MODUL 10
DATA MODELLING DAN APLIKASI WEB SEDERHANA MENGGUNAKAN SHINY

A. Tujuan Praktikum

- Memahami langkah-langkah pemodelan *machine learning*: *supervised* dan *unsupervised* dan melakukan prediksi menggunakan model tersebut
- Membuat aplikasi web sederhana menggunakan `shiny` untuk visualisasi data dan *machine learning*.

B. Alokasi Waktu

1 x pertemuan = 120 menit

C. Dasar Teori

Data modelling dengan menggunakan pembelajaran mesin (*machine learning*) merupakan aktivitas yang seringkali dilakukan dalam siklus *data science*. Sangat menarik apabila memiliki kemampuan untuk membuat prediksi suatu hal sehingga kita dapat melakukan persiapan lebih matang. Bahasa pemrograman R memiliki fitur bawaan yang sangat mumpuni untuk melakukan pembelajaran mesin. Selain itu terdapat sangat banyak paket-paket tambahan lain yang dapat melengkapi dan memudahkan pengguna untuk melakukan prediksi dengan menggunakan metode-metode mutakhir.

Namun keragaman dan banyaknya jumlah paket tersebut juga menyebabkan adanya hambatan lain, yaitu hambatan pada alur kerja. Seringkali beberapa paket memiliki tujuan dan metode yang sama namun menggunakan istilah berbeda dalam argumen serta keluaran fungsinya. Selain itu juga tidak jarang antar paket memiliki API yang berbeda-beda sehingga pengguna harus menyesuaikan bentuk data sebelum dapat digunakan oleh paket lainnya.

Supervised learning

Dalam modul ini kita akan memanfaatkan sebuah sistem bernama `tidymodels` yang dirancang untuk mengatasi permasalahan yang disebutkan diatas sehingga pengguna dapat melakukan pembelajaran mesin dengan lebih mudah. Silakan Anda aktifkan paket tersebut terlebih dahulu beserta paket `vroom` dan `here`.

```
library(__________)
library(__________)
library(__________)
```

Pada modul ini, kita akan langsung belajar menerapkan *data modelling* melalui studi kasus. Dalam studi kasus pertama, kita akan melakukan pemodelan dan prediksi untuk nilai ujian nasional tingkat SMP di Kota Bandung. Imporlah "*un_smp.csv*" yang berada dalam subdirektori data-raw dan simpan sebagai obyek R bernama '*un_smp*'. Selanjutnya jalankan fungsi `glimpse` pada '*un_smp*' untuk mempelajari strukturnya. `glimpse()` adalah fungsi dari paket `dplyr` yang serupa dengan fungsi `str()`

```
__________<- un_smp %>%
  mutate(tahun = as.character(tahun))
__________ (un_smp)
```

Mari kita coba membuat perbandingan nilai ujian IPA antara sekolah negeri dan swasta. Gunakan *script* berikut:

```
un_smp %>%
  group_by(status) %>%
  summarise(ipa = mean(ipa)) %>%
  ggplot(aes(x = status, y = ipa)) +
  geom_col() +
  labs(
    x = "Status sekolah",
    y = "Rerata nilai ujian IPA",
    title = "SMP Negeri vs SMP Swasta",
    subtitle = "Ujian Nasional SMP di Kota Bandung 2015-2019",
    caption = "Sumber: Open Data Kota Bandung"
  ) +
  theme_light()
```

Melalui grafik yang dihasilkan pada *script* diatas, sebagai analisa awal, dapat disimpulkan bahwa terdapat kemungkinan adanya pengaruh dari status sekolah terhadap nilai ujian IPA. Selanjutnya, kita akan membuat grafik yang serupa untuk mempelajari kemungkinan pengaruh dari tahun ujian dan menganalisa kesimpulan apa yang dapat kita peroleh melalui grafik yang kita buat.

```
un_smp %>%
  group_by(tahun) %>%
  summarise(ipa = mean(ipa)) %>%
  ggplot(aes(x = tahun, y = ipa)) +
  geom_col() +
  labs(
    x = "Tahun pelaksanaan",
    y = "Rerata nilai ujian IPA",
    title = "SMP Negeri vs SMP Swasta",
    subtitle = "Ujian Nasional SMP di Kota Bandung 2015-2019",
    caption = "Sumber: Open Data Kota Bandung"
  ) +
  theme_light()
```

