R and Docker Image Instructions

Documents from GitHub

Download all the needed documents from my Github repo:

https://github.com/XiaohanSun01/Open_Science_Project

Mushroom Classification in R

Explanations of the R Codes

- The analysis is to classify mushrooms into **poisonous (p) or edible (e).**
- The dataset has **16** variables, including `cap.diameter`, `stem.height`, `season`, and so on.
- **Random Forest Classifier** is implemented, and the final classification performance is **99.9%.**

---

1. Import the requested libraries.

```
library(stringr)
library(purrr)
library(ggplot2)
library(caret)
library(randomForest)
```

2. Import the data and edit them to be in the form of a Dataframe.

```
d1 = read.csv("secondary_data_no_miss.csv")
colnames(d1)[1] = "col1"
t <- str_split(d1$col1, ";")
data <- do.call(rbind, t)

colnames(data) <- c('class','cap-diameter','cap-shape','cap-surface',
                    'cap-color','does-bruise-or-bleed','gill-attachment',
                     'gill-spacing','gill-color','stem-height','stem-width',
                    'stem-color','has-ring','ring-type','habitat','season')
data = data.frame(data)

# Change the variables to become factors or integers.
sapply(data, class)cols.num = c("cap.diameter","stem.height","stem.width")
data[cols.num] = sapply(data[cols.num],as.numeric)
data[sapply(data, is.character)] <- lapply(data[sapply(data, is.character)],
                                           as.factor)

levels(data$class) <- c("edible", "poisonous")
str(data)
```

```
'data.frame':   61069 obs. of  16 variables:
 $ class               : Factor w/ 2 levels "edible","poisonous": 1 1 2 2 1 2 2 2 1 1 ...
 $ cap.diameter        : num  1.26 10.32 0.92 4.27 3.08 ...
 $ cap.shape           : Factor w/ 7 levels "b","c","f","o",..: 7 3 7 7 3 6 3 3 7 7 ...
 $ cap.surface         : Factor w/ 11 levels "d","e","g","h",..: 3 2 3 9 8 8 9 6 8 4 ...
 $ cap.color           : Factor w/ 12 levels "b","e","g","k",..: 12 1 8 8 11 12 9 11 6 12
...
 $ does.bruise.or.bleed: Factor w/ 2 levels "f","t": 1 1 1 1 1 1 1 2 2 2 ...
 $ gill.attachment     : Factor w/ 7 levels "a","d","e","f",..: 2 1 1 7 2 4 1 2 3 7 ...
 $ gill.spacing        : Factor w/ 3 levels "c","d","f": 1 1 1 1 2 3 1 2 1 1 ...
 $ gill.color          : Factor w/ 12 levels "b","e","f","g",..: 11 1 8 11 11 3 12 11 6 1
2 ...
 $ stem.height         : num  5.04 4.68 4.59 4.55 2.67 3.2 4.87 6.94 8.03 7.55 ...
 $ stem.width          : num  1.73 19.44 1.15 6.52 5.18 ...
 $ stem.color          : Factor w/ 13 levels "b","e","f","g",..: 13 12 5 12 12 13 13 12 1
2 12 ...
 $ has.ring            : Factor w/ 2 levels "f","t": 1 2 1 1 1 1 1 1 2 1 ...
 $ ring.type           : Factor w/ 8 levels "e","f","g","l",..: 2 2 2 2 2 2 2 2 4 2 ...
 $ habitat             : Factor w/ 8 levels "d","g","h","l",..: 1 1 1 1 5 1 1 1 5 1 ...
 $ season              : Factor w/ 4 levels "a","s","u","w": 1 1 3 1 1 1 1 1 1 3 ...
```

3. Visualize the data

   a. Count the number in classes of `edible` and `poisinous`.

```
table(data$class)
```

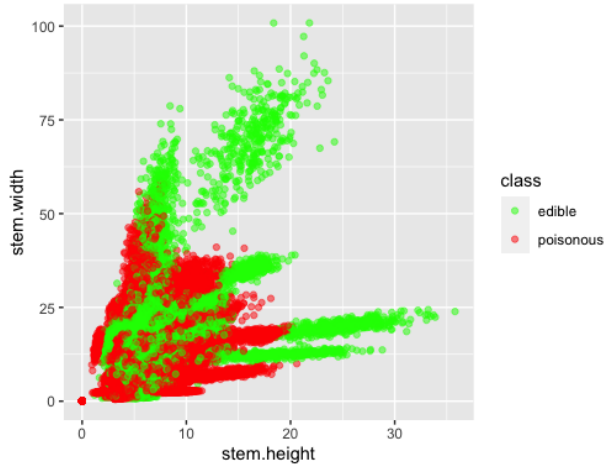
```
> table(data$class)

   edible poisonous
    27181     33888
```

   b. Use a function to compare the class distribution of different variables.

