r ml library Cheat Sheet

by Niki (worlddoit) via cheatography.com/170195/cs/36005/

Intro

Additional material about top libraries for Machine Learning in R.

(**Basics)

1. XML

You can read a xml file in R using the "-XML" package.

```
install.packages("XML")
# Also load the other required package.
library("methods")
# Give the input file name to the function.
result <- xmlParse(file = "input.xml")
# Exract the root node form the xml file.
rootnode <- xmlRoot(result)
# Find number of nodes in the root.
rootsize <- xmlSize(rootnode)
```

(*Additional) Resources: 1, 2

2. dplyr

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

`mutate()` adds new variables that are functions of existing variables

`select()` picks variables based on their names.

`filter()` picks cases based on their values.

`summarise()` reduces multiple values down to a single summary.

`arrange()` changes the ordering of the rows.

These all combine naturally with `group_by()` which allows you to perform any operation “by group”.

2. dplyr (cont)

Example:

```
starwars %>%
group_by(species) %>%
summarise( n = n(), mass = mean(mass, na.rm = TRUE)
) %>%
filter( n > 1, mass > 50)
```

(*Additional) Resources: 1, 2

3. xgboost

```
library(xgboost)
# load data
data(agaricus.train, package = 'xgboost')
data(agaricus.test, package = 'xgboost')
train <- agaricus.train
test <- agaricus.test
# fit model
bst <- xgboost(data = train$data, label = train$label, max_depth = 2, eta = 1, nrounds = 2,
nthread = 2, objective = "binary:logistic")
# predict
pred <- predict(bst, test$data)
```

(*Additional) Resources: 1, 2

4. mlr3

Lots of functionality, you can deal with clustering, regression, classification, and survival analysis.

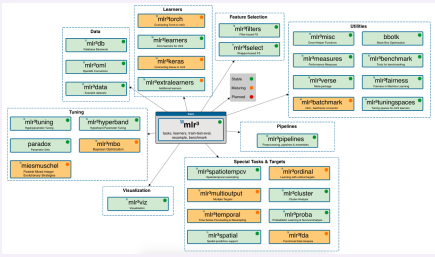
```
library(mlr3)
# create learning task
task_penguins = as_task_classif(species ~ ., data = palmerpenguins::penguins)
# load learner and set hyperparameter
```

4. mlr3 (cont)

```
learner = lrn("classif.rpart", cp = .01)
# train/test split
split = partition(task_penguins, ratio = 0.67)
# train the model
learner$train(task_penguins, split$train_set)
# predict data
prediction = learner$predict(task_penguins, split$test_set)
# calculate performance
prediction$confusion
measure = msr("classif.acc")
prediction$score(measure)
# 3-fold cross validation
resampling = rsmp("cv", folds = 3L)
# run experiments
rr = resample(task_penguins, learner, resampling)
# access results
rr$score(measure)[, .(task_id, learner_id, iteration, classif.acc)]
rr$aggregate(measure)
```

(*Additional) Resources: 1, 2, 3

4. mlr3 additional



(*Additional)

5. knitr

It is reproducible, used for report creation, and integrates with various types of code structures like LaTeX, HTML, Markdown, LyX, etc.
This package is an amazing one, you can make a beautiful pdf report and editable pdf forms with the help of latex coding.

```
kable
xtable
tikzDevice
```

(*Additional) Resources: 1, 2

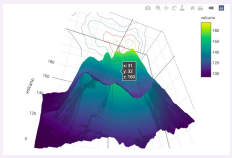
6. plotly

Plotly's R graphing library makes interactive, publication-quality graphs. Example of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, and 3D (WebGL based) charts.

```
library(plotly)
# volcano is a numeric matrix
that ships with R
fig <- plot_ly(z = ~volcano) %>%
add_surface(
contours = list(
z = list( show=TRUE,
usecolormap=TRUE,
highlightcolor="#ff0000",
project=list(z=TRUE) ) ) )
fig <- fig %>% layout(
scene = list( camera= list(
eye = list(x=1.87, y=0.88, z=-
0.64) )))
fig
```

(*Additional) Resources: 1, 2, 3

6. plotly: Output



(*Additional)

7. e1071

Dealing with clustering, Fourier Transform, Naive Bayes, SVM, and other types of modeling data analysis then you can't avoid e1071.
Example:

```
#Author DataFlair
library("e1071")
data("iris")
head(iris)
x <- iris[,-5]
y <- iris[5]
model_svm <- svm(Species ~ .,
data=iris)
summary(svm_model)
pred <- predict(model_svm,x)
confusionMatrix(pred,y$Species)
```

