# Tutorial 1: Introduction to Machine Learning with Python¶

*Jonathan Ish-Horowicz*

*12/01/2020*

## Tutorial 1: Introduction to Machine Learning with Python

The goal of this tutorial is to introduce a typical workflow in carrying out ML in R. Similarly to last week's Python tutorial, this includes:

1. accessing and organising data,
2. assessing the data,
3. visualising the data,
4. a) creating training, b) test datasets and c) learning a model using them and evaluating its performance.

## 1) Load Data

We load the same iris dataset as last week, which has 150 samples and 4 attributes. There are 3 classes (species).

In R we can load the Iris dataset from the `datasets` package:

```
library(datasets)
data(iris)
iris$Species <- as.character(iris$Species)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

## 2) Statistics of the dataset

Now compute the mean, standard deviation, minimum and maximum of each attribute.

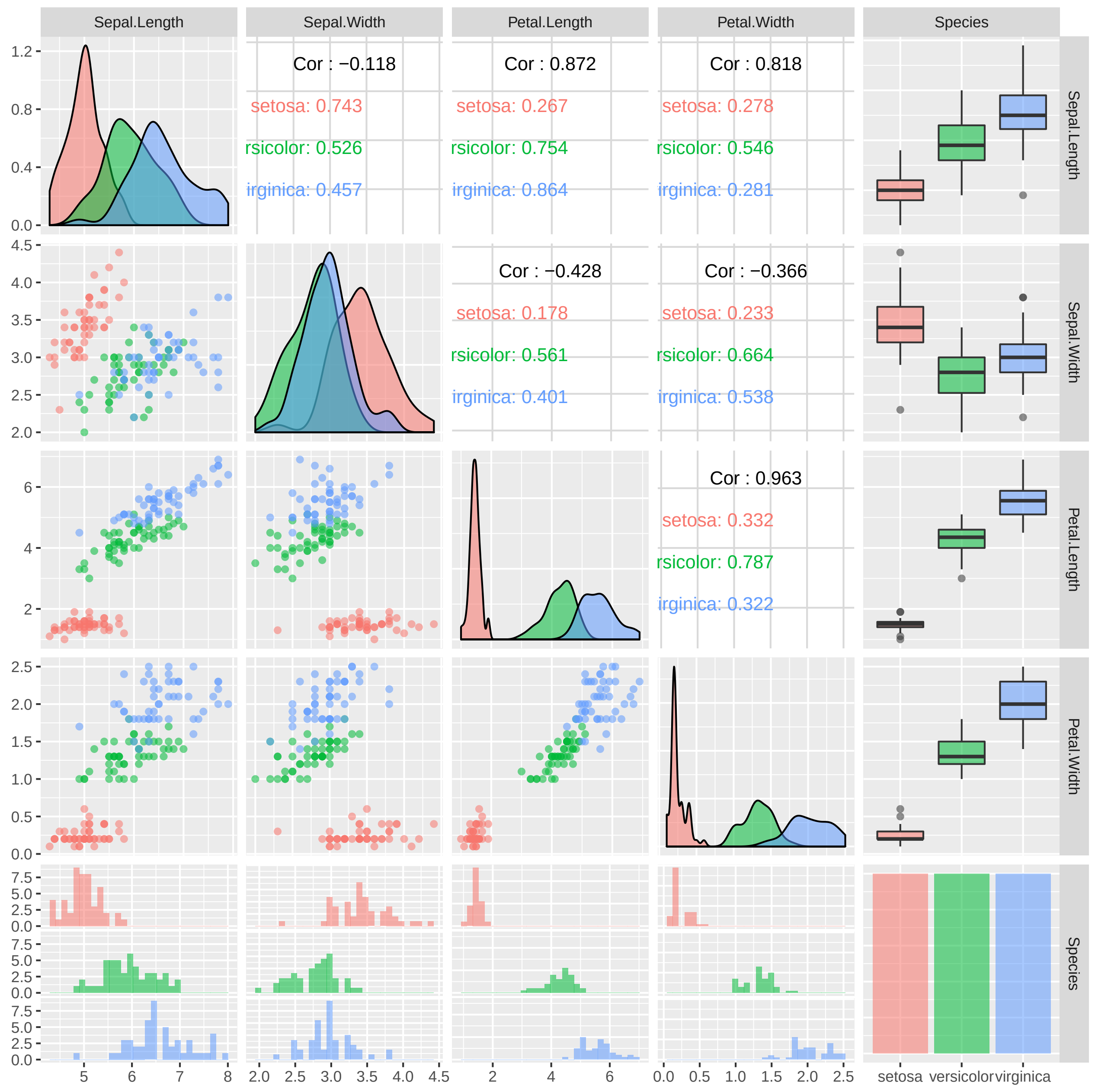
```
library(dplyr)