   Note: the seed is set to be `1` here.

```
dataVis <- function(data, x, y, col)
{  x <- rlang::sym(x)
y <- rlang::sym(y)
col <- rlang::sym(col)
ggplot(data = data, aes(x = !!x , y = !!y , col = !!col)) +
geom_jitter(alpha = 0.5) +
scale_color_manual(values = c("green", "red"))}

set.seed(1)

dataVis(data = data, x = 'stem.height', y = 'stem.width', col = 'class')
dataVis(data = data, x = 'habitat', y = 'season', col = 'class')
```

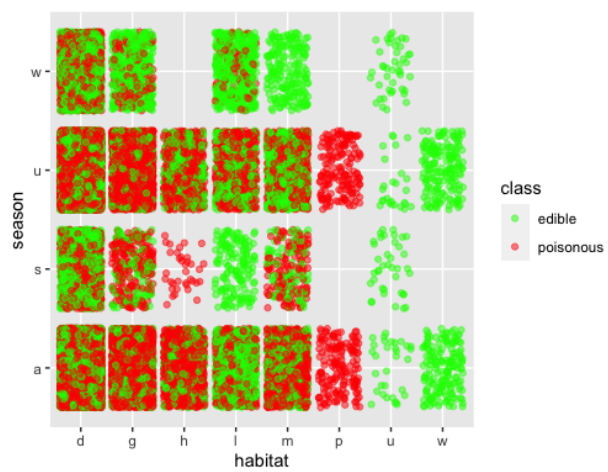
- `stem.height` vs. `stem.width`
  - The figure shows mushrooms with less height and width tend to be **poisonous.**



- `habitat` vs. `season`

  From the comparison of `habitat` and `season`, we can obtain a lot of information such as:

  - Mushrooms with habitats of `urban` and `waste` are always `edible` no matter what season it is.
  - Mushrooms with the habitat of `paths` in the season of `summer` and `autumn` are always `poisonous`.



4. Split the data to have 70% of the training data, and 30% of the test data.

   Note: the seed is set to be `1023` here.

```
# Data Splitting
set.seed(1023)
trainsamples <- createDataPartition(y = data$class, p = 0.7, list = FALSE)
train_mushroom <- data[ trainsamples, ]test_mushroom  <- data[-trainsamples, ]
```

5. Implement the Random Forest Model on the training data.

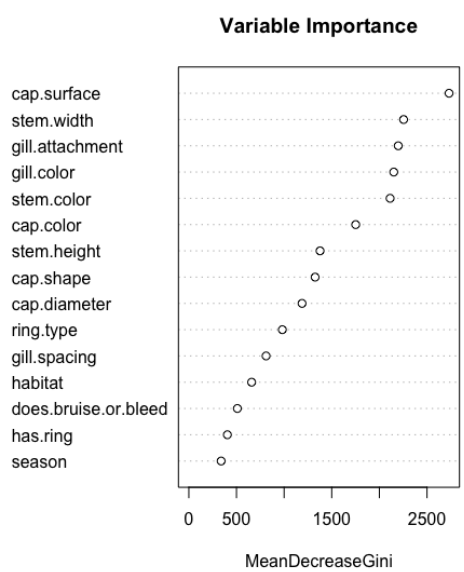
```
# Random Forest Model on the training data.
rf = randomForest(class ~ .,
                  ntree = 100,
                  data = train_mushroom)
print(rf)
```

```
Call:
 randomForest(formula = class ~ ., data = train_mushroom, ntree = 100)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 3

        OOB estimate of  error rate: 0.02%
Confusion matrix:
          edible poisonous  class.error
edible     19023         4 0.0002102276
poisonous      3     23719 0.0001264649
```

6. Interpret the variable importance.

```
# Variable importance
varImpPlot(rf,
          sort = T,
          main = "Variable Importance")

var.imp = data.frame(importance(rf, type=2))
```

- Top variables that influence the classification most: `cap.surface`, `stem.width`, `gill.attachment`.



