KUIS PRAK DS

```
library(dslabs)
library(tidyverse)
data(gapminder)
```

2. Tampilkan banyak baris/entri dan semua nama kolom/variabel pada dataset tersebut!

```
length(gapminder$country)
names(gapminder)
```

3. Tampilkan tipe data dari kolom/variabel "continent"!

```
class(gapminder$continent)
```

4. Tampilkan ada berapa negara berbeda pada dataset tersebut! Jangan hitung "Israel"!

```
gapminder_without_israel = gapminder %>% filter(country != "Israel")
length(unique(gapminder_without_israel$country))
```

5. Tampilkan nama kolom/variabel dari dataset tersebut yang terdapat nilai NA!

```
for (col in names(gapminder)) {if (any(is.na(gapminder[[col]]))) {print(col)}}
```

6. Untuk setiap kolom/variabel dari dataset tersebut yang terdapat nilai NA, buatlah sebuah vector yang menyimpan nilai-nilai dari kolom tersebut. Jika nilai sel di kolom tersebut adalah NA, simpan angka 0 ke dalam vector. Nama vector harus mengikuti format "vector_namaKolom", di mana "namaKolom" adalah nama kolom dari dataset. Contoh: vector_infant_mortality. Tampilkan 5 data dari masing-masing vector tersebut!

```
vector_infant_mortality = ifelse(is.na(gapminder$infant_mortality), 0, gapminder$infant_mortality)
head(vector_infant_mortality, 5)
vector_fertility = ifelse(is.na(gapminder$fertility), 0, gapminder$fertility)
head(vector_fertility, 5)
vector_population = ifelse(is.na(gapminder$population), 0, gapminder$population)
head(vector_population, 5)
vector_gdp = ifelse(is.na(gapminder$gdp), 0, gapminder$gdp)
head(vector_gdp, 5)
```

7. Ubah setiap kolom/variabel dari dataset tersebut yang terdapat nilai NA, dengan nilai masing-masing vector yang telah dibuat di No.6!

```
gapminder = gapminder %>% mutate(infant_mortality=vector_infant_mortality)
gapminder = gapminder %>% mutate(fertility=vector_fertility)
gapminder = gapminder %>% mutate(population=vector_population)
gapminder = gapminder %>% mutate(gdp=vector_gdp)
```

8. Tampilkan 10 data dari "continent" Asia atau Africa diurutkan menurun berdasarkan "gdp" pada tahun 1998! Tampilkan kolom/variabel "country" dan "gdp" saja!

```
gapminder %>% filter(continent == "Asia" | continent == "Africa") %>% filter(year == 1998) %>% arrange(desc(gdp)) %>% select(country, gdp) %>% head(10)
```

9. Carilah rata-rata untuk "gdp" dari "continent" Asia yang dikelompokkan berdasarkan "region" (ada 2 kolom dengan nama "region" dan "average_gdp")!

```
gapminder %>% filter(continent == "Asia") %>% group_by(region) %>% summarize(average_gdp=mean(gdp))
```

10. Visualisasikan "gdp" negara "Indonesia" dari tahun 1990 sampai 2010 dalam bentuk line graph dengan judul "Indonesia's GDP Over the Years (1990-2010)"! Beri nama xlab = "Year" dan ylab = "GDP"!

```
king_indo = gapminder %>% filter(country == "Indonesia") %>% filter(1990 <= year & year <= 2010) %>% select(year, gdp)
plot(king_indo$year, king_indo$gdp, type="l", xlab="Year", ylab="GDP", main="Indonesia's GDP Over the Years (1990-2010)" )
```

LATIHAN RESPONSI

```
library(tidyverse)
library(tidymodels)
library(here)
```

2. Import *dataset* **airquality1.csv** dan **airquality2.csv** dengan menggunakan *library* `here`, lalu tampilkan 10 data pertama.

```
airquality1 = read.csv(here("data_raw", "airquality1.csv"))
head(airquality1, 10)
airquality2 = read.csv(here("data_raw", "airquality2.csv"))
head(airquality2, 10)
```

3. Dari soal sebelumnya, dapat dilihat bahwa *dataset* **airquality1** memiliki nilai N/A pada beberapa kolom. Hapus nilai N/A atau lakukan imputasi data sederhana untuk mengisi nilai N/A, lalu tampilkan 10 data pertama.

```
airquality1 = airquality1 %>%
  drop_na()
head(airquality1, 10)
```