Sebagai analisa lanjut, kita akan melakukan tahap pertama dari pembelajaran mesin, yaitu membagi data yang kita miliki ke dalam dua kelompok. Kelompok pertama disebut *data training* yang akan digunakan untuk membuat pemodelan. Sedangkan kelompok kedua disebut *data testing* yang akan digunakan untuk menguji performa model yang kita buat. Kita akan memanfaatkan Fungsi `initial_split` dari paket `rsample` untuk melakukan hal tersebut. Berapakah komposisi *data training* terhadap *data testing* bawaan pada fungsi tersebut?

```
set.seed(270719)
un_smp_split <- initial_split(un_smp)
un_smp_split
```

‘*un_smp_split*’ merupakan obyek R yang memiliki metadata atau informasi baris mana dari data ‘*un_smp*’ yang akan digunakan sebagai *data training* dan *data testing*. Kita dapat mengakses

data training dengan cara menjalankan fungsi `training()` pada '*un_smp_split*'. Selanjutnya, *data testing* diperoleh dengan cara menjalankan fungsi `testing()` pada '*un_smp_split*'.

Data training

```
training(un_smp_split)
```

Data testing

```
__________ (__________)
```

Sebelum membuat pemodelan, biasanya kita akan melakukan *pre-processing* pada data yang kita miliki terlebih dahulu. Hal tersebut penting dilakukan agar data tersebut dapat memenuhi prasyarat sehingga algoritma dari model dapat berjalan dengan baik dengan performa bagus. Contoh: pada metode regresi linear diharapkan tidak terjadi *multicolinearity* antar variabel prediktor; pada beberapa metode klasifikasi diharapkan agar skala yang digunakan adalah sama untuk semua prediktor; dan lain-lain.

Dalam `tidymodels` tahap *pre-processing* tersebut dianalogikan sebagai "resep" sehingga dapat mudah dipahami. Kita dapat menggunakan fungsi `recipe`, `update_role`, dan `step_*` dari paket `recipes` untuk membuat "resep" tersebut. Perhatikan baris *script* berikut untuk membuat "resep" berdasarkan *data training*

```
un_smp_recipe <- training(un_smp_split) %>%
  recipe() %>%
  update_role(ipa, new_role = "outcome") %>%
  update_role(tahun, status, jumlah_peserta, bahasa_indonesia,
bahasa_inggris, matematika, new_role = "predictor") %>%
  update_role(nama_sekolah, new_role = "ID") %>%
  step_corr(all_predictors(), -tahun, -status)

un_smp_recipe
summary(un_smp_recipe)
```

Berdasarkan *output* fungsi di atas informasi apakah yang dapat kita peroleh?

Tahap selanjutnya adalah kita harus menerapkan "resep" yang telah di buat pada data yang kita miliki, baik pada *data training* maupun *data testing*. Hal tersebut dapat dilakukan dengan menggunakan fungsi `prep` dan `bake` yang juga berasal dari paket `parsnip`. Setelah "resep" diterapkan kita perlu menyimpan hasilnya sebagai obyek R untuk selanjutnya malakukan pemodelan dan evaluasi performa. Perhatikan baris *script* berikut untuk menerapkan "resep" serta menyimpannya obyek R dengan nama '*un_smp_training*' dan '*un_smp_testing*'. Jalankan `glimpse` pada dua obyek tersebut. Apakah ada perbedaan kedua obyek tersebut dengan `training(un_smp_split)` dan `testing(un_smp_split)`?

```
un_smp_training <- un_smp_recipe %>%
  __________ () %>%
  __________ (training(un_smp_split))
__________(__________)

un_smp_testing <- un_smp_recipe %>%
  __________ () %>%
  __________ (testing(un_smp_split))
__________ (__________)
```

Setelah membagi data awal menjadi dua, *data training* dan *data* testing, serta menerapkan "resep" *pre-processing* pada keduanya, sekarang saatnya kita melakukan pemodelan. Dalam studi kasus ini kita akan menggunakan model *Linear Regression*.