7. Compute the test performance and save the result to a txt file.

```
sink(file = "mushroom_classification_accuracy.txt")

test_mushroom$predicted.response = predict(rf , test_mushroom)

test_mushroom$predicted.response <- as.factor(test_mushroom$predicted.response)

print(confusionMatrix(data = test_mushroom$predicted.response,
                      reference = test_mushroom$class,
                      positive = 'edible'))
sink()
```

- The final accuracy is almost **100%.**

```
Confusion Matrix and Statistics

           Reference
Prediction  edible poisonous
  edible      8153         0
  poisonous      1     10166

               Accuracy : 0.9999
                 95% CI : (0.9997, 1)
    No Information Rate : 0.5549
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9999

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9999
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9999
             Prevalence : 0.4451
         Detection Rate : 0.4450
   Detection Prevalence : 0.4450
      Balanced Accuracy : 0.9999

       'Positive' Class : edible
```

Run the R file

1. Download the `mushroom_classification.R` and `secondary_data_no_miss.csv` from the GitHub repo.
2. Open the file and set the working directory to **source file location**. Note: Please ensure the R file and the csv file are in the same folder.
3. If there is no problem after the running, you should have the classification result saved to `mushroom_classification_accuracy.txt`. We will also get this file again when we successfully run the Docker image later.

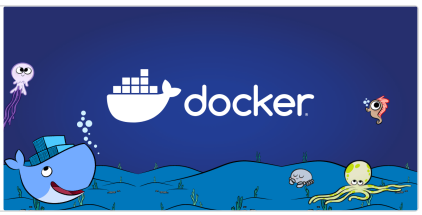
Download the Docker Desktop

1. Use the link below to download Docker Desktop and choose the one that fits your operating system.

Download Docker Desktop | Docker

Install Docker Desktop - the fastest way to containerize applications. The Docker Subscription Service Agreement has been updated. Our Docker Subscription Service Agreement includes a change to the terms for Docker Desktop It remains free for

https://www.docker.com/products/docker-desktop/



2. Follow the instructions to finish the download and open the application.

Docker Image Creation

Docker Images vs. Docker Containers

Before we use Dockerfile to create a docker image, here are two definitions that I want to clarify: Docker images and Docker containers. So, images are the definition of the OS, while the containers are the actual running instances of the images. You will need to **install the image once**, but **multiple containers of the same images can be run simultaneously**.

Dockerfile

From my GitHub repo, here is one document called `Dockerfile`, which is the one to build a docker image. Below are the step-by-step explanations about this document, and you can create your own Dockerfile based on the instructions.

---

1. Describe what image we are building our image from. Here we use `rocker`, a collection of Docker images for R. `4.2.0` is my R version, and you can change it here to your current R version.

```
FROM rocker/r-ver:4.2.0
```

2. Create a directory to receive the analysis. Here is the **directory for the container** and you can change it to your preferred working directory.

```
RUN mkdir /home/analysis
```

3. Use these commands below to install the packages needed from the `mushroom_classification.R`.

```
RUN R -e "install.packages('stringr')"
RUN R -e "install.packages('purrr')"
RUN R -e "install.packages('caret')"
RUN R -e "install.packages('ggplot2')"
RUN R -e "install.packages('randomForest')"
```

4. Now, we need to get the script for the analysis from the machine (host) to the container. For that, we use `COPY localfile pathinthecontainer`.

Note: `mushroom_classification.R` and `secondary_data_no_miss.csv` have to be in the same folder as the `Dockerfile` on your computer.

```
COPY mushroom_classification.R /home/analysis/mushroom_classification.R
COPY secondary_data_no_miss.csv /home/analysis/secondary_data_no_miss.csv
```

5. `CMD` is the command to be run every time you will launch the docker. What we want is `mushroom_classification.R` to be sourced.

   Also, we move the `mushroom_classification_accuracy.txt` from `/home/analysis` to `/home/results` for later export this file from the container to the host.

```
CMD cd /home/analysis \
  && R -e "source('mushroom_classification.R')" \
  && mv /home/analysis/mushroom_classification_accuracy.txt /home/results/mushroom_classification_accuracy.txt
```