4. Perhatikan *dataset* **airquality1** dan **airquality2**, ada satu kolom yang sama dari kedua *dataset* tersebut. Gunakan kolom tersebut untuk menyatukan kedua *dataset* ke dalam variabel baru bernama **airquality**. Tampilkan 6 data terakhirnya.

```
# Gabungkan tabel menggunakan kolom yang sama
airquality = left_join(airquality1, airquality2, by = "X")
tail(airquality, 6)
```

5. Buat kolom baru bernama Date yang menyimpan kombinasi tanggal dari kolom Month dan Day dengan format yyyy-MM-dd (tahun = 1973). Gunakan fungsi `paste` untuk menggabungkan string dan fungsi `as.POSIXct` untuk mengubah string menjadi tanggal. Manfaatkan fungsi *help* sebaik-baiknya.

```
# Buat kolom Date
airquality = airquality %>% mutate(Date = as.POSIXct(paste("1973", Month, Day, sep = "-"), format = "%Y-%m-%d"))
head(airquality)
```

Setelah itu, buang kolom X, Day, dan Month yang tidak akan digunakan untuk membuat model. Kemudian, ubah nama kolom Solar.R menjadi Solar_Radiation. Gunakan operator pipeline.

```
airquality = airquality %>% select(-X, -Day, -Month) %>% rename(Solar_Radiation = Solar.R)
head(airquality)
```

6. Gambarkan perubahan kualitas udara (Ozone) setiap harinya dengan menggunakan ggplot2. Kombinasikan 2 jenis geom yang sesuai dengan data yang ada.

```
library(ggplot2)
# Visualisasi Ozone per hari
ggplot(airquality, aes(x = Date, y = Ozone)) + geom_line() + geom_point() + labs(title = "Perubahan Kualitas Udara (Ozone) Setiap Hari", x = "Tanggal", y = "Ozone")
```

7. Variabel pada dataset ini memiliki range yang berbeda-beda. Lakukan scaling agar berada di range yang sama.

```
# Scaling data to have a mean of 0 and standard deviation of 1 (z-score scaling)
airquality_scaled = airquality %>% mutate(across(where(is.numeric), ~ as.numeric(scale(.))))
# Tampilkan data untuk verifikasi
head(airquality_scaled)
```

8. Bagi dataset untuk *training* dan *testing* dengan proporsi *training* 80%. Pastikan dataset diacak sebelum dibagi, dan pastikan hasil acak akan tetap konsisten walaupun dijalankan berkali-kali dari perangkat berbeda sekalipun.

```
# Data splitting
set.seed(192)
airquality_split <- initial_split(airquality_scaled, prop = 0.8)
airquality_split
```

9. Buat resep untuk *training* data. Tentukan 3 variabel yang menjadi prediktor dan 1 variabel yang menjadi *outcome*. Biarkan variabel Date sebagai ID.

```
airquality_recipe = training(airquality_split) %>% recipe() %>% update_role( Solar_Radiation, Wind, Temp, new_role = "predictor" ) %>% update_role( Ozone, new_role = "outcome" ) %>% update_role( Date, new_role = "ID" ) %>% step_corr( all_predictors() )
airquality_recipe
```

10. Terapkan resep yang sudah dibuat ke data *training* dan *testing*.

```
# airquality_training
airquality_training = airquality_recipe %>% prep() %>% bake( training(airquality_split) )
# airquality_testing
airquality_testing = airquality_recipe %>% prep() %>% bake( testing(airquality_split) )
```

11. *Training* model berdasarkan data yang sudah diolah.

```
airquality_lm = linear_reg(mode = "regression") %>% set_engine("lm") %>% fit( Ozone ~ . -Date, data = airquality_training )
summary(airquality_lm)
```

12. Evaluasi performa model menggunakan data *testing* (performanya jelek juga gapapa)

```
airquality_lm %>% predict(airquality_testing) %>% bind_cols(airquality_testing) %>% metrics( truth = Ozone, estimate = .pred )
```

