



# R을 통한 Multiprocessing

Foreach와 h2o

김영석



# CONTENT

## 1. 병렬처리?

1-1. 병렬처리 란?

1-2. 병렬처리 실습

## 2. h2o 패키지

2-1. h2o 패키지 란?

2-2. h2o 실습

---

# 01

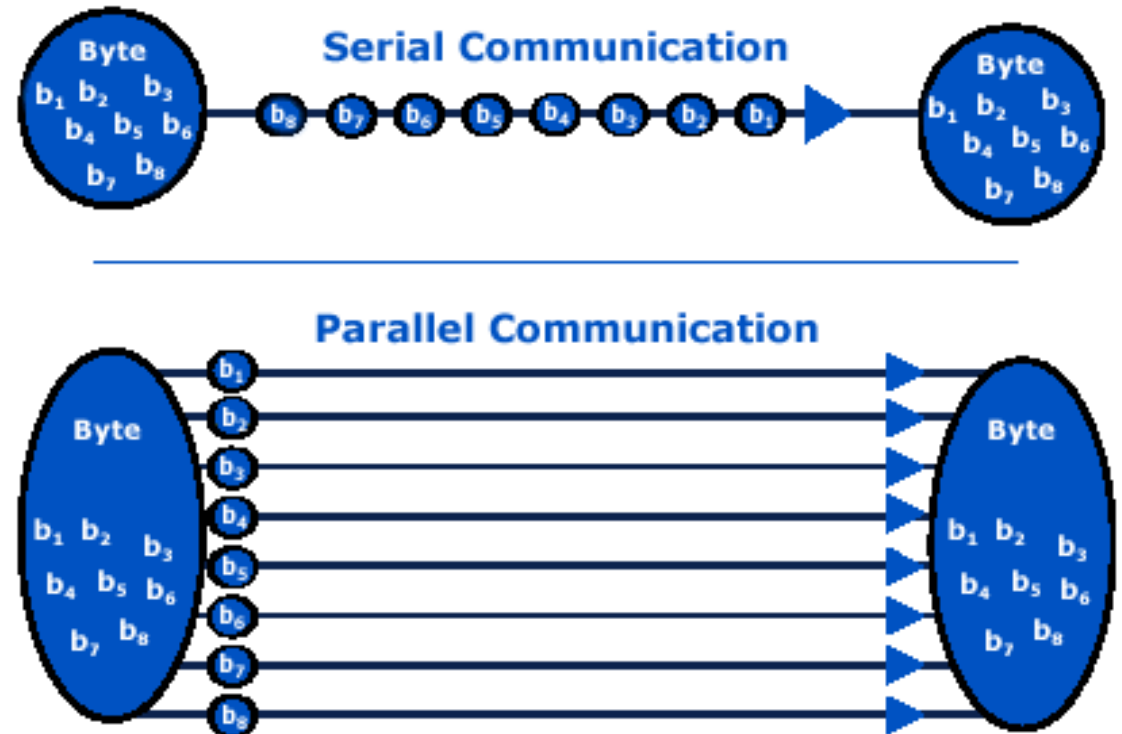
## 병렬처리

1-1. 병렬처리란?

1-2. 병렬처리 실습

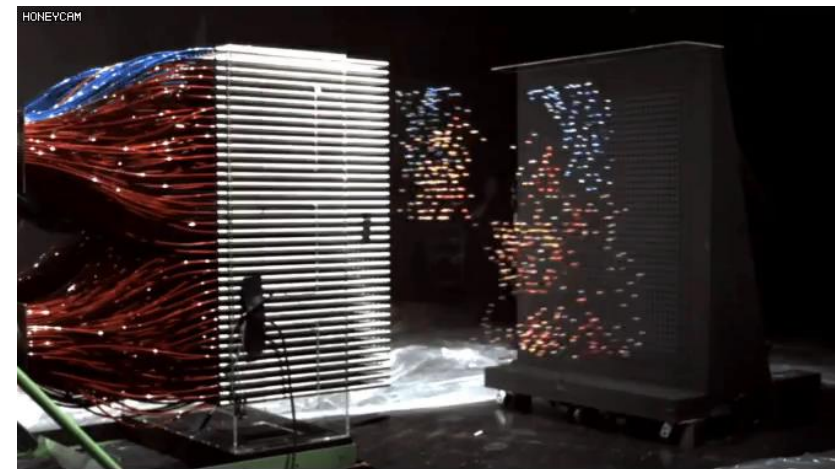
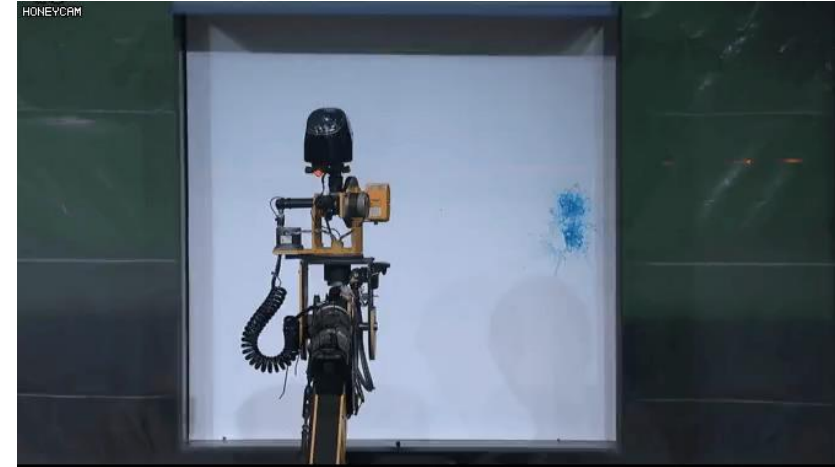
# 1-1 병렬처리(Parallel Processing) 란?

- 동시에 많은 계산을 하는 연산의 한 방법이다.
- 크고 복잡한 문제를 작게 나눠 동시에 병렬적으로 해결하는 데에 주로 사용
- CPU와 GPU에서 모두 병렬처리 가능
  - ✓ GPU가 더 빠른 성능을 보임 ( 현 시점 CPU는 코어 개수가 많아봐야 32개 정도이지만 GPU에는 수 백개의 코어가 있으므로 )

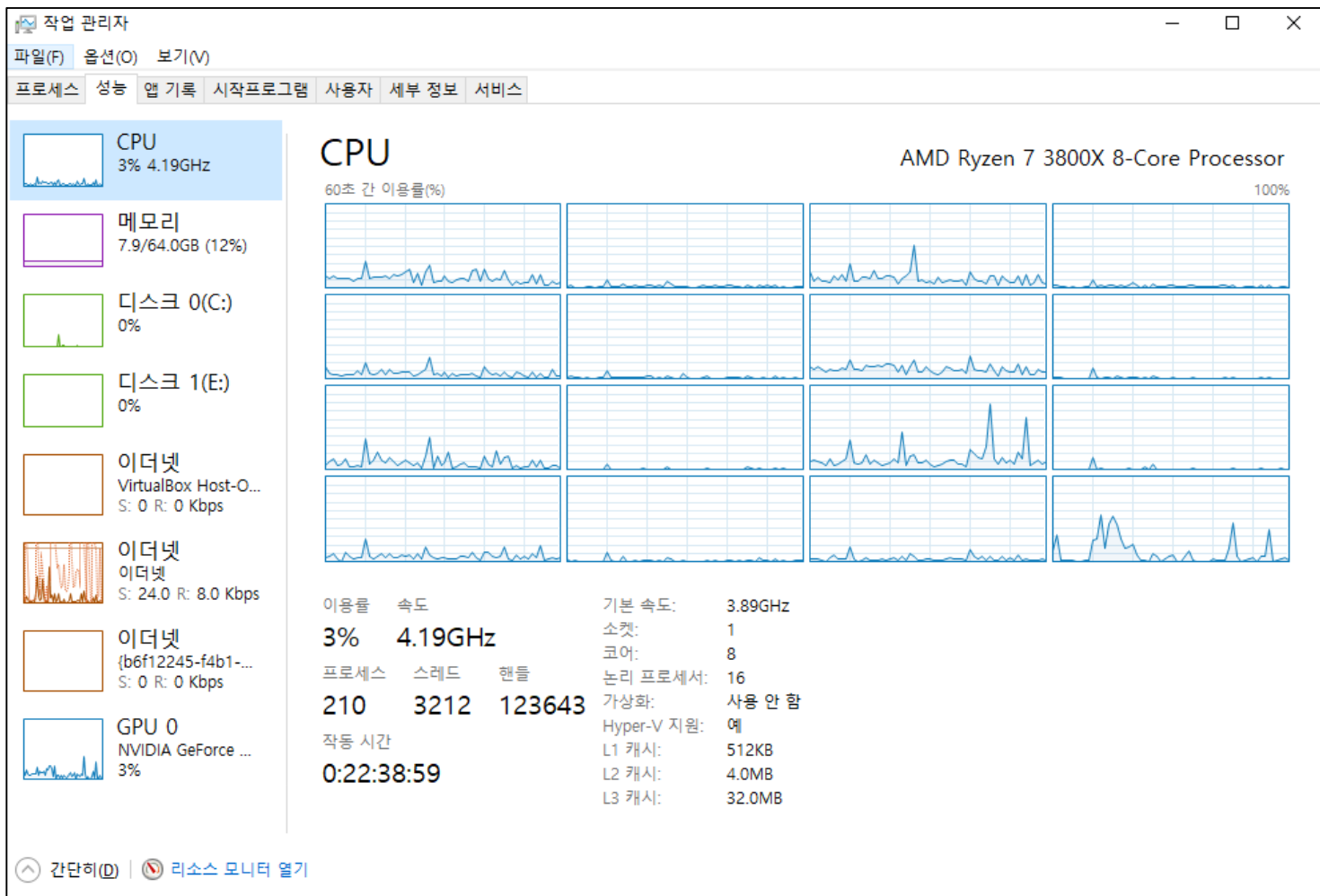


# 1-1 병렬처리(Parallel Processing) 란?

- 나뉘서 할 수 있는 연산의 수는 코어(혹은 쓰레드)의 수로 결정된다. (컴퓨팅 파워)
- CPU : 중앙처리장치라고하며 기본적으로 명령어의 해석 자료의 연산 비교등을 처리하는 핵심장치이다. 쉽게 말해 컴퓨터의 두뇌이다.
- GPU : 그래픽 처리 장치라고 하며 그래픽 카드 (VGA)를 구성하는 가장 중요한 요소. 이미지를 화면에 표현하는 역할을 함.
- 각자의 컴퓨터의 CPU, GPU를 간단하게 확인해보자.