Pertama kita akan membuat model regresi sebagai berikut:

```
un_smp_lm <- linear_reg(mode = "regression") %>%
  set_engine("lm") %>%
  fit(ipa ~ . -nama_sekolah, data = un_smp_training)

un_smp_lm
```

Selanjutnya kita akan menguji performa model tersebut dengan menggunakan fungsi `metrics` dari paket `yardstick`. Pengujian dilakukan terhadap *data testing* seperti contoh di bawah ini:

```
un_smp_lm %>%
  predict(un_smp_testing) %>%
  bind_cols(un_smp_testing) %>%
  metrics(truth = ipa, estimate = .pred)
```

Unsupervised learning

Kita mungkin sudah terbiasa membaca narasi teks, seperti artikel, cerita pendek, cuitan *twitter*, atau bahkan buku dan novel. Seringkali secara tidak sadar kita melakukan pengkategorian atas narasi-narasi yang dibaca. Bagaimana caranya melakukan pengkategorian tersebut dengan menggunakan mesin/komputer? Dalam studi kasus selanjutnya, kita akan melakukan pemodelan topik (*topic modeling*) untuk mengkategorikan topik dari buku cerita. Kita akan menggunakan beberapa paket untuk melakukan hal tersebut, diantaranya adalah `tidytext` dan `topicmodels`. Selain kedua paket tersebut aktifkanlah juga paket `vroom`, `here`, `dplyr`, dan `ggplot2`.

```
library(__________)
library(__________)
library(__________)
library(__________)
library(__________)
library(__________)
```

Pada subdirektori data-raw terdapat berkas bernama "*sherlock.csv*". Imporlah berkas tersebut menjadi obyek R bernama '*sherlock*'. Kita akan melakukan pengkategorian topik buku "*The Adventures of Sherlock Holmes*" karya Arthur Conan Doyle. Jangan lupa untuk melakukan inspeksi terhadap data '*sherlock*' tersebut.

```
sherlock <- __________ (__________ ("data-raw", "sherlock.csv"))
__________ (sherlock)
```

Data "*sherlock"* tersebut berisi kolom '*story*' yang merupakan subcerita dari cerita keseluruhan dan kolom '*text*' yang merupakan naskah cerita pada setiap baris. Sekarang kita akan melakukan transformasi data untuk mencatat kata apa saja yang muncul untuk setiap subcerita. Jalankan *script* berikut dan simpanlah hasilnya sebagai obyek R bernama '*sherlock_tidy*'

```
__________<- sherlock %>%
  filter(!is.na(text)) %>%
  group_by(story) %>%
  unnest_tokens(word, text) %>%
  ungroup() %>%
  anti_join(stop_words)

sherlock_tidy
```

Analisa apa saja yang dilakukan pada setiap tahap transformasi data "*sherlock*" menjadi '*sherlock_tidy*'? Apa isi dari '*stop_words*'?

