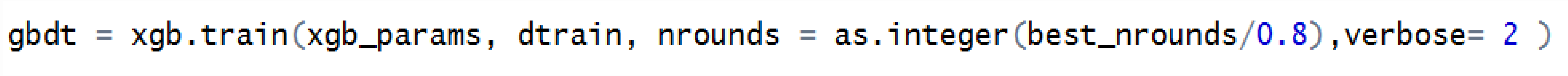
```
result_cv = xgb.cv(xgb_params,
             dtrain,
             nrounds=5000,
             nfold=5,
             early_stopping_rounds=15,
             print_every_n = 10,
             verbose= 2,
             feval=xg_eval_mae,
             maximize=FALSE)
```



| Run | train.erro | train.erro | test.error. | test.error.std |
|---|---|---|---|---|
| 1 | 1802.807 | 2.569193 | 1803.061 | 9.832857 |
| 2 | 1796.791 | 2.4881 | 1797.264 | 9.896342 |
| 3 | 1790.728 | 2.566367 | 1791.397 | 9.833825 |
| 4 | 1785.151 | 2.621967 | 1786.006 | 9.758668 |
| 5 | 1779.404 | 2.897789 | 1780.465 | 9.706412 |
| ⋮ | | | | |
| 4994 | 1016.108 | 1.347139 | 1133.35 | 9.030217 |
| 4995 | 1016.106 | 1.344371 | 1133.35 | 9.030849 |
| 4996 | 1016.102 | 1.342314 | 1133.351 | 9.029959 |
| 4997 | 1016.098 | 1.342152 | 1133.35 | 9.030432 |
| 4998 | 1016.089 | 1.332938 | 1133.349 | 9.030292 |
| 4999 | 1016.085 | 1.330257 | 1133.348 | 9.031479 |
| 5000 | 1016.081 | 1.33346 | 1133.348 | 9.031222 |

# Xgboost : : gradient boost tree

Out of these results we selected the best performing round of the cross validation and run the model again with same parameters only, this time, we ran it on the test data to get final scores:



```
gbdt = xgb.train(xgb_params, dtrain, nrounds = as.integer(best_nrounds/0.8),verbose= 2 )
```

We used 80% of the number of rounds to prevent overfit. This gave us our best performing models at MAE 1133.

Is this enough? – Further rounds led to not much improvement in the individual performance – Two ways to proceed

03

# Compare Models

# Compare Models

| Model | Package | data/features | Parameter | MAE |
|---|---|---|---|---|
| Gradient Boosted Model | h2o | all | default | 1214 |
| NeuralNet | h2o | all | use_all_factor_levels = T,<br>standardize=T,distribution = 'laplace',<br>hidden=c(10,5),epochs=5,<br>diagnostics=T | 1160 |
| NeuralNet | h2o | first half important variables | use_all_factor_level<br>standardize=T,distrib<br>hidden=c(12,6),epoc<br>diagnostics=T | |
| NeuralNet | h2o | first half important variables | use_all_factor_level<br>standardize=T,<br>activation = "Maxou<br>distribution = "laplac<br>hidden=c(12,6),<br>epochs=16,<br>nesterov_accelerate<br>seed=101, | |
| NeuralNet | h2o | first half important variables & logged "loss" | use_all_factor_levels = T,<br>standardize=T,<br>hidden=c(14,7),epochs=5,<br>diagnostics=T | 110 |
| Gradient boosted trees | XGboost | All | seed = 0,<br>colsample_bytree = 0.7,<br>subsample = 0.7,<br>eta = 0.075,<br>objective = 'reg:linear',<br>max_depth = 6,<br>num_parallel_tree = 1,<br>min_child_weight = 1,<br>base_score = 7 | 1149 |