# 1-1 병렬처리(Parallel Processing) 란?



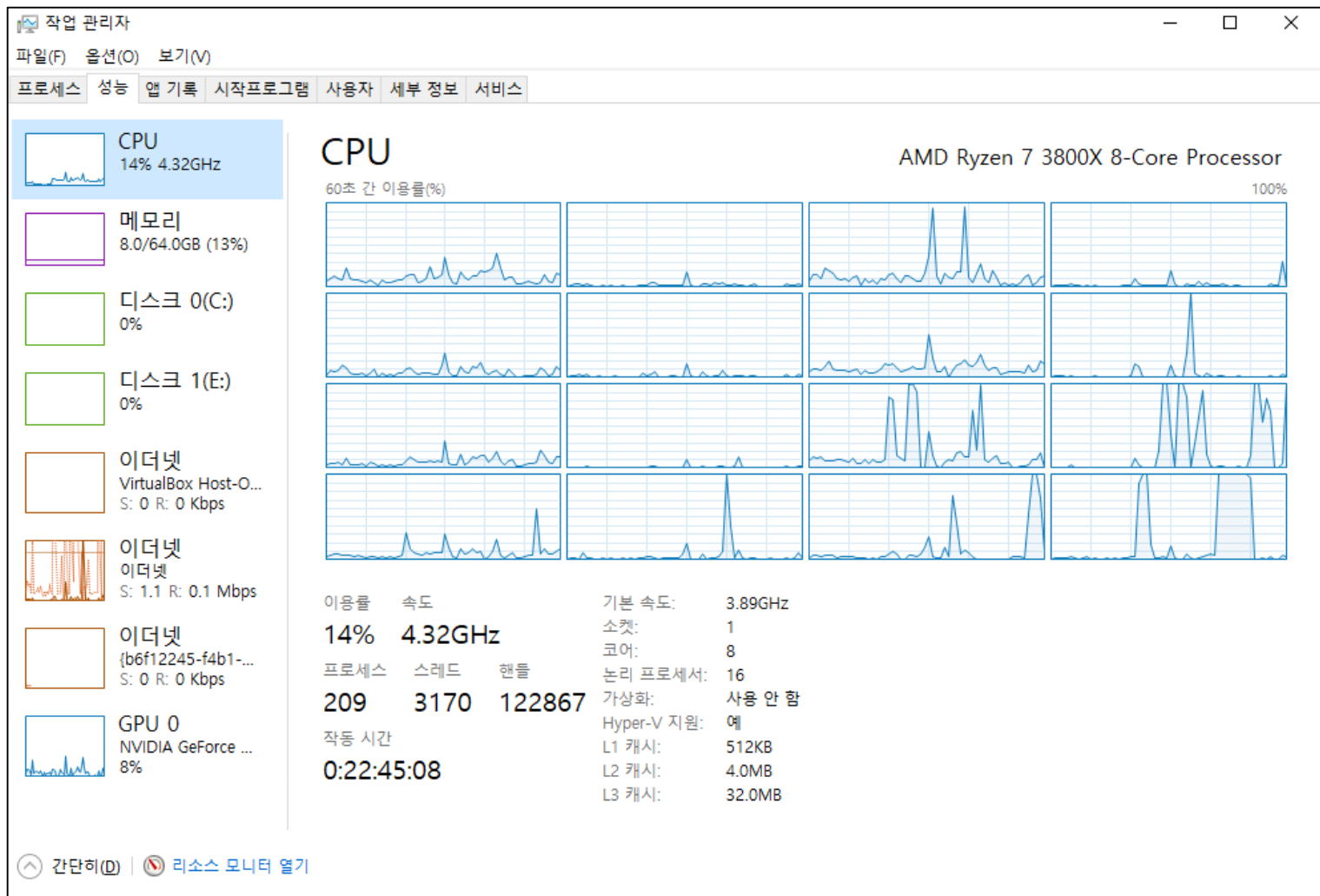
■ CPU 종류 : AMD Ryzen 7 3800X

■ 코어 : 8

■ 논리 프로세서(쓰레드) : 16

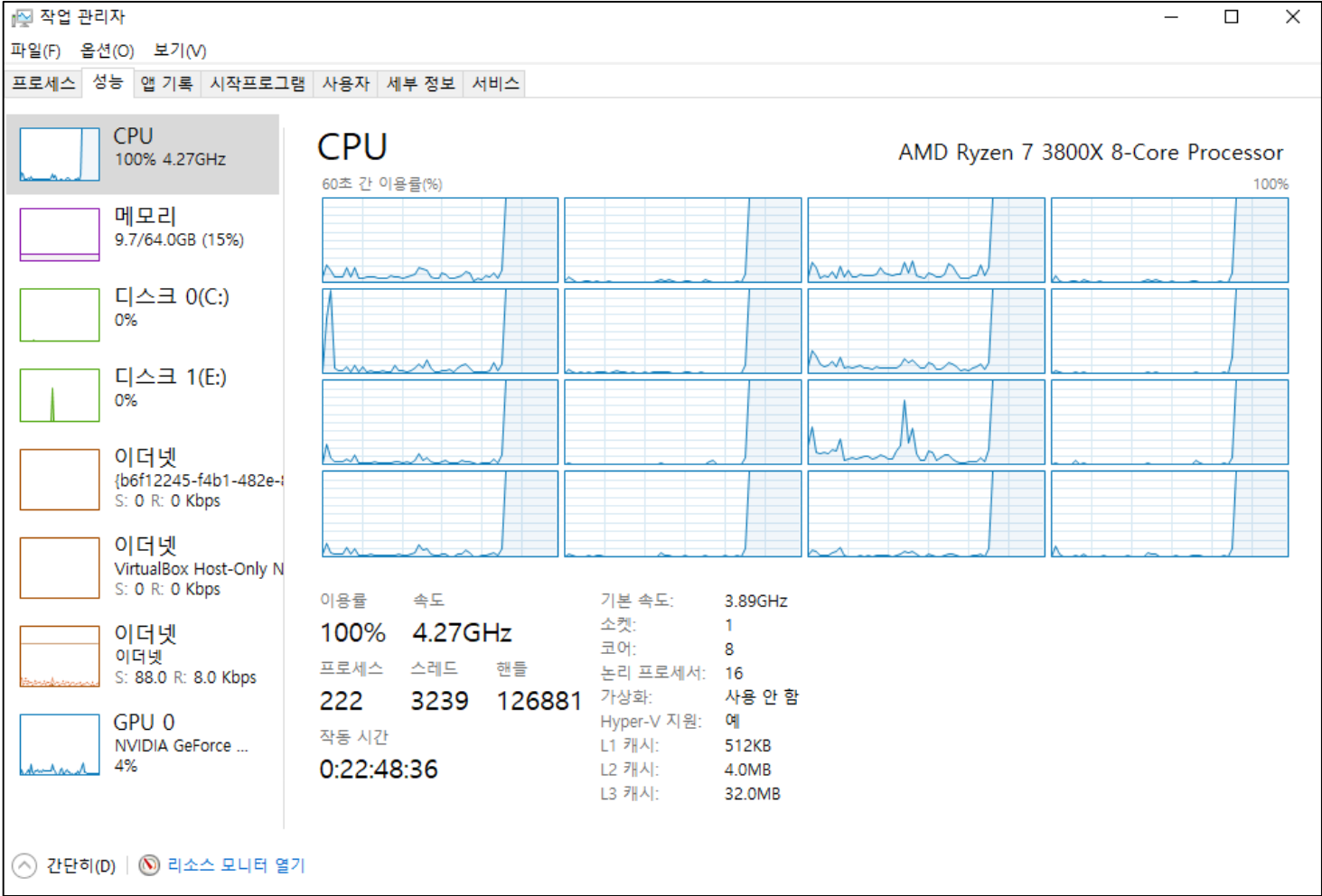
-> 총 16개로 작업을 나누어 처리할 수 있다.

# 1-1 병렬처리(Parallel Processing) 란?



- 일반적인 작업을 할 때 현황
- 이용률 14%
- 일부 프로세서만 사용

# 1-1 병렬처리(Parallel Processing) 란?



- 병렬처리를 했을 때 현황
- 이용률 100%
- 모든 프로세서가 작업을 처리하는 것을 볼 수 있음.