Kita dapat menghitung frekuensi penggunaan kata untuk setiap subcerita dengan menggunakan fungsi `count`. Tambahkanlah argumen untuk mengurutkan frekuensi kata '*n*' dari paling tinggi hingga paling rendah

```
sherlock_tidy %>%
  __________ (story, word, __________)
```

Selanjutnya kita dapat dapat mengetahui kata penting apa yang terdapat dapat suatu subcerita dengan menggunakan analisis *Term Frequency - Inverse Document Frequency* (tf-idf). Fungsi `bind_tf_idf` dapat dimanfaatkan untuk melakukan hal tersebut. Simpanlah hasil keluaran fungsi tersebut dalam obyek bernama '*sherlock_tfidf*'

```
__________<- sherlock_tidy %>%
  __________ (story, word, __________) %>%
  bind_tf_idf(word, story, n)

sherlock_tfidf
```

'*sherlock_tfidf*' tersebut akan lebih mudah dicerna jika ditampilkan dalam bentuk grafik. Pada *script* berikut kita akan memvisualisasikan data tersebut dengan menampilkan 15 kata terpenting untuk subcerita "*ADVENTURE I. A SCANDAL IN BOHEMIA*"

```
sherlock_tfidf %>%
  filter(story == "ADVENTURE I. A SCANDAL IN BOHEMIA") %>%
  top_n(15, tf_idf) %>%
  ggplot(aes(x = reorder(word, tf_idf), y = tf_idf)) +
  geom_col() +
  coord_flip() +
  labs(
    x = "",
    y = "tf-idf",
    title = "Kata terpenting pada cerita Sherlock Holmes",
    subtitle = "Subcerita 'Adventure I. A Scandal in Bohemia'"
  ) +
  theme_light()
```

Berdasarkan analisis tf-idf di atas kita dapat melihat bahwa masih banyak kata yang merupakan nama tokoh dalam cerita termasuk dalam kata penting. Apakah sebaiknya kita menghapus kata-kata tersebut terlebih dahulu sebelum melakukan pemodelan topik? Sekarang kita akan mulai melakukan pemodelan topik. Adapun algoritma yang akan kita pergunakan adalah *Latent Dirichlet allocation* (LDA). LDA merupakan algoritma yang biasa digunakan dalam pemodelan

topik. Untuk menjalankan algoritma LDA dengan paket `topicmodels`, '*sherlock_tidy*' harus diubah menjadi obyek berjenis *DocumentTermMatrix* dengan cara sebagai berikut:

```
sherlock_dtm <- sherlock_tidy %>%
  count(story, word) %>%
  cast_dtm(story, word, n)

sherlock_dtm
```

Selanjutnya kita dapat mengimplementasikan algoritma LDA dengan menggunakan fungsi `LDA`. Pada fungsi ini kita harus menentukan nilai '*k*', yaitu jumlah kategori topik yang diinginkan. Sebagai contoh kita akan menggunakan nilai *k*= 5.

```
sherlock_lda <- LDA(sherlock_dtm, k = 5)

sherlock_lda
```

Bagaimanakah kategorisasi subcerita Sherlock Holmes dalam 5 topik yang telah kita buat pemodelannya tersebut? Kita dapat mengetahuinya dengan cara mengamati peluang suatu topik per dokumen yang dinyatakan sebagai nilai `$gamma$`. Fungsi `tidy` dari paket `broom` dapat digunakan untuk melakukan hal tersebut. Perhatikan contoh di bawah ini:

```
sherlock_gamma <- sherlock_lda %>%
  tidy(matrix = "gamma") %>%
  rename(story = document) %>%
  arrange(story, desc(gamma))

sherlock_gamma
```

Kita juga dapat membuat visualisasi untuk '*sherlock_gamma*' seperti ditunjukan pada *script* di bawah. Silakan berikan kostumisasi pada grafik tersebut.

```
sherlock_gamma %>%
  ggplot(aes(x = rev(story), y = gamma, fill = factor(topic))) +
  geom_col() +
  coord_flip() +
  labs(
    x = ",",
    y = expression(gamma),
    title = "Subcerita Sherlock Holmes berdasarkan topik",
    fill = "Topik"
  ) +
  theme_light()
```