(*Additional) Ressources: 1, 2

8. tidyverse

For data manipulation. Covered in lectures (see previous sections).
Example:

```
library(tidyverse)
library(lubridate)
library(nycflights13)
```

Create a new column basis count option

```
flights %>%
mutate(long_flight = (air_time
>= 6 * 60)) %>%
```

8. tidyverse (cont)

```
View()
```

Randomly Shuffle the data

```
flights %>%
slice_sample(n = 15)
```

(*Additional) Ressources: 1, 2

9. caret

If you are dealing with classification and regression problems then caret is one of the essential packages.
`caret` package is the extension of the caret is CaretEnsemble which is used for combining different models.
Example:
Visualization of feature distribution by class

```
featurePlot(x =
GermanCredit[,c("EmploymentDuratio
"Age")],
y = GermanCredit$Class,
plot = "box")
```

Pre-processing: imputation of missing data, on hot encoding, and normalization

```
set.seed(355)
bagMissing <- preProcess(trainingS
method = "bagImpute")
trainingSet <- predict(bagMissing,
newdata = trainingSet)
dummyModel <- dummyVars(Class ~ .,
data = trainingSet)
trainingSetX <-
as.data.frame(predict(dummyModel,
newdata = trainingSet))
```

(*Additional) Ressources: 1, 2

10. shiny

If you are thinking about an interactive and beautiful web interface then Shiny is the solution.
Shiny interfaces are directly written in R and provide a customizable slider widget that has built-in support for animation.
Example:

```
library(shiny)
# See above for the definitions of ui and server
ui <- ...
server <- ...
shinyApp(ui = ui, server = server)
```

(*Additional) Ressources: 1, 2

10. shiny: Output

(*Additional) Ressources: 1, 2

11. tidyquant

`tidyquant` is considered as a financial package that is used to carry out quantitative financial analysis.
Package tidyquant is also widely used for importing, analyzing, and visualizing data.
Example:

```
library(tidyquant)
google <- tq_get(x = "GOOG")
tq_get_options()
?tq_get
tq_exchange_options()
nyse <- tq_exchange("NYSE")
```

(*Additional) Ressources: 1, 2

12. tidyr

`tidyr` is a new package that makes it easy to “tidy” your data. tidyr package is an evolution of Reshape2.
The data is considered tidy when each variable represents columns and each row represents an observation.
`gather()` makes “wide” data longer
`spread()` makes “long” data wider
`separate()` splits a single column into multiple columns
`unite()` combines multiple columns into a single column
and More...
Example:

```
wide_DF <- unite_DF %>%
spread(Quarter, Revenue)
head(wide_DF, 24)
```

(*Additional) Ressources: 1, 2

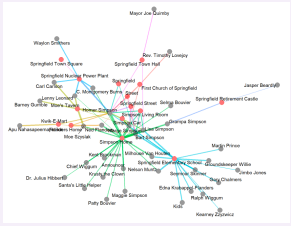
13. ggraph

Takes away all the limitations of ggplot2.
Example for Networks:

```
ggraph(ig_loc, layout="fr") +
geom_edge_link(aes(color =
factor(to), width =
log(weight)), alpha = 0.5,
start_cap = circle(2, 'mm'),
end_cap = circle(2, 'mm')) +
scale_edge_width(range = c(0.5,
2.5)) +
geom_node_point(color =
V(ig_loc)$color, size = 5, alpha
= 0.5) +
geom_node_text(aes(label =
name), repel = TRUE) +
theme_void() +
theme(legend.position = "none")
```

(*Additional) Ressources: 1, 2, 3

13. ggraph: Output



(*Additional)

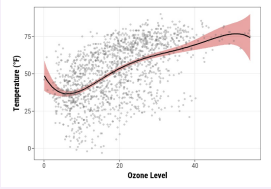
14. ggplot2

`ggplot2` is one of the most popular visualization package in R.
It is famous for its functionality and high-quality graphs that set it apart from other visualization packages.
Example:

```
ggplot(chic, aes(x = o3, y =
temp))+
labs(x = "Ozone Level", y =
"Temperature (°F)") +
geom_smooth(
method = "lm",
formula = y ~ x + I(x^2) +
I(x^3) + I(x^4) + I(x^5),
color = "black",
fill = "firebrick") +
geom_point(color = "gray40",
alpha = .3)
```

(*Additional) Ressources: 1, 2, 3

14. ggplot2: Output



(*Additional)