# 1-1 병렬처리(Parallel Processing) 란?

사용자	시스템	elapsed
86.16	0.00	86.17

사용자	시스템	elapsed
1.04	0.27	31.61



- 위는 일반 for문 아래는 병렬처리를 한 결과
- 약 2.5배 속도 향상

# 1-2 병렬처리실습

## ■ R에서의 병렬처리 기본 사이클 (parallel 패키지 - apply계열적용)

1. 패키지로드
2. CPU 코어 개수 획득.
3. 획득된 CPU 코어개수만큼 클러스터 등록
4. 병렬 연산 수행
5. 클러스터 중지

```
### parallel 패키지
library(parallel)

# 코어 개수 획득
numCores <- parallel::detectCores() - 1

# 클러스터 초기화 -> 백엔드 결과 추가하기 (작업관리자)
myCluster <- parallel::makeCluster(numCores)

# CPU 병렬처리
parallel::parLapply(cl = myCluster, x = 2:4, fun = function(x) {2^x})

# 클러스터 중지
parallel::stopCluster(myCluster)
```

# 1-2 병렬처리실습

## 15개의 front-end가 생성

작업 관리자

파일(F) 옵션(O) 보기(V)

프로세스 성능 열 기록 시작프로그램 사용자 세부 정보 서비스

이름	상태	8% CPU	13% 메모리	0% 디스크	0% 네트워크	전력 사용량	전력 사용량 ...
RStudio(18)		0.1%	1,467.3MB	0MB/s	0Mbps	매우 낮음	매우 낮음
RStudio R Session		0%	673.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
Qt Qtwebengineprocess		0.1%	126.9MB	0MB/s	0Mbps	매우 낮음	매우 낮음
RStudio		0.1%	114.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
R for Windows front-end		0%	36.8MB	0MB/s	0Mbps	매우 낮음	매우 낮음
Google Chrome(16)		0.4%	801.4MB	0.1MB/s	0Mbps	매우 낮음	매우 낮음
TeamViewer(32비트)		1.1%	351.5MB	0MB/s	0Mbps	낮음	보통
Microsoft Edge(16)		0%	260.0MB	0MB/s	0Mbps	매우 낮음	매우 낮음

간단히(D) 작업 끝내기(E)

- lapply와 동일 한 결과 확인
- 그 외 Sapply등 다른 함수도 적용 가능

```
> parallel::parLapply(cl = myCluster, X = 2:4, fun = function(x) {2^x})
[[1]]
[1] 4

[[2]]
[1] 8

[[3]]
[1] 16
```

# 1-2 병렬처리실습

```
parallel::parLapply(cl = myCluster, X = 2:4, fun = function(x) {2^x})
```

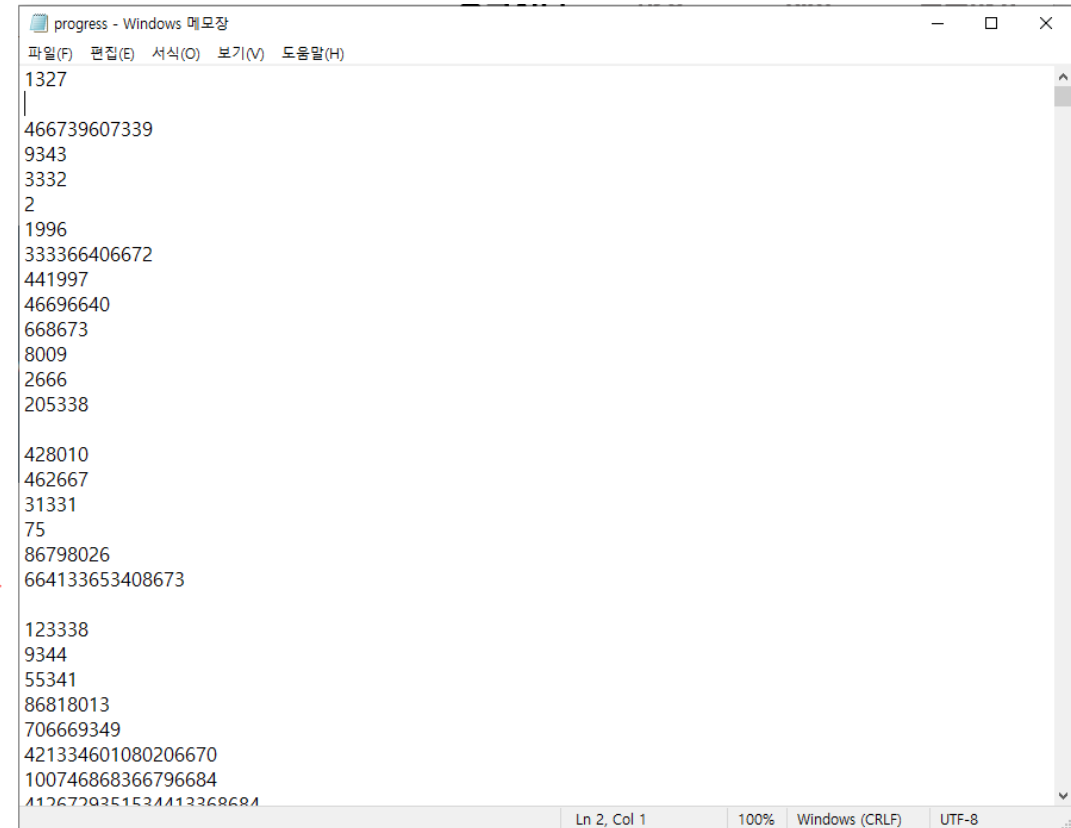
## ■ parLapply(cl , x, fun, ...)

1. cl : 앞서 만든 클러스터 객체
  2. x : 처리할 벡터, 리스트 등등
  3. fun : 적용할 함수
- parSapply (cl , x, fun, USE.NAMES = TRUE)
  - parRApply (cl , x, fun) : Row apply
  - parCApply (cl , x, fun) : Column apply
    - ✓ 이 둘이 parApply보다 효율적

# 1-2 병렬처리실습

```
myCluster <- parallel::makeCluster(numCores)
setwd("C:/Users/82104/Desktop")
iseq <- seq(1, 10000, 1)
parLapply(myCluster, iseq, function(y){
  write(y, "progress.txt", append=T)
})
# 클러스터 중지
parallel::stopCluster(myCluster)
```

- 각자 처리하는 순서가 다르므로 원하는 순서대로 합쳐지지 않음
- 순서가 중요하지 않을 때 적용가능



```
progress - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
1327
|
466739607339
9343
3332
2
1996
333366406672
441997
46696640
668673
8009
2666
205338

428010
462667
31331
75
86798026
664133653408673

123338
9344
55341
86818013
706669349
421334601080206670
100746868366796684
4126729351534413369684
Ln 2, Col 1 100% Windows (CRLF) UTF-8
```

# 1-2 병렬처리실습

## ■ 변수 스코프

```
# 코어 개수 획득
numCores <- parallel::detectCores() - 1

# 클러스터 초기화

myCluster <- parallel::makeCluster(numCores)

# 변수 등록
base <- 2
parallel::clusterExport(myCluster, "base")
# CPU 병렬처리
parallel::parLapply(cl = myCluster,
                    x = 2:4,
                    fun = function(x) {
                        base^x
                    })

# 클러스터 중지
parallel::stopCluster(myCluster)
```

- parallel 패키지의 parApply() 종류의 함수들은 변수를 미리 등록하지 않으면 함수 밖에서 선언 하였어도 사용X
- clusterExport 함수를 활용하여 변수(객체)를 등록해야함.
- 이 때, 변수(객체) 이름은 반드시 따옴표 안에 정의해야 함을 유념!!

# 1-2 병렬처리실습

- foreach 패키지 – for loop 와 lapply() 함수를 융합한 것

```
###foreach 패키지
library(foreach)
library(doParallel)

# 코어 개수 획득
numCores <- parallel::detectCores() - 1

# 클러스터 초기화
myCluster <- parallel::makeCluster(numCores)
doParallel::registerDoParallel(myCluster)

# 변수 등록, 안해도 상관없음
base <- 2
parallel::clusterExport(myCluster, "base")

# CPU 병렬처리, c는 cbind 느낌
foreach::foreach(exponent = 2:4, .combine = c) %dopar% {
  base^exponent
}

# rbind는 행으로 붙임
#foreach::foreach(exponent = 2:4, .combine = rbind) %dopar% {
#  base^exponent
#}

# 클러스터 중지
parallel::stopCluster(myCluster)
```

- registerDoParallel() : 만들어진 클러스터에서 병렬처리를 할 수 있도록 할당해주는 함수
- clusterExport 함수를 활용하지 않아도 무방.
- exponent 는 for문의 i 라고 생각하면 됨.
- combine = 부분은 행으로 붙힐지 열로 붙힐지에 대한 옵션! c : cbind , rbind : rbind

<b>[1]</b>	<b>4</b>	<b>8</b>	<b>16</b>	
				<b>[,1]</b>
				result.1
				4
				result.2
				8
				result.3
				16

- %dopar% : 병렬처리하게 하는 구문

# 1-2 병렬처리실습

- 함수 안에서 foreach() 를 사용할 경우 외부변수를 선언해야 하므로 .export 함수 제공

```
# foreach() 함수를 별도의 외부 함수로 정의할 경우 오류가 나므로 외부변수를  
#사용하는데 있어 불편함을 해소하기 위하여 export 옵션 제공  
test <- function(exponent) {  
  foreach::foreach(exponent = 2:4,  
                    .combine = c,  
                    .export = "base") %dopar% {  
    base^exponent  
  }  
}  
  
test()
```



# 1-2 병렬처리실습

- foreach 는 front-end를 불러오므로 packages() 를 통해 추가로 패키지를 지정해줘야함.

```
# 코어 개수 획득
numCores <- parallel::detectCores() - 1

# 클러스터 초기화
myCluster <- parallel::makeCluster(numCores)
doParallel::registerDoParallel(myCluster)

set.seed(1234)
folds = createFolds(iris$Sepal.Length,k=3)

td_tmp = foreach::foreach(k = 1:3,
  .combine = rbind,
  .packages=c("dplyr","broom","caret"),
  .inorder = TRUE) %dopar% {
  tmp = iris[-unlist(folds[k]),] %>% group_by(Species) %>%
    do(fit = lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data = .))
  tidy(tmp,fit)
}

# 클러스터 중지
parallel::stopCluster(myCluster)
```