| Model | Package | data/features | Parameter | MAE |
|---|---|---|---|---|
| Gradient boosted trees | XGboost | Reduced with Chi Squared test. Down to almost half of the features | seed = 0,<br>colsample_bytree = 0.7,<br>subsample = 0.7,<br>eta = 0.075,<br>objective = 'reg:linear',<br>max_depth = 6,<br>num_parallel_tree = 1,<br>min_child_weight = 1,<br>base_score = 7 | 1183 |
| | | All<br>(shift = 200) | seed = 0,<br>colsample_bytree = 0.5,<br>subsample = 0.8,<br>eta = 0.01,<br>objective = 'reg:linear',<br>max_depth = 12,<br>alpha = 1,<br>gamma = 2,<br>min_child_weight = 1,<br>base_score = 7.76 | 1133 |
| | | Chi Sq reduced dataset.<br>(shift = 200) | seed = 0,<br>colsample_bytree = 0.5,<br>subsample = 0.8,<br>eta = 0.01,<br>objective = 'reg:linear',<br>max_depth = 12,<br>alpha = 1,<br>gamma = 2,<br>min_child_weight = 1,<br>base_score = 7.76 | 1150 |
| Gradient boosted trees | XGboost | Random Forest importance reduced dataset (shift = 200) | seed = 0,<br>colsample_bytree = 0.5,<br>subsample = 0.8,<br>eta = 0.01,<br>objective = 'reg:linear',<br>max_depth = 12,<br>alpha = 1,<br>gamma = 2, | 1135 |

# Thinking

## Two ways

Either choose to filter the feature selection in order to improve individual models.

Options:

Exp transformation of continuous data

Exp(SQRT) transformation of continuous data

Categorical Correlation

Categorical + Continuous importance ranks (RF importance)

We tried almost all of the above options but with little improvement in our models.

The models always seemed to be losing some of the information when we reduced the features indicating direct correlation identification is not a good option to eliminate unnecessary features.

Ensemble Methods – Where the magic happens!

04

# Ensemble

Bagging

Ensemble of ensembles

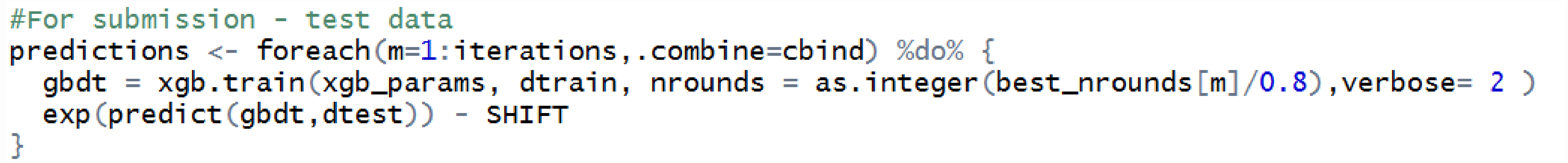
# Bagging

## Initiate

We decided to collectively enhance the results by creating ensemble of the data. We proceeded with simple averaging of the results which is the method found in “bagging”

How? – Remember our CV rounds of 5000? We selected a few best rounds of Train MAE performance i.e. lowest MAE obv…

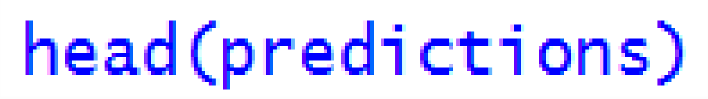
We built a prediction data frame which accrued Prediction result of each round in the iteration into one column of the prediction df:



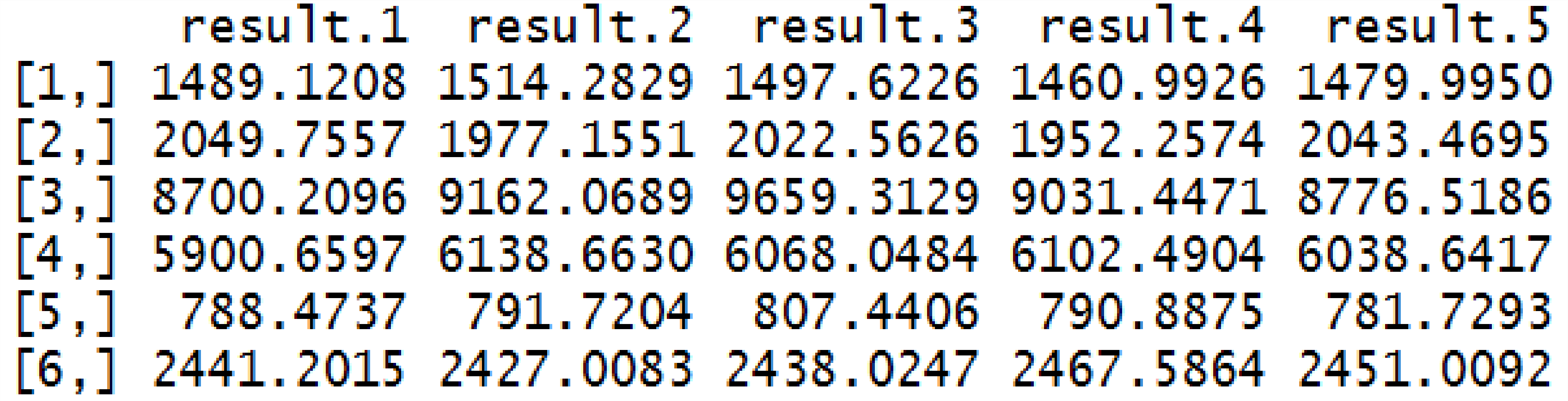
```
#For submission - test data
predictions <- foreach(m=1:iterations,.combine=cbind) %do% {
  gbdt = xgb.train(xgb_params, dtrain, nrounds = as.integer(best_nrounds[m]/0.8),verbose= 2 )
  exp(predict(gbdt,dtest)) - SHIFT
}
```