PERTEMUAN 9

```
# Regresi Logistik dan Shiny App
### Import Library
library(tidyverse)
library(tidymodels)
library(nnet) # Model multinomial logistic regression
library(shiny)# Aplikasi
library(bslib)#
library(datasets)# dataset iris
### Import Data
data(iris)
## Regresi Logistik
### Inisialisasi Kelas
kelas = levels(iris$Species)
kelas
### Data Splitting
set.seed(192)
split = initial_split(iris, prop = 0.8, strata = Species) #strata untuk menyamakan jumlah data tiap kelas di training dan testing

iris_train = split %>% training()
iris_test = split %>% testing()

iris_train %>% select(Species) %>% group_by(Species) %>% summary(freq=n())
iris_test %>% select(Species) %>% group_by(Species) %>% summary(freq=n())
### Modelling
hasil_model = multinom(
  Species ~ .,
  data = iris_train
)
summary(hasil_model)
### Data Testing
#type = "probs", outputnya persentase/kecenderungan sebuah data kepada kelas
#type = "class", outputnya langsung ke kelas apa data itu termasuk
hasil_prediksi_probs = predict(hasil_model, newdata = iris_test, type = "probs")
hasil_prediksi_probs = round(hasil_prediksi_probs*100, digits =2)
hasil_prediksi_probs

hasil_prediksi_class = predict(hasil_model, newdata = iris_test, type = "class")
hasil_prediksi_class
### Evaluasi Model
table(
  predicted_class = hasil_prediksi_class,
  actual_class = iris_test$Species
)
#Akurasi
df = data.frame(
  predicted_class = hasil_prediksi_class,
  actual_class = iris_test$Species
)

nrow(df %>% filter(predicted_class == actual_class))/nrow(df)*100
## Shiny App
### Membuat UI
ui = page_fluid(
  titlePanel("Dataset Iris"),
  selectInput(
    inputId = "species",
    label = "pilih jenis spesies: ",
    choices = kelas
  ),
  tableOutput(
    outputId = "table_iris"
  ),
  tags$br(),
  titlePanel("Uji Coba"),
  layout_columns(
    numericInput(
      inputId = "sl",
      label = "Sepal Length",
      value = 1
    ),
    numericInput(
      inputId = "sw",
      label = "Sepal Width",
      value = 2
    ),
    numericInput(
      inputId = "pl",
      label = "Petal Length",
      value = 3
    ),
    numericInput(
      inputId = "pw",
      label = "Petal Width",
      value = 4
    )
  ),
  actionButton(
    inputId = "klasifikasi",
    label = "Klasifikasi"
  ),
  tags$br(),
  tags$br(),
  textOutput(
    outputId = "hasil_klasifikasi",
  ),
  tags$br()
)
### Membuat logika di belakang layer
server = function(input, output) {
  output$table_iris = renderTable(
    head(
      iris %>%
      filter(
        Species == input$species
      ),
      10
    )
  )

  output$hasil_klasifikasi = renderText({
    input_prediksi = data.frame(
      Sepal.Length = input$sl,
      Sepal.Width = input$sw,
      Petal.Length = input$pl,
      Petal.Width = input$pw
    )


    hasil_class = predict(
      hasil_model,
      newdata = input_prediksi,
      type = "class"
    )

    nama_kelas = kelas[hasil_class]

    hasil_probs = predict(
      hasil_model,
      newdata = input_prediksi,
      type = "probs"
    )

    persentase = round(
      max(hasil_probs)*100,
      digits = 2
    )

    paste(
      nama_kelas,
      "(",
      persentase,
      "%)",
      sep = ""
    )
  })|>bindEvent(input$klasifikasi)
}
### RUN
shinyApp(ui, server)
```