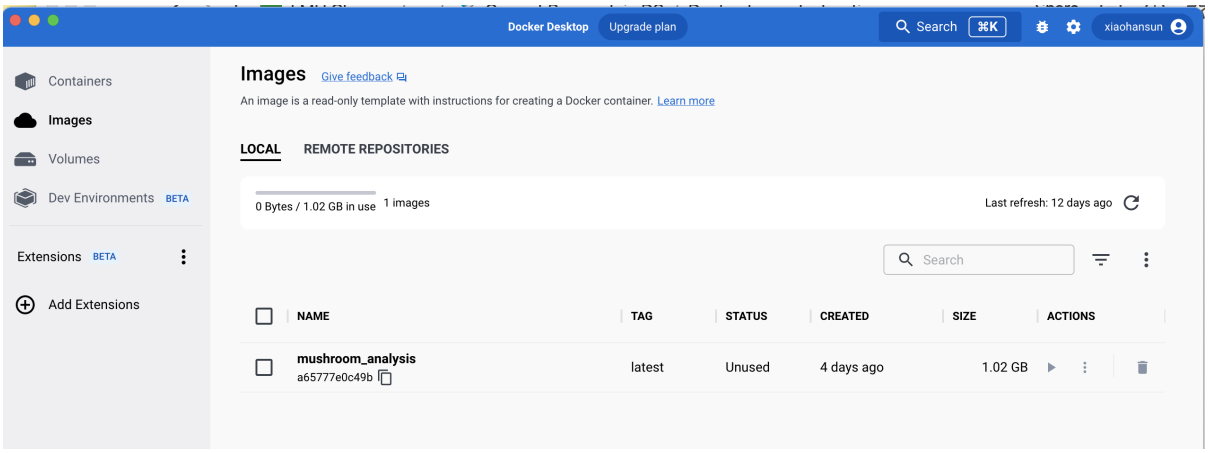
Build and Run

1. Open a terminal and direct to the directory where `Dockerfile` is located.
2. Run the command under the directory above:

```
docker build --build-arg WHEN=2022-01-05 -t mushroom_analysis .
```

   Note: `WHEN` specifies the date when you build this image, `-t name` is the name of the image (here `mushroom_analysis`), and `.` means it will build the `Dockerfile` in the current working directory.

3. If the image is created successfully, it should now appear in the Docker Desktop:



4. Now, let's create a folder `docker_results` under a directory where you want to export the container content. I created the folder under the same folder in which the `Dockerfile` is saved.

```
mkdir your_prefered_directory_here
cd ~/your_prefered_directory_here
mkdir ~/your_prefered_directory_here/docker_results
```

5. Move to the directory of the created folder:

```
cd ~/your_prefered_directory_here/docker_results
```

6. Finally, we run the docker container using `docker run`.

   And to export the container content, we use `-v` flag when running the container, with `/path/from/host:/path/in/container`.

   Last is the image name (Here is `mushroom_classification`).

```
docker run -v ~/your_prefered_directory_here/docker_results:home/results mushroom_classification
```

7. DONE! Your analysis will be run 🎉. You should get the `mushroom_classification_accuracy.txt` again under the folder `docker_results`.

Dockr Package in R

The `dockr` package in R is no longer supported on CRAN, and it is supposed to provide functions to create the Dockerfile instead of writing it from scratch. Based on my exploration, I indeed could use this package to generate a Dockerfile. However, the functions listed from this package is not available and could not be run in R. The codes below listed the **failure.**

---

1. Install the `dockr` package from the R terminal.

```
remotes::install_github("smaakage85/dockr")
```

2. Use the function `prepare_docker_image` to constitute the Docker image. Set up the directory based on you preference (Here, I saved it to `/Users/macbook/Desktop/dockr`).

```
image_dockr <- prepare_docker_image(pkg = package_dir,
                                    dir_image = "/Users/macbook/Desktop/dockr",
                                   dir_install="/Users/macbook/Desktop/dockr")

list.files(image_dockr$paths$dir_image)
list.files(image_dockr$paths$dir_source_packages)
```

- Now the Dockerfile should be generated automatically.

```
> list.files(image_dockr$paths$dir_image)
[1] "Dockerfile"      "source_packages"
> list.files(image_dockr$paths$dir_source_packages)
[1] "dockr_0.8.6.tar.gz"
```

3. Write other lines to the Dockerfile. However, I got stuck from here since R could not find function `write_lines_to_file`.

```
# write lines to the end of the file.
write_lines_to_file(c("# Write lines next."),
                    image_dockr$paths$path_Dockerfile,
                    prepend = FALSE,
                    print_file = FALSE)
```

```
> # write lines to the end of the file.
> write_lines_to_file(c("# Write lines next."),
+                     image_dockr$paths$path_Dockerfile,
+                     prepend = FALSE,
+                     print_file = FALSE)
Error in write_lines_to_file(c("# Write lines next."), image_dockr$paths$path_Dockerfile,
:
  could not find function "write_lines_to_file"
>
```

- I checked the pakcage using `??dockr` from the terminal, and it seemed that this function is not supported anymore.