# Calculate the mean of each attribute
group.means <- iris %>%
  group_by(Species) %>%
  summarise_all(list(mean))
```

```r
# Calculate the standard deviation of each attribute
group.sds <- iris %>%
  group_by(Species) %>%
  summarise_all(list(sd))

# Calculate the minimum of each attribute
group.mins <- iris %>%
  group_by(Species) %>%
  summarise_all(list(min))

# Calculate the maximum of each attribute
group.maxs <- iris %>%
  group_by(Species) %>%
  summarise_all(list(max))
```

## 3) Visualise the dataset

Make some exploratory plots here.

```r
library(ggplot2)
library(GGally)

# A nice pairs plot of the 4 attributes and the class.
# The plot is colored by the species
ggpairs(iris, aes(color=Species, alpha=0.5), progress=FALSE)
```



We can see from the scatter plots that we should be able to separate the classes quite easily.

## 4) Classification using Least Squares

Here we will be carrying out classification using the least squares formulation on 2 classes of the dataset - `setosa` and `versicolor`.

### a) Create separate datasets for the classes `setosa` and `versicolor`.

```
# Extract the rows that have the correct species
setosa <- iris[iris$Species=="setosa",]
```

```
versicolor <- iris[iris$Species=="versicolor",]
```

**b) add a column to each dataset where the column is $1$ if the class is `setosa` and $-1$ otherwise.**

```
# Add a column to each dataframe with the appropriate label
setosa$output <- 1
versicolor$output <- -1
```

**c) create training and test datasets, with 20% of the data for testing. This 80 training points and 20 testing points in total (half this per class).**

```
# Splits a dataframe into training and test sets
train.test.split <- function(df, test.size=0.2) {
  train.idxs <- sample(1:nrow(df),
                       as.integer((1.0-test.size)*nrow(df)),
                       replace=FALSE)
  return(list(train=df[train.idxs,], test=df[-train.idxs,]))
}

# Split each dataset
split.setosa <- train.test.split(setosa)
split.versicolor <- train.test.split(versicolor)

# Combine the two classes
training.data <- rbind(split.setosa$train, split.versicolor$train)
test.data <- rbind(split.setosa$test, split.versicolor$test)
```

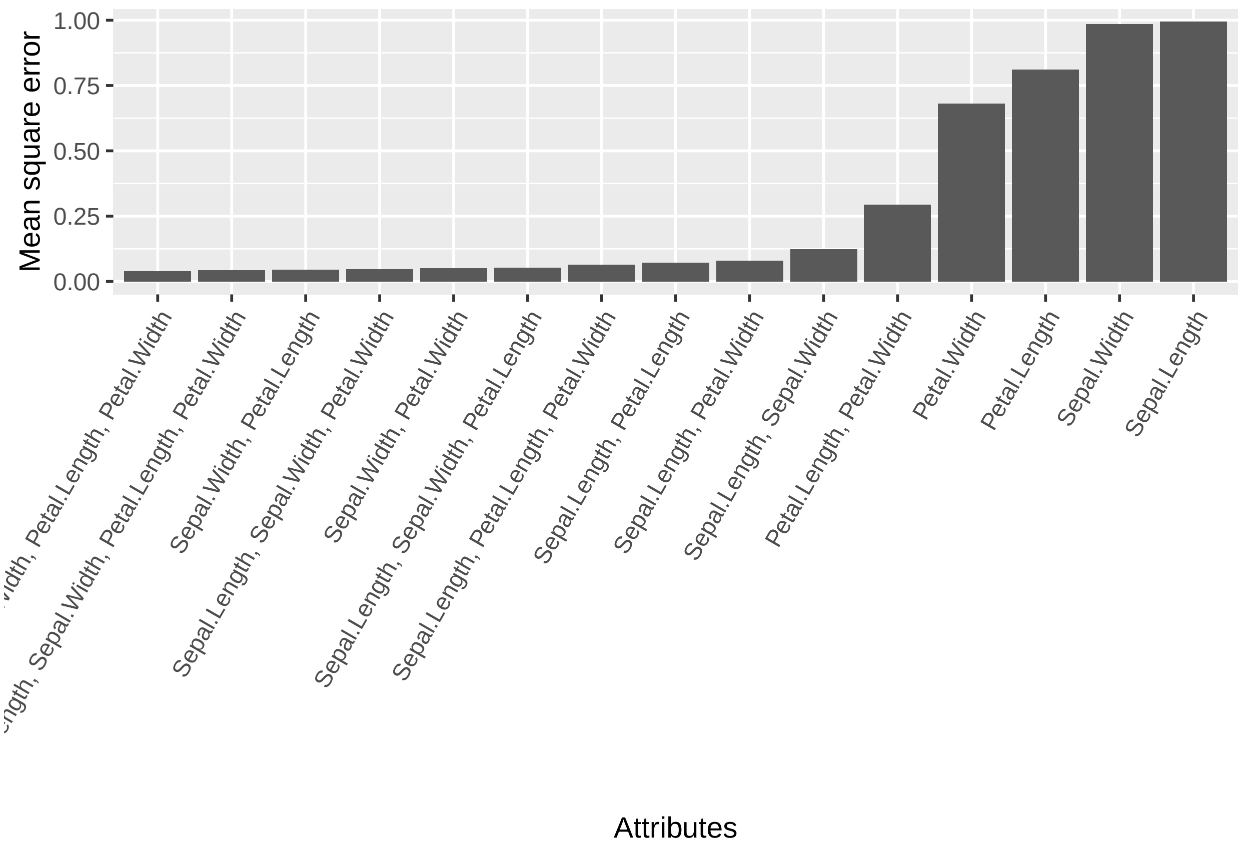
**d) apply the least squares solution to obtain an optimal solution for different combinations of the 4 available attributes.**