# Bagging

This Prediction data frame bore the results of all iterations into each of its columns:



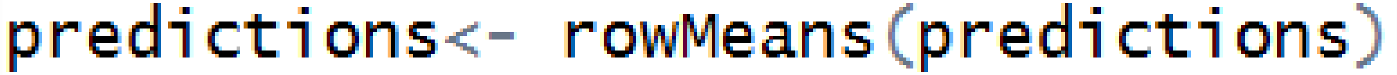
```
head(predictions)
```



```
       result.1  result.2  result.3  result.4  result.5
[1,] 1489.1208 1514.2829 1497.6226 1460.9926 1479.9950
[2,] 2049.7557 1977.1551 2022.5626 1952.2574 2043.4695
[3,] 8700.2096 9162.0689 9659.3129 9031.4471 8776.5186
[4,] 5900.6597 6138.6630 6068.0484 6102.4904 6038.6417
[5,]  788.4737  791.7204  807.4406  790.8875  781.7293
[6,] 2441.2015 2427.0083 2438.0247 2467.5864 2451.0092
```

# Bagging

We now calculated the mean of each row and stored the result as the ensemble of all five columns data:



```
predictions<- rowMeans(predictions)
```

This gave us the consolidated ensemble of all the five xgb models used in our ensemble.

This results were staggeringly improved from the individual models that we built. The very first ensemble gave a combined MAE of 1117 on Train error which was a significant improvement from a consistent individual MAE of 1130-1150 range.

# Ensemble of ensembles

The next Ensemble was the Neural Net Ensemble. The average of the four Neural Net models (mentioned in report) resulted in a Train MAE of 315 which indicated that the model had almost fully learned the Training data which indicated overfitting.

Our intuition was confirmed with a test error of around 1136 on Kaggle.

We wanted a method to deal with this situation, not rejecting and still somewhat improving model performance:

What we did? - Ensemble of Ensembles!!!

# Ensemble of ensembles

We derived at a novel way to “stack” the models together. But how?

We chose some of the not so good performing models, XGB range 1130-1150 and fully learned NN ensemble

Ensembled the poor performing XGBoost models using bagging

Now we used the method of a weighted ensemble:

$$(A*x+B*y)/(x+y)$$

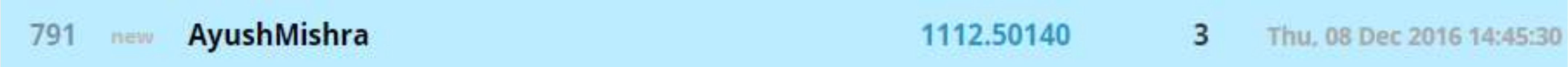
This allowed us to combine the ensembles as a weighted ratio. (x,y)

So a higher x:y ratio indicated more influence of ensemble A on the results and a higher y:x indicated the opposite.

# Ensemble of ensembles

This model too, to our surprise returned a performance of TRAIN MAE 1119 which is way higher than expected results of the individual models.

Thus, Ensemble methods proved to be the best option to go with when it came to optimizing results.
Finally, after using 10000 rounds of cross validation and using a pure XGB ensemble gave us TEST Score of 1112.5 on Kaggle which is our best performing model yet:



| | | | | | |
|---|---|---|---|---|---|
| 791 | new | AyushMishra | 1112.50140 | 3 | Thu, 08 Dec 2016 14:45:30 |

# SUMMARY

- **Better individual model**

  In the future we hope to fine tune our models individually using data transformation techniques and use better feature selection methods that would result in an even better solution when ensembled together.

- **Power of ensemble**

  We realized the predictive potential of ensembling even if the models we built were moderately good.

# THANK U