



SQLJava

Yellowfin

	100%
1-9	1-91
10	10
N	N
N	N
9-1	9-11
	-/
	2
	2
	2
%	%
%	/%
	2
	e
Null	Null
%	%
%	
%	%
%	%
%	%
%	%
	2

<b>10</b>	10
<b>N</b>	N
<b>N</b>	NN
<b>/N%</b>	N%
	N

	2
	2
	2

	10-1
<b>Deviation</b>	
<b>Linear Regression</b>	
<b>Mean</b>	
<b>Median</b>	
<b>Mode</b>	
<b>Moving Average</b>	22
<b>Moving Total</b>	N
<b>Naïve Forecasting</b>	1tt-1
<b>Polynomial Regression</b>	23
<b>Quartile</b>	4-1
<b>Standard Deviation</b>	
<b>Standard Deviation from Mean</b>	
<b>Standard Score</b>	
<b>Stepped Regression</b>	
<b>Trend</b>	
<b>Triple Exponential Smoothing</b>	
<b>Variance</b>	
<b>Weighted Moving Average</b>	

	2
--	---

1,000

**-R**

R

**R**

YellowfinR

<R\_file\_name>.RYellowfin<R\_file\_name>.R.input.csvR<R\_file\_name>.R.result.csv

Neural NetworksRR

### Sample R-Script : neural-net-script.R

```
setwd("C:/R/R-3.2.3/bin/x64")
library(rattle) #
To access the weather dataset and utility commands.
library(magrittr) # For the
%>% and %<>% operators.
building <- TRUE
scoring <- ! building
# A pre-defined value is used
to reset the random seed so that results are repeatable.
crv$seed <- 42
# Load the data.
rPATH <-
Sys.getenv("RSCRIPT_PATH")
rINPUT <- paste0(rPATH, "/neural-net-script.r.input.csv")
rOUTPUT <- paste0(rPATH
, "/neural-net-script.r.result.csv")
dataset <-
read.csv(file=rINPUT, header=FALSE, sep=",")
# Note the user
selections.
# Build the
training/validate/test datasets.
set.seed(crv$seed)
crs$nobs <- nrow(dataset) #
366 observations
crs$sample <- crs$train
<- sample(nrow(dataset), 0.7*crs$nobs) # 256 observations
crs$validate <-
sample(setdiff(seq_len(nrow(dataset)), crs$train), 0.15*crs$nobs) # 54
observations
crs$test <-
setdiff(setdiff(seq_len(nrow(dataset)), crs$train), crs$validate) # 56
observations
# The following variable
selections have been noted.
crs$input <-
c("V1", "V2", "V3", "V4", "V5")
crs$target
<- "V6"
#=====
# Neural Network
#=====
# Build a neural network model
using the nnet package.
library(nnet, quietly=TRUE)
# Build the NNet model.

set.seed(199)
crs$nnet <-
nnet(as.factor(V6) ~ ., data=dataset[crs$sample, c(crs$input, crs$target)], size=10,
skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
#=====
# Score a dataset.
#=====
# Obtain probability scores for
the Neural Net model on weather.csv [validate].
#crs$pr <- predict(crs$nnet,
newdata=dataset[crs$validate, c(crs$input)], type="class")
#crs$pr <- predict(crs$nnet,
newdata=dataset[crs$validate, c(crs$input)], type="class")
crs$pr <- predict(crs$nnet,
newdata=dataset, type="class")
write.table(crs$pr,
file=rOUTPUT, row.names=FALSE, col.names = FALSE)
```

R

Advanced Metrics

Select Function:  
Rserve R

Search

Apply R-Script

Apply R-Script

This function will invoke an R-script (in a separate file) which is pointed to by a parameter. The R-script will return a result value which will be included in the report

Attribute	Setting	User Prompt
R Script File Name	scriptName.r	Name Of The R Script to Invoke
Value For MinTemp	MinTemp	First Parameter
Value For MaxTemp	MaxTemp	Second Parameter
Value For Rainfall	Rainfall	Third Parameter
Value For Evaporation	Evaporation	Fourth Parameter
Rain Today	RainToday	Fifth Parameter
Value For RainTomorrow	RainTomorrow	Sixth Parameter

Step 1: Use "Apply R-Script" to convert to

-

210

1.

a. /

カラム(列)	Year	Σ Sum Invoiced ...
ロウ(行)		

b.

Year	Sum Invoiced Amount
2009	\$17,633,473
2010	\$8,611,470
2011	\$11,012,244
2012	\$81,690,100
2013	\$158,353,519
2014	\$152,912,577
2015	\$28,207,858
2016	\$12,522,605

- 2.
- 3.

高度な関数

高度な関数データ変換

保存キャンセル

請求金額 (表頭)

1234567890

1234567890

1234567890

1234567890

グラフにのみ表示

関数の選択

分析

関数1: 関数A

最上位から 10 位

最上位から N 位

最上位から N 位 (同順位を含む)

最下位から 10 位

最上位から N 位  
選択したフィールドの最上位の値  
を返します

関数

設定

ユーザープロンプト

最上位

10

表示数

ステップ 1: 最上位から N 位 を使用し 数値へ変換

- a.
- b.
- c.
- d. /

グラフにのみ表示

- 4.

-

1. +

高度な関数

Year

Sum Invoiced Amount

+

- 2.
- 3.

- a.
- b.
- c.
- d. /

グラフにのみ表示

- 4.

-

1.

国(顧客)	請求金額 の順位
日本	請求金額 の順位
アメリカ	Σ 集約 ▶
イタリア	並べかえ ▶
オーストラリア	書式 ▶
オーストリア	Σ 高度な関数 ▶
カナダ	合計 ▶
スイス	フィールドを非表示化
スペイン	
ドイツ	7
韓国	6

2.

3.

- a.
- b.
- c.
- d. /

グラフにのみ表示

☒

4.

1.

2.

3.

4. Java1,000

5.

6.