```
# Creates all possible combinations of attributes
# attribute.combinations is a list whose elements are lists of attributes
attribute.names <- colnames(iris)[1:4]
attribute.combinations <- do.call(
  c,
  lapply(1:4, function(i) as.list(
    data.frame(combn(attribute.names, i), stringsAsFactors=FALSE)))
  )
names(attribute.combinations) <- 1:length(attribute.combinations)

# Now fit the least squares model and make predictions
return.predictions <- function(attribute.names, training.data, test.data) {
  # Format training and test data (as matrices)
  X <- as.matrix(training.data[attribute.names])
  y <- as.matrix(training.data$output)
  X.test <- as.matrix(test.data[attribute.names])

  # Calculate optimal weights
  W.opt <- solve(t(X) %*% X, t(X) %*% y)
```

```r
  # Make predictions
  predictions <- X.test %*% W.opt
  return(predictions)
}
```

**e) evaluate which input attributes are the best.**

```r
# Calculate the mean square error between some predictions and
# the corresponding testing data
return.mse <- function(predictions, test.data) {
  y.test <- as.numeric(test.data$output)
  error  <- y.test - predictions
  square.error <- error**2.0
  mse <- mean(square.error)
  return(mse)
}

# Calculate the test MSE for each the elements of attribute.combinations
# by calling return.predictions and return.mse

# MSE as a dataframe
mse.df <- as.data.frame(
  sapply(
    attribute.combinations,
    function(a.names) return.mse(
      return.predictions(
        a.names, training.data, test.data),
      test.data
      )
    )
  )
colnames(mse.df)[[1]] <- "mean.square.error"

# attribute.combinations is a list of lists, but for plotting we want to flatten
# it into a single list
mse.df$attributes <- sapply(attribute.combinations, function(x) paste(x, collapse=", "))

# Plot the result
ggplot(mse.df, aes(x=reorder(attributes, mean.square.error), y=mean.square.error)) +
  geom_bar(stat="identity") +
  theme(axis.text.x=element_text(angle=60, hjust=1)) +
  xlab("Attributes") + ylab("Mean square error")
```



Mean square error
1.00
0.75
0.50
0.25
0.00
Width, Petal.Length, Petal.Width
Length, Sepal.Width, Petal.Length, Petal.Width
Sepal.Width, Petal.Length
Sepal.Length, Sepal.Width, Petal.Width
Sepal.Width, Petal.Width
Sepal.Length, Sepal.Width, Petal.Length
Sepal.Length, Petal.Length, Petal.Width
Sepal.Length, Petal.Length
Sepal.Length, Petal.Width
Sepal.Length, Sepal.Width
Petal.Length, Petal.Width
Petal.Width
Petal.Length
Sepal.Width
Sepal.Length
Attributes