Agar dapat memahami makna dari setiap topik, kita dapat menghimpun kata-kata apa saja yang menjadi kunci dalam suatu topik. Hal tersebut dapat dilakukan dengan cara mengekstrak probabilitas kata dalam suatu topik yang dinyatakan sebagai nilai `$beta$`. Dalam *script* berikut kita akan menggunakan fungsi `tidy` dari paket `broom` untuk mengekstrak nilai `$beta$` dan selanjutnya menampilkan 10 kata teratas dari setiap topik:

```
sherlock_beta <- sherlock_lda %>%
  tidy(matrix = "beta") %>%
  rename(word = term) %>%
  arrange(topic, desc(beta))

sherlock_beta

sherlock_beta %>%
  group_by(topic) %>%
  top_n(10, beta)
```

shiny

Pada bagian modul ini, kita akan membuat aplikasi *web* sederhana untuk menampilkan grafik kualitas udara di Kota Bandung selama periode tertentu. Kita akan mencoba membuat dan memodifikasi aplikasi web dengan menggunakan dokumen R Markdown sebagai *sandbox*. Kita akan menggunakan empat *script* utama untuk menghasilkan aplikasi web dengan menggunakan `shiny`. Masing-masing *script* tersebut akan kita beri nama sebagai '*global*', '*ui*', '*server*', dan '*run-app*'.

Pada *script* pertama, kita akan mengaktifkan paket-paket yang diperlukan sehingga aplikasi *web* dapat berjalan dengan baik. Paket yang kita perlukan adalah `shiny`, `here`, `vroom`, `dplyr`, `ggplot2`, dan `plotly`. Selain itu, kita juga akan mengimpor berkas "*udara_bandung.csv*" yang berada dalam subdirektori data-raw sebagai obyek R bernama '*udara_bandung*'. '*udara_bandung*' akan menjadi obyek global yang nanti akan dipergunakan oleh aplikasi *web*. Selanjutnya kita akan membuat dua obyek global lain yaitu '*option_station*' dan '*option_parameter*'.

```
library(__________)
library(__________)
library(__________)
library(__________)
library(__________)
library(__________)

__________<- vroom(__________("__________", "__________"))
__________<- unique(udara_bandung[["station"]])
__________<- colnames(udara_bandung)[-c(1, 2)]
```

Berikutnya kita akan menuliskan kode antarmuka (*front-end*/*user interface*) pada *script* UI. Terdapat berbagai jenis tata letak laman antarmuka yang disediakan oleh paket `shiny` maupun oleh paket tambahan lainnya. Dalam studi kasus kali ini kita akan menggunakan `fluidPage` dan `sidebarLayout` untuk mengatur tata letak laman aplikasi *web*. Tambahkan pula judul yang sesuai pada `headerPanel`. Selanjutnya tambahkanlah satu `selectInput` dengan argumen `inputId = "parameter"`, `label = "Parameter"`, dan `choices = option_parameter`. Buat satu `textInput` tambahan yang berfungsi untuk mengatur judul pada grafik. Serta tambahkan `plotlyOutput` pada `mainPanel` dengan `outputId = "aq_plot"` sebagai argumen.

Selanjutnya kita perlu menuliskan kode logika aplikasi yang dieksekusi oleh aplikasi *web* untuk melakukan pemrosesan data. *Script* ketiga merupakan area yang kita gunakan untuk

mengembangkan *back-end* aplikasi *web*. Disini kita harus mendefinisikan obyek R berupa fungsi dengan tiga argumen, yaitu: `input`, `output`, dan `session`. Kemudian di dalam badan fungsi tersebut kita akan menempatkan seluruh kode pemrosesan data. Kita akan menggunakan `input` yang telah dituliskan pada bagian '*ui*' sebagai "bahan baku" dan kemudian menampilkan hasilnya pada `output`. Konten `input` dapat kita akses di `input$inputId`, sedangkan komponen `output` dapat kita simpan pada `output$outputId`. Pastikan semua Id pada script ketiga sama dengan yang terdapat di `script` '*ui*'!

Setelah selesai menuliskan komponen '*global*', '*ui*', dan '*server*'. Sekarang saatnya Anda mencoba aplikasi *web* yang telah kita kembangkan. Jalankanlah baris kode pada *script* terakhir.

```
ui <- fluidPage(
  title = "_____________________________________",
  headerPanel("Air Quality of Kota Bandung"),
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "station",
                  label =  "Station",
                  choices = option_station,
                  multiple = TRUE,
                  selected = option_station[[1]]),
      _____________________________________________,
                  _____________________________________,
                  _____________________________________,
                  _____________________________________,
      textInput(inputId = "parameter_label",
                label = "Label for parameter",
                value = ""),
      _____________________________________,
                _____________________________,
                _____________________________)
    ),
    mainPanel(
      plotlyOutput(outputId = "aq_plot")
    )
  )
)
```

```
server <- function(input, output, session) {
  aq_plot <- reactive({
    udara_bandung %>%
      filter(station %in% input$station) %>%
      ggplot(aes_string(x = "__________", y = _____$__________, colour =
"station")) +
      geom______() +
      scale_x_date(date_breaks = "2 weeks") +
      labs(
        x = "",
        y = ifelse(_____$parameter_label == "", _____$parameter,
_____$parameter_label),
        colour = "Station",
        title = input$__________
      ) +
      theme_light()
  })

  output$__________ <- renderPlotly({
    ggplotly(__________())
  })
}
```

```
shinyApp(ui = ui, server = server, options = list(height = "500px"))
```

D. Latihan

1. Berdasarkan grafik SMP Negeri vs SMP Swasta di atas, analisis lain yang dapat dilakukan adalah adanya kemungkinan pengaruh dari status sekolah terhadap nilai ujian IPA. Buatlah grafik yang serupa untuk mempelajari kemungkinan pengaruh dari tahun ujian.
2. Buatlah pemodelan dan pengujian performa *supervised learning* menggunakan Random Forest! (Petunjuk: gunakan fungsi `rand_forest` dan "mesin" : `ranger`)
   Kesimpulan apa yang dapat di ambil berdasarkan hasil analisis yang dilakukan?
3. Pilihlah dua subcerita lain dari data "*sherlock.csv*" dan buatlah grafik serupa sesuai contoh yang telah dibahas diatas!
4. Buatlah visualisasi untuk 3 topik '*sherlock_beta*'!
   (Petunjuk: baris kode serupa dengan kode untuk membuat visualisasi `sherlock_tfidf`)
5. Buat aplikasi *web* menggunakan bahasa pemrograman R. Aplikasi *web* yang akan kita buat sekarang memiliki tujuan untuk menyajikan model pembelajaran mesin kepada pengguna. Skenarionya adalah kita membuat sistem prediksi nilai ujian mata pelajaran IPA untuk ujian nasional tingkat SMP di Kota Bandung.
   Instruksi:
   Pelajari kembali *script-script* sebelumnya yang pernah kita gunakan pada langkah pembuatan `shiny` untuk visualisasi data
   Buat sistem *deploy* aplikasi *web* seperti contoh tersebut.

   a. Tambahkan sebuah direktori baru bernama `ipa` di bawah subdirektori `inst`
   b. Di dalam subdirektori `run_ipa` buatlah tiga R *script* baru dengan nama masing-masing adalah "*global.R*", "*ui.R*", dan "*server.R*" (Petunjuk: menu "File -- New File -- R Script")
   c. Buatlah R *Script* baru dan simpan dalam direktori "R" dengan nama berkas "run-ipa.R". Salinlah baris kode berikut pada R *Script* tersebut.

```
run_ipa <- function() {
  app_dir <- system.file("ipa", package = "tidyds")
  shiny::runApp(app_dir, display.mode = "normal")
}
```

   d. Jalankanlah baris kode berikut:

```
?initial_split
```