- .packages : for문 안에 사용되는 패키지 지정
- inorder : 원래 순서대로 저장할 것인지

아래 코드는 병렬처리와는 관련 없는 코드 설명

- do : dplyr로 만든 데이터를 뒤에서 함수를 적용
- tidy : 각 데이터 별로 모형결과 저장
- 요약하자면 iris를 fold3으로 나누어 그룹별로 회귀모형을 만든 후 그 결과를 저장하는 코드

# 1-2 병렬처리실습

- 하나의 processor가 error가 났다고 해서 전체 연산을 중지시키는 것은 비효율적
- try를 이용하여 에러를 catch하고 에러의 발생원인을 설명하는 텍스트를 반환

```
# 코어 개수 획득
numCores <- parallel::detectCores() - 1

# 클러스터 초기화
myCluster <- parallel::makeCluster(numCores, type = "PSOCK")

# CPU 병렬처리
foreach(x=list(1, 2, "a")) %dopar% {
  tryCatch({
    c(1/x, x, 2^x)
  }, error = function(e) {
    return(paste0("The variable '", x, "'", " caused the error: '", e, "'"))
  })
}

# 클러스터 중지
parallel::stopCluster(myCluster)
```

```
[[1]]
[1] 1 1 2

[[2]]
[1] 0.5 2.0 4.0

[[3]]
[1] "The variable 'a' caused the error: 'Error in 1/x: 이항연산자에 수치가 아닌 인수입니다\n'"
```

# 1-2 병렬처리실습

- 병렬처리가 무조건 빠른 것은 아니다!!!

```
> system.time({for(i in 1:10000){  
+   i+5  
+ }})
```

사용자	시스템	elapsed
0	0	0

```
> n_core = detectCores()  
> cl = makeCluster(n_core-1)  
> registerDoParallel(cl)  
> system.time(  
+   {foreach(i = 1:10000) %dopar%{  
+     i+5  
+   }}  
+ )
```

사용자	시스템	elapsed
1.91	0.30	2.26

# 1-2 병렬처리실습



- 각 클러스터 15개에 i+5라는 작업을 할당하는 시간이 더 오래 걸리기 때문
- 일상의 예를 들자면, 조별과제를 하는데 1부터 100까지 받아쓰기를 4명에서 누가 어디서부터 어디까지 할지 정하고 하는 것 보다 한명이 쓰는 것이 훨씬 빠른 경우이다.

---

## 02

### **h2o 패키지**

2-1. h2o 패키지란?

2-2. h2o 실습

## 2-1 h2o란?



- 위 패키지들의 특징은?

## 2-1 h2o란?



- 선형 확장성을 갖춘 분산형 메모리 내 머신러닝 플랫폼을 가진 오픈소스
  1. Gradient boosted machines, Generalized Linear Models, Deep Learning 알고리즘, word2vec 등을 포함하여 가장 널리 사용되는 통계 및 머신러닝 알고리즘을 지원한다.
  2. 모든 알고리즘과 그 하이퍼 파라미터들을 자동으로 리더보드를 통해 찾는다.
  3. R과 Python 모두 지원하며 Jupyter Notebook 과 비슷하게 생긴 로컬호스트 기반의 웹(UI)을 지원
  4. 병렬처리 기능 및 GPU기능(H204GPU in Linux) 지원.

# 2-1 h2o란?

localhost:54321/flow/index.html

H<sub>2</sub>O FLOW

Untitled Flow

Frame Name	Rows	Columns	Size
RTMP_sid_a23e_5	1	3	810Bytes
RTMP_sid_a23e_6	101032	1	13KB
RTMP_sid_a23e_7	101032	1	13KB
RTMP_sid_a23e_8	101032	1	13KB
RTMP_sid_a23e_9	101032	1	13KB
predictions_8060_rf_model_on_test_data_h2o	101032	3	2MB
test_data_h2o	101032	7	910KB
train_data_h2o	235744	7	2MB
transformation_80df_rf_model_on_test_data_h2o	101032	3	2MB

데이터가 Java Virtual Machine 상에 올라가있음.

getModels

Ready

Connections: 0 H<sub>2</sub>O



# 2-1 h2o란?

## 1. 전처리 기능 지원

The screenshot displays the H2O.ai documentation interface. The top navigation bar is yellow with the H2O.ai logo and version 3.20.0.3. A search bar is present. The left sidebar lists various topics, with 'Data Manipulation' expanded to show a list of sub-topics. The main content area is titled 'Data Manipulation' and includes an introductory paragraph, a note about data sources, a list of links to specific topics, and navigation buttons for 'Previous' and 'Next'. The footer contains copyright information and mentions the use of Sphinx and Read the Docs theme.

**H2O.ai**  
3.20.0.3

Search docs

Welcome to H2O 3  
Quick Start Videos  
Cloud Integration  
Downloading & Installing H2O  
Starting H2O  
Getting Data into Your H2O Cluster

**Data Manipulation**

- Uploading a File
- Importing a File
- Importing Multiple Files
- Combining Columns from Two Datasets
- Combining Rows from Two Datasets
- Fill NAs
- Group By
- Imputing Data
- Merging Two Datasets
- Pivoting Tables
- Replacing Values in a Frame
- Slicing Columns
- Slicing Rows
- Sorting Columns
- Splitting Datasets into Training/Testing/Validating
- Target Encoding

[Docs](#) » Data Manipulation [View page source](#)

### Data Manipulation

This section provides examples of common tasks performed when preparing data for machine learning. These examples are run on a local cluster.

**Note:** The examples in this section include datasets that are pulled from GitHub and S3.

- [Uploading a File](#)
- [Importing a File](#)
- [Importing Multiple Files](#)
- [Combining Columns from Two Datasets](#)
- [Combining Rows from Two Datasets](#)
- [Fill NAs](#)
- [Group By](#)
- [Imputing Data](#)
- [Merging Two Datasets](#)
- [Pivoting Tables](#)
- [Replacing Values in a Frame](#)
- [Slicing Columns](#)
- [Slicing Rows](#)
- [Sorting Columns](#)
- [Splitting Datasets into Training/Testing/Validating](#)
- [Target Encoding](#)

[Previous](#) [Next](#)

© Copyright 2016-2018 H2O.ai. Last updated on Jul 11, 2018.  
Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).

# 2-1 h2o란?

## 2. 모델링

The screenshot displays the H2O.ai documentation interface. The top header is yellow with the H2O.ai logo and version 3.20.0.3. A search bar is present. The left sidebar is dark grey with a list of navigation links. The main content area is white and titled 'Algorithms'. It includes a 'Common' section with links to Quantiles and Early Stopping, a 'Supervised' section with links to Cox Proportional Hazards (CoxPH), Deep Learning (Neural Networks), Distributed Random Forest (DRF), Generalized Linear Model (GLM), Gradient Boosting Machine (GBM), Naïve Bayes Classifier, Stacked Ensembles, and XGBoost, an 'Unsupervised' section with links to Aggregator, Generalized Low Rank Models (GLRM), K-Means Clustering, and Principal Component Analysis (PCA), and a 'Miscellaneous' section with a link to Word2vec. Navigation buttons for 'Previous' and 'Next' are at the bottom.

**H2O.ai**  
3.20.0.3

Search docs

Welcome to H2O 3  
Quick Start Videos  
Cloud Integration  
Downloading & Installing H2O  
Starting H2O  
Getting Data into Your H2O Cluster  
Data Manipulation

Algorithms

- Common
- Supervised
- Unsupervised
- Miscellaneous

Cross-Validation  
Grid (Hyperparameter) Search  
Checkpointing Models  
AutoML: Automatic Machine Learning  
Saving and Loading a Model  
Productionizing H2O  
Using Flow - H2O's Web UI  
Downloading Logs  
H2O Architecture  
Security  
FAQ  
Glossary  
Migrating to H2O 3

Docs » Algorithms [View page source](#)

### Algorithms

This section provides an overview of each algorithm available in H2O. For detailed information about the parameters that can be used for building models, refer to [Appendix A - Parameters](#).

#### Common

- [Quantiles](#)
- [Early Stopping](#)

#### Supervised

- [Cox Proportional Hazards \(CoxPH\)](#)
- [Deep Learning \(Neural Networks\)](#)
- [Distributed Random Forest \(DRF\)](#)
- [Generalized Linear Model \(GLM\)](#)
- [Gradient Boosting Machine \(GBM\)](#)
- [Naïve Bayes Classifier](#)
- [Stacked Ensembles](#)
- [XGBoost](#)

#### Unsupervised

- [Aggregator](#)
- [Generalized Low Rank Models \(GLRM\)](#)
- [K-Means Clustering](#)
- [Principal Component Analysis \(PCA\)](#)

#### Miscellaneous

- [Word2vec](#)

[Previous](#) [Next](#)

# 2-1 h2o란?

## 3. Auto ML

The screenshot shows the H2O Flow web interface. The main panel displays a list of data frames and their associated actions. A red box highlights the 'Run AutoML' button for the 'train\_data\_h2o' frame.

Frame Name	Build Model	Run AutoML	Predict	Inspect
RTMP_sid_a23e55	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RTMP_sid_a23e56	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RTMP_sid_a23e57	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RTMP_sid_a23e58	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RTMP_sid_a23e59	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
predictions_806_rf_model_on_test_data_h2o	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
test_data_h2o	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
train_data_h2o	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
transformation_806_rf_model_on_test_data_h2o	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

At the bottom of the interface, there is a console area showing the command `getModels` and a status bar indicating 'Ready' and 'Connections: 0'.

## 2-1 h2o란?

### ■ 장점

1. 모델링이 쉽다. (웹에서 클릭기반의 모델링 가능)
2. 대용량 데이터를 이용한 모델링이 빠르다
3. 모델배포가 어렵지 않다.

### ■ 단점

1. 작은 데이터의 경우 올리는 시간이 오히려 오래 걸릴 수 있다.
2. Deep Learning의 경우 DNN 이외에 다른 구조(CNN, RNN 등)를 지니는 모델을 생성할 수 없다.

## 2-2h2o 실습 - 데이터준비

- 비행기 연착 시간이 30분 이하인 경우 정상으로 처리, 30분 이상인 경우 연착으로 타겟변수 생성

```
library(dplyr)
library(caret)

flights_data = readRDS("C:/Users/82104/Desktop/flights.RDS")

head(flights_data)
str(flights_data)

flights_data$target <- ifelse((is.na(flights_data$dep_delay) | (flights_data$dep_delay <= 30 & flights_data$dep_delay >= -30)) &
                             (is.na(flights_data$arr_delay) | (flights_data$arr_delay <= 30 & flights_data$arr_delay >= -30)), "normal", "delay")
table(flights_data$target)
```

```
> str(flights_data)
tibble [336,776 x 22] (s3: tbl_df/tbl/data.frame)
 $ year      : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ month     : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
 $ day       : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
 $ dep_time  : chr [1:336776] "0517" "0533" "0542" "0544" ...
 $ sched_dep_time: chr [1:336776] "0515" "0529" "0540" "0545" ...
 $ dep_delay : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -2 ...
 $ arr_time  : chr [1:336776] "0830" "0850" "0923" "1004" ...
 $ sched_arr_time: chr [1:336776] "0819" "0830" "0850" "1022" ...
 $ arr_delay : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
 $ carrier   : chr [1:336776] "UA" "UA" "AA" "B6" ...
 $ flight    : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
 $ tailnum   : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
 $ origin    : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
 $ dest      : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
 $ air_time  : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
 $ distance  : num [1:336776] 1400 1416 1089 1576 762 ...
 $ hour      : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
 $ minute    : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
 $ time_hour : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
 $ dep_dt    : POSIXct[1:336776], format: "2013-01-01 05:15:00" "2013-01-01 05:29:00" "2013-01-01 05:40:00" "2013-01-01 05:45:00" ...
 $ arr_dt    : POSIXct[1:336776], format: "2013-01-01 08:19:00" "2013-01-01 08:30:00" "2013-01-01 08:50:00" "2013-01-01 10:22:00" ...
 $ target    : chr [1:336776] "normal" "normal" "delay" "normal" ...
```

```
> table(flights_data$target)
```

```
delay normal
79969 256807
```

## 2-2h2o 실습 - 데이터준비

- 모델링을 위해 일부 변수만 사용 및 categorical 변수 factor로 변경

```
final_data <- flights_data %>% select("month","carrier","flight","dest","air_time","distance","target")
str(final_data)

final_data$carrier <- as.factor(final_data$carrier)
final_data$dest <- as.factor(final_data$dest)
final_data$target <- as.factor(final_data$target)
```

- 데이터 분할

```
##train, test 나누기
set.seed(1234)
train_idx <- createDataPartition(final_data$target, p=0.7, list = F)
train<-final_data[train_idx, ]
test<-final_data[-train_idx, ]
```

## 2-2 h2o 실습 - h2o 이해

```
library(h2o)
Sys.setenv(JAVA_HOME="C:/Program Files/Java/jdk-13.0.2")
h2o.init(nthreads = 15, max_mem_size = "10g")
#h2o::h2o.shutdown(prompt = FALSE)
```

- h2o 라이브러리 설치 및 불러오기
- Sys.setenv : Java 경로 설정 (h2o는 자바 가상환경을 사용하므로)
- h2o.init : h2o에 등록하는 과정, nthreads : thread개수,  
max\_mem\_size : 자바에 할당할 메모리, m : megabytes, g: gigabytes  
그 밖에 아이피, 포트넘버, 아이디 비밀번호 등록 등의 기능 존재
- h2o.shutdown : h2o와 연결종료, prompt = T는 web에 있는 h2o instance까지 전부 끄는 옵션

## 2-2 h2o 실습 - h2o 이해

```
> h2o.init(nthreads = 15, max_mem_size = "10g")
```

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

Note: In case of errors look at the following log files:

C:\Users\82104\AppData\Local\Temp\RtmpsxmLAK\file3e4c40917f11\h2o\_82104\_started\_from\_r.out

C:\Users\82104\AppData\Local\Temp\RtmpsxmLAK\file3e4c423b6fc5\h2o\_82104\_started\_from\_r.err

java version "13.0.2" 2020-01-14

Java(TM) SE Runtime Environment (build 13.0.2+8)

Java HotSpot(TM) 64-Bit Server VM (build 13.0.2+8, mixed mode, sharing)

Starting H2O JVM and connecting: Connection successful!

R is connected to the H2O cluster:

H2O cluster uptime: 5 seconds 31 milliseconds

H2O cluster timezone: Asia/Seoul

H2O data parsing timezone: UTC

H2O cluster version: 3.30.0.1

H2O cluster version age: 3 months and 18 days !!!

H2O cluster name: H2O\_started\_from\_R\_82104\_uwy737

H2O cluster total nodes: 1

H2O cluster total memory: 10.00 GB

H2O cluster total cores: 16

H2O cluster allowed cores: 15

H2O cluster healthy: TRUE

H2O connection ip: localhost

H2O connection port: 54321

H2O connection proxy: NA

H2O Internal Security: FALSE

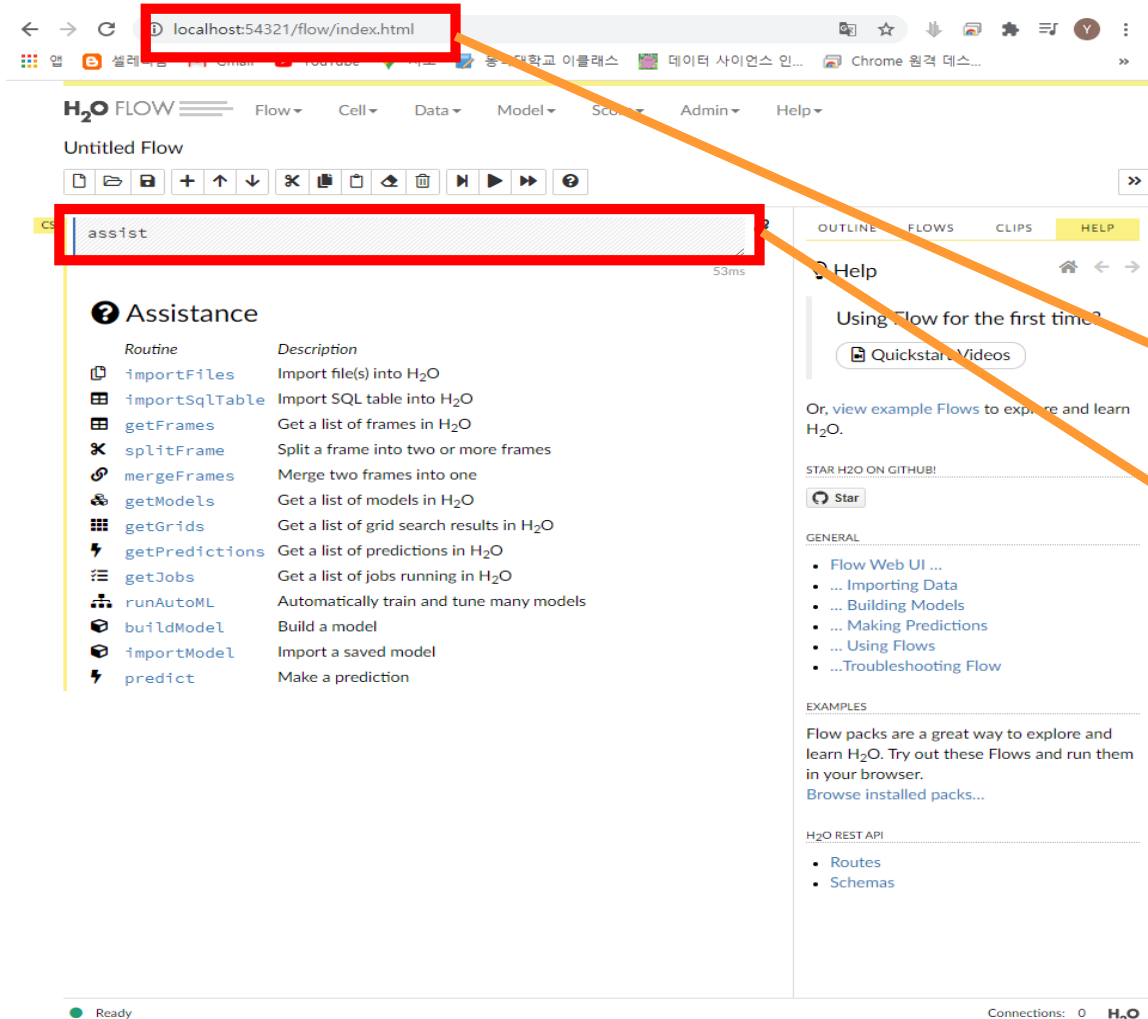
H2O API Extensions: Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4

R Version: R version 3.6.3 (2020-02-29)

- Java 14 version은 지원하지 않음에 유의!!
- 연결에 성공했다는 알림 확인
- 연결하는데 걸린시간
- 배정 메모리 및 코어(쓰레드) 개수
- 웹에서의 포트 local host 넘버
- 오래된 버전의 경우 경고창이 나오나 무시



## 2-2 h2o 실습 - h2o이해



■ <http://localhost:54321/> 를 주소창에 입력  
할 경우 접속

■ h2o의 코드를 입력하거나 클릭을 통해 여러  
기능 사용

## 2-2 h2o 실습 - h2o 데이터 등록

```
> train_data_h2o <- as.h2o(train, destination_frame = "train_data_h2o")
|=====| 100%
> test_data_h2o <- as.h2o(test, destination_frame = "test_data_h2o")
|=====| 100%
```

- as.h2o : Java Virtual Machine (자바가상환경) 상에 데이터를 등록
- destination\_frame : ui 상에 등록될 이름
- h2o의 특징으로 progress 진행 상황을 보여줌

# 2-2 h2o 실습 - h2o 데이터 등록

H2O FLOW

Untitled Flow

getFrames

Frames

현재 등록되어있는 모든 데이터 확인

Type	ID	Rows	Columns	Size
test_data_h2o		101032	7	910KB
train_data_h2o		235744	7	2MB

getFrameSummary "train\_data\_h2o"

train\_data\_h2o

특정 데이터 요약 보기

Actions: View Data, Split, Build Model, Run AutoML, Predict, Download, Export

Rows	Columns	Compressed Size
235744	7	2MB

COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
month	int	0	0	0	0	1.0	12.0	6.5399	3.4110	16	Convert to enum
carrier	enum	0	12947	0	0	0	15.0	.	.	16	Convert to numeric
flight	int	0	0	0	0	1.0	8500.0	1972.1308	1634.2719	164	Convert to enum
dest	enum	0	182	0	0	0	103.0	.	.	164	Convert to numeric
air_time	int	6602	0	0	0	20.0	695.0	150.6495	93.7266	164	Convert to enum
distance	int	0	0	0	0	80.0	4983.0	1039.6007	733.5502	164	Convert to enum
target	enum	0	55979	0	0	0	1.0	0.7625	0.4255	2	Convert to numeric

Previous 20 Columns, Next 20 Columns

CHUNK COMPRESSION SUMMARY

FRAME DISTRIBUTION SUMMARY

Ready

Connections: 0 H2O

Help

Importing Data

If you don't have any data of your own to work with, you can find some example datasets at <http://data.h2o.ai>.

There are multiple ways to import data in H2O flow:

- Click the **Assist Me!** button in the row of buttons below the menus, then click the **importFiles** link. Enter the file path in the auto-completing **Search** entry field and press **Enter**. Select the file from the search results and confirm it by clicking the **Add All** link.

Note: For S3 file locations, use the format

```
importFiles [
  "s3n:/path/to/bucket/file/"
]
```

## 2-2h2o 실습 - h2o 모델링

### ◆ 모델링시 사용할 변수를 지정

```
> target <- "target"
> features <- names(train)[!names(train) %in% target]
> target ; features
[1] "target"
[1] "month"      "carrier"    "flight"     "dest"       "air_time"   "distance"
```

### ◆ 랜덤포레스트 모형 구축

```
> rf_model <- h2o.randomForest(x = features, y = target, training_frame = train_data_h2o,
+                               model_id = "rf_model", ntrees = 500, seed = 1234, mtries = floor(ncol(train) / 3), verbose = F)
|=====| 100%
```

- x, y : 독립변수명, 종속변수명 지정
- model\_id : ui상에 저장될 이름
- 그 외 : 기존의 랜덤포레스트와 옵션 동일
- training\_frame : train에 사용될 데이터
- verbose : 그림과 같이 진행상황을 줄을 바꿔가며 보여줌

```
Scoring History for Model rf_model at 2020-07-22 05:34:53
[1] "Model Build is 0% done..."
[1] "Scoring History is not available yet..."

Scoring History for Model rf_model at 2020-07-22 05:34:54
[1] "Model Build is 0% done..."
|=====| 1%

Scoring History for Model rf_model at 2020-07-22 05:34:55
[1] "Model Build is 0.6% done..."
|=====| 2%

Scoring History for Model rf_model at 2020-07-22 05:34:56
[1] "Model Build is 1.6% done..."
|=====| 3%

Scoring History for Model rf_model at 2020-07-22 05:34:57
[1] "Model Build is 3% done..."
|=====| 5%

Scoring History for Model rf_model at 2020-07-22 05:34:58
[1] "Model Build is 4.8% done..."
|=====| 6%

Scoring History for Model rf_model at 2020-07-22 05:34:59
[1] "Model Build is 6.2% done..."
|=====| 8%
```

## 2-2h2o 실습 - h2o 모델링

```
> rf_model
Model Details:
=====
H2OBinomialModel: drf
Model ID: rf_model
Model Summary:
  number_of_trees number_of_internal_trees model_size_in_bytes min_depth max_depth mean_depth min_leaves max_leaves mean_leaves
1             500              500          115879417         20         20      20.00000     14007     20430 17533.12100

H2OBinomialMetrics: drf
** Reported on training data. **
** Metrics reported on Out-Of-Bag training samples **

MSE: 0.1620764
RMSE: 0.4025871
LogLoss: 0.497202
Mean Per-Class Error: 0.4642114
AUC: 0.7096272
AUCPR: 0.8769027
Gini: 0.4192544
RA2: 0.1049018

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
      delay normal Error Rate
delay  4842  51137 0.913503 =51137/55979
normal 2682 177083 0.014919 =2682/179765
Totals 7524 228220 0.228294 =53819/235744

Maximum Metrics: Maximum metrics at their respective thresholds
 1      metric threshold      value idx
 2      max f1  0.390464    0.868086 332
 3      max f2  0.100963    0.941397 394
 4      max f0point5 0.661286 0.826481 217
 5      max accuracy 0.447686 0.772647 314
 6      max precision 0.993787 0.972973  0
 7      max recall  0.068001 1.000000 398
 8      max specificity 0.993787 0.999982  0
 9      max absolute_mcc 0.689157 0.270698 201
10      max min_per_class_accuracy 0.765750 0.651178 152
11      max mean_per_class_accuracy 0.764136 0.652260 153
12      max tns 0.993787 55978.000000  0
13      max fns 0.993787 179729.000000  0
14      max fps 0.044693 55979.000000 399
15      max tps 0.068001 179765.000000 398
16      max tnr 0.993787 0.999982  0
17      max fnr 0.993787 0.999800  0
18      max fpr 0.044693 1.000000 399
19      max tpr 0.068001 1.000000 398

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
```

- 모델 hyper parameter 확인
- 여러 지표들 확인
- train data의 confusion matrix 확인
- thresholds (cut off) 확인

## 2-2 h2o 실습 - h2o 모델링

◆ Test 데이터로 예측 : `h2o.predict (model, newdata)`

```
> test_predict<-h2o.predict(rf_model,newdata=test_data_h2o)
|=====| 100%
경고메시지(들):
In doTryCatch(return(expr), name, parentenv, handler) :
  Test/Validation dataset column 'dest' has levels not trained on: [LGA]
> test_predict
  predict    delay    normal
1  normal 0.17359126 0.8264087
2  normal 0.14547741 0.8545226
3  normal 0.11709233 0.8829077
4  normal 0.22367329 0.7763267
5  normal 0.04325018 0.9567498
6  normal 0.13608877 0.8639112

[101032 rows x 3 columns]
```

- 각 delay와 normal로 예측할 확률 및 predict 값 제공

## 2-2h2o 실습 - h2o 모델링

### ◆ 혼동행렬 : h2o.confusionMatrix(model, newdata)

```
> h2o.confusionMatrix(rf_model, newdata = test_data_h2o)
Confusion Matrix (vertical: actual; across: predicted) for max f1 @ threshold = 0.356487281410093:
      delay normal  Error      Rate
delay  1655  22335 0.931013  =22335/23990
normal   703   76339 0.009125  =703/77042
Totals  2358  98674 0.228027  =23038/101032

> h2o.confusionMatrix(rf_model, newdata = test_data_h2o, metrics = "accuracy")
Confusion Matrix (vertical: actual; across: predicted) for max accuracy @ threshold = 0.460905919278958:
      delay normal  Error      Rate
delay   3380  20610 0.859108  =20610/23990
normal   2156  74886 0.027985  =2156/77042
Totals   5536  95496 0.225335  =22766/101032

> h2o.confusionMatrix(rf_model, newdata = test_data_h2o, metrics = "accuracy", thresholds = 0.5)
[[1]]
Confusion Matrix (vertical: actual; across: predicted) @ threshold = 0.499245563679363:
      delay normal  Error      Rate
delay   4239  19751 0.823301  =19751/23990
normal   3062  73980 0.039745  =3062/77042
Totals   7301  93731 0.225800  =22813/101032

[[2]]
Confusion Matrix (vertical: actual; across: predicted) for max accuracy @ threshold = 0.460905919278958:
      delay normal  Error      Rate
delay   3380  20610 0.859108  =20610/23990
normal   2156  74886 0.027985  =2156/77042
Totals   5536  95496 0.225335  =22766/101032
```

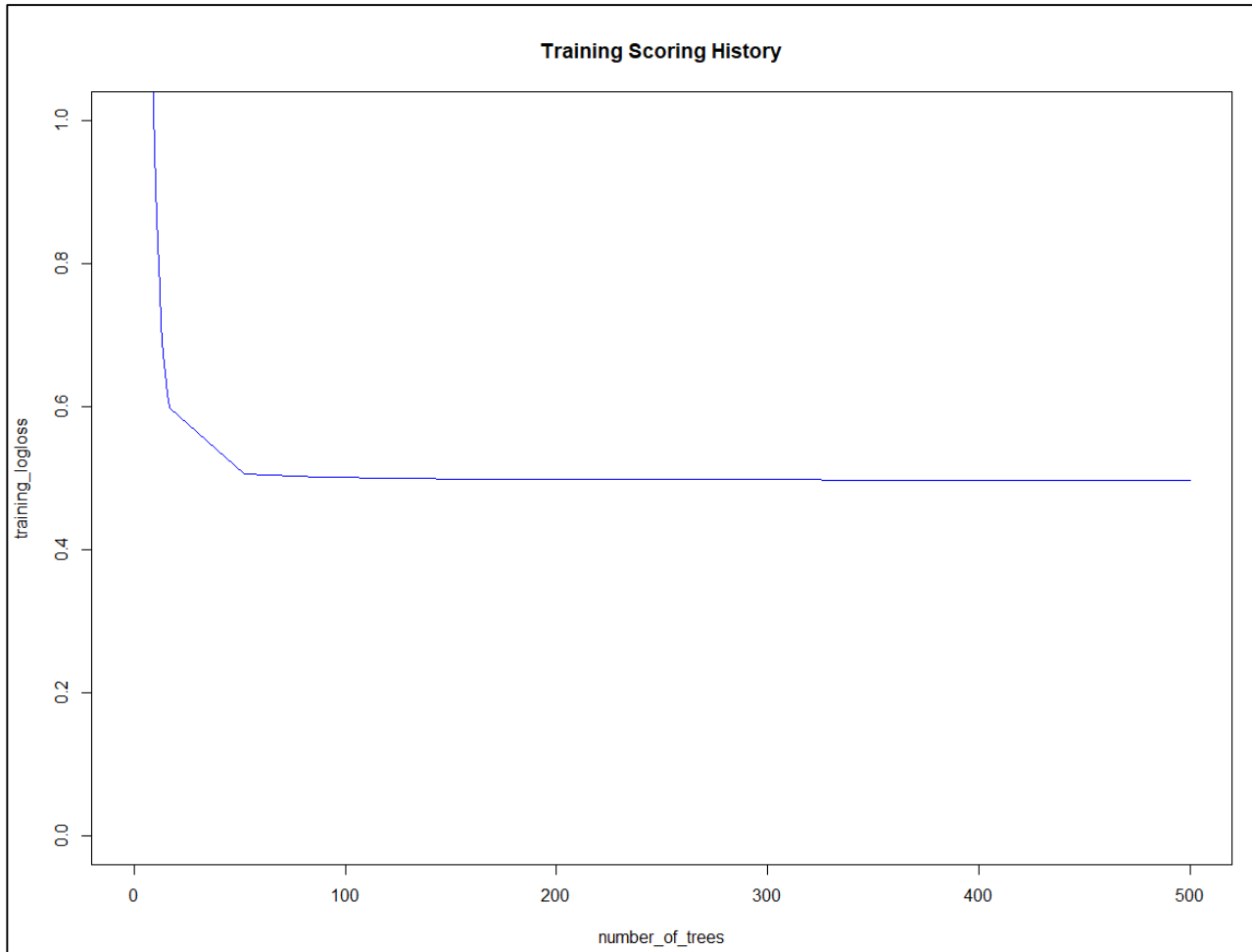
경고메시지(들):

In h2o.find\_row\_by\_threshold(object, t) :

could not find exact threshold: 0.5 for this set of metrics; using closest threshold found: 0.499245563679363. Run `h2o.predict` and apply your desired threshold on a probability column.

- default는 f1-score가 max가 되는 threshold를 적용
- metrics : 지표 변경 , tnr, fpr, precision 등
- thresholds : cut off 값 변경

## 2-2h2o 실습 - h2o 모델링



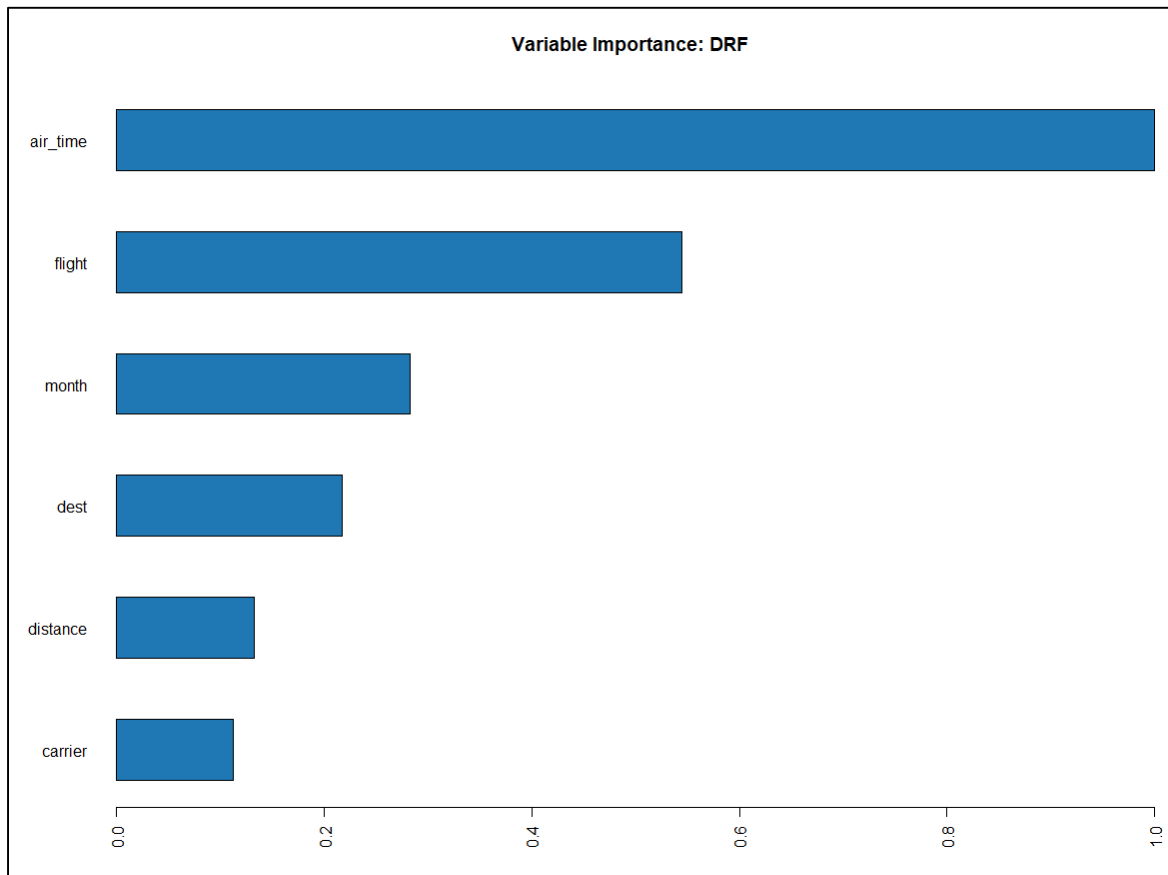
```
> plot(rf_model)
```

- tree 수의 증가에 따른 loss 변화
- 해당 그래프에서는 logloss



## 2-2 h2o 실습 - h2o 모델링

```
> h2o.varimp_plot(rf_model, num_of_features = 6)
```



- `h2o.varimp` : 변수중요도 값 확인
- `h2o.varimp_plot(model, 변수 개수)` : 해당 모델의 변수 중요도 그래프

## 2-2 h2o 실습 - h2o 모델 web ui에서 확인하기

◆ getModels : 저장되어있는 모든 모델 확인

The screenshot displays the H2O FLOW web interface. At the top, the navigation bar includes 'H2O FLOW' and several menu items: 'Flow', 'Cell', 'Data', 'Model', 'Score', 'Admin', and 'Help'. Below the navigation bar, the main area is titled 'Untitled Flow'. A toolbar with various icons is visible. The command 'getModels' is entered in the command bar, and the execution time is shown as 27ms. Below the command bar, the 'Models' section is displayed, showing a table with columns 'Key', 'Algorithm', and 'Actions'. The table contains one entry: 'rf\_model' with the algorithm 'Distributed Random Forest'. The 'Actions' column for 'rf\_model' includes 'Predict...' and 'Inspect' buttons. Below the table, there are 'Inspect' and 'Delete selected' buttons. The second part of the screenshot shows the command 'getModels "rf\_model"' with an execution time of 25ms. The 'Models' section again shows the 'rf\_model' entry. An orange arrow points to the 'rf\_model' entry in the table, with the text '클릭!!!' (Click!!!) next to it.

H2O FLOW Flow Cell Data Model Score Admin Help

Untitled Flow

getModels

27ms

Models

Key	Algorithm	Actions
rf_model	Distributed Random Forest	Predict... Inspect

Inspect Delete selected

getModels "rf\_model"

25ms

Models

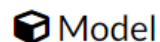
Key	Algorithm	Actions
rf_model	Distributed Random Forest	Predict... Inspect

Inspect Delete selected

클릭!!!

## 2-2 h2o 실습 - h2o 모델 web ui에서 확인하기

### ◆ 모델에 대한 상세 parameter 확인



Model ID: rf\_model

Algorithm: Distributed Random Forest

Actions: [Refresh](#) [Predict...](#) [Download POJO](#) [Download Model Deployment Package \(MOJO\)](#) [Export](#) [Inspect](#) [Delete](#) [Download Gen Model](#)

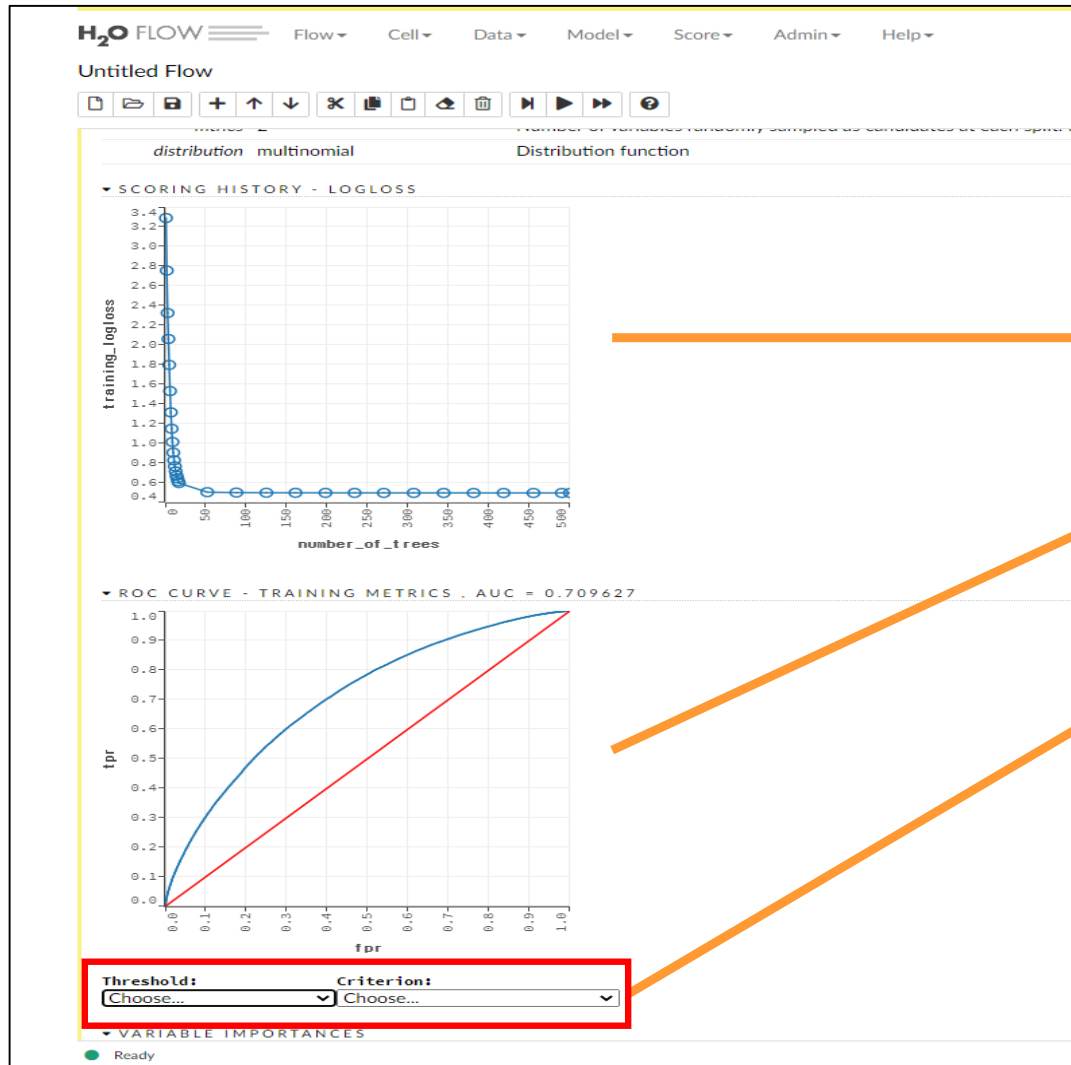
모델 predict값 저장 및 모델 저장기능 지원

#### ▼ MODEL PARAMETERS

Show all parameters

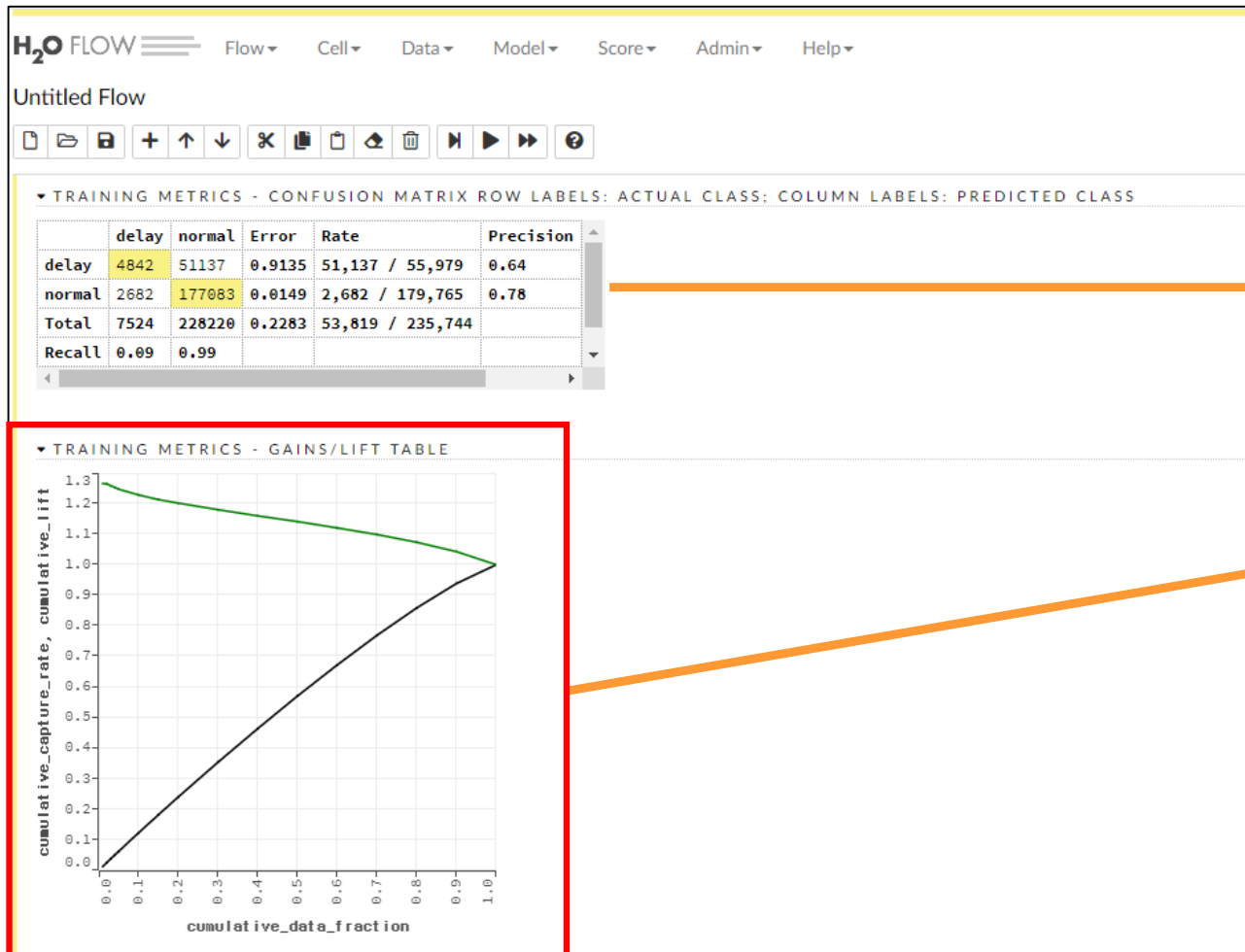
Parameter	Value	Description
model_id	rf_model	Destination id for this model; auto-generated if not specified.
training_frame	train_data_h2o	Id of the training data frame.
response_column	target	Response variable column.
ignored_columns		Names of columns to ignore for training.
ntrees	500	Number of trees.
r2_stopping	1.7976931348623157e+308	r2_stopping is no longer supported and will be ignored if set - please use stopping_rounds, stopping_metric and stopping_tolerance instead. Previous version of H2O would stop making trees when the R^2 metric equals or exceeds this
seed	1234	Seed for pseudo random number generator (if applicable)
mtrees	2	Number of variables randomly sampled as candidates at each split. If set to -1, defaults to sqrt(p) for classification and p/3 for regression (where p is the # of predictors)
distribution	multinomial	Distribution function

## 2-2 h2o 실습 - h2o 모델 web ui에서 확인하기



- plot() 으로 확인했었던 loss 변화 그래프
- Roc curve 확인
- Threshold 값과 f1-score , accuracy 등의 값 지정 가능


## 2-2 h2o 실습 - h2o 모델 web ui에서 확인하기



Confusion Matrix 확인

과적합 및 모델 성능 확인에 사용되는 lift 값도 계산해줌!

## 2-2 h2o 실습 - h2o 모델 web ui에서 확인하기

H<sub>2</sub>O FLOW  Flow ▾ Cell ▾ Data ▾ Model ▾ Score ▾ Admin ▾ Help ▾

Untitled Flow



▶ OUTPUT

▶ COLUMN\_TYPES

▶ OUTPUT - MODEL SUMMARY

▶ OUTPUT - SCORING HISTORY

▶ OUTPUT - TRAINING\_METRICS

▶ DOMAIN

▶ OUTPUT - TRAINING\_METRICS - METRICS FOR THRESHOLDS (BINOMIAL METRICS AS A FUNCTION OF CLASSIFICATION THRESHOLDS)

▶ OUTPUT - TRAINING\_METRICS - MAXIMUM METRICS (MAXIMUM METRICS AT THEIR RESPECTIVE THRESHOLDS)

▶ OUTPUT - TRAINING\_METRICS - GAINS/LIFT TABLE (AVG RESPONSE RATE: 76.25 %, AVG SCORE: 76.24 %)

▶ OUTPUT - VARIABLE IMPORTANCES

▼ PREVIEW POJO

 Preview POJO

- 그 외에 밑에 내려보면 더 많은 기능이 있습니다!!!



# R을 통한 자원관리와 활용

R 코딩을 실생활에 적용해보기

데