PERTEMUAN 8

```
# Data Modelling
## Data Modelling
### Import Library
library(tidyverse)
library(dslabs)
library(tidymodels)
library(vroom)
library(here)
### Import Data
path = here('data_raw', 'un_smp.csv')
un_smp = vroom(path)
un_smp = un_smp %>% mutate(tahun = as.character(tahun))
str(un_smp)
## Supervised Learning
set seed untuk mengontrol pengacakan data sebelum splitting menjadi data training dan testing
set.seed(42)
un_smp_split = un_smp %>% initial_split(prop = 0.8)
un_smp_split
### Buat Resep
un_smp_recipe = training(un_smp_split) %>%
 recipe() %>%
 update_role(
  tahun,
  status,
  jumlah_peserta,
  bahasa_indonesia,
  bahasa_inggris,
  matematika,
  new_role = "predictor"
 ) %>%
 update_role(
  ipa,
  new_role = "outcome"
 ) %>%
 update_role(
  nama_sekolah,
  new_role = "ID"
 ) %>%
 step_corr(
  all_predictors(),
  -tahun,
  -status
 )
un_smp_recipe
### Terapkan Resep
Terapkan resep ke data training dan data testing
un_smp_training = un_smp_recipe %>%
 prep() %>%
 bake(
  training(un_smp_split)
 )
un_smp_testing = un_smp_recipe %>%
 prep() %>%
 bake(
  testing(un_smp_split)
 )
un_smp_training
un_smp_testing
### Training Model
Training model dengan metode linear regression
un_smp_lm = linear_reg(mode = "regression") %>%
 set_engine("lm") %>%
 fit(
  ipa ~ . -nama_sekolah,
  data = un_smp_training
 )
un_smp_lm
### Prediksi dan Evaluasi
Lakukan prediksi menggunakan data testing, lalu evaluasi model dengan data testing menggunakan fungsi metrics.
un_smp_lm %>%
 predict(un_smp_testing) %>%
 bind_cols(un_smp_testing) %>%
 metrics(
  truth = ipa,
  estimate = .pred
 )
## Unsupervised Learning
### Load Data
data(gapminder)
### Preprocessing Data
Mengganti nilai NA menjadi nilai rata-rata dari kolom tersebut
gapminder$infant_mortality[is.na(gapminder$infant_mortality)] = mean(gapminder$infant_mortality, na.rm = TRUE)
gapminder$life_expectancy[is.na(gapminder$life_expectancy)] = mean(gapminder$life_expectancy, na.rm = TRUE)
gapminder$fertility[is.na(gapminder$fertility)] = mean(gapminder$fertility, na.rm = TRUE)
gapminder$gdp[is.na(gapminder$gdp)] = mean(gapminder$gdp, na.rm = TRUE)

#Ambil data gapminder di tahun 2024
gapminder_2004 = gapminder %>%
 filter(year == 2004) %>%
 select(country, infant_mortality, life_expectancy, fertility, population, gdp)
head(gapminder_2004)
### Scalling data
gapminder_2004_scaled = gapminder_2004 %>%
 select(-country) %>%
 scale()
head(gapminder_2004_scaled)
### Training data
set.seed(123)
kmeans_result = kmeans(
 gapminder_2004_scaled,
 center = 4,
 nstart = 25
)
gapminder_2004$cluster = as.factor(kmeans_result$cluster)
### Elbow method
Untuk menentukan nilai k yang paling optimal
wss = sapply(
 1:10,
 function(k) {
  kmeans(
   gapminder_2004_scaled,
   center = k,
   nstart = 10
  )$tot.withinss
 }
)

elbow_data = data.frame(
 k = 1:10,
 wss = wss
)

ggplot(
 elbow_data,
 aes(
  x = k,
  y = wss
 )
) + geom_line(
 color = "blue"
) + geom_point(
 color = "red",
 size = 3
) + labs(
 title = "Elbow Method for Optimal k",
 x = "Jumlah Klaster (k)",
 y = "Total Within-Cluster Sum of Square (WSS)"
) + theme_minimal()
### Visualisasi data
ggplot(
 gapminder_2004,
 aes(
  x = gdp,
  y = life_expectancy,
  color = cluster
 )
) + geom_point(
 size = 3
) + labs(
 title = "Clustering Gapminder Data (2004)",
 x = "GDP",
 y = "Life Expetancy"
) + theme_minimal() + scale_x_log10()
```

PERTEMUAN 7

```
# Visualisasi Data dengan ggplot2
## Geom Point
### Import Library
library(tidyverse)
library(readxl)
library(dslabs)
### Import Data
data_orang = read.csv("data_raw/data_orang.csv")
data_vg = read_excel("data_raw/data_video_game.xlsx")
## Geom Point
ggplot(
 data = data_vg,
 aes(
  x = NA_Sales,
  y = Global_Sales,
  color = Genre
 )
) + geom_point(
 # color = "#45ea89",
 size = 2
) + labs(
 title = "NA Sales dan Global Sales Video Game",
 x = "NA Sales (Juta)",
 y = "Global Sales (Juta)"
) + theme_linedraw() + facet_wrap(~Genre, nrow = 4, ncol = 3, scales = "fixed")
#untuk scales ada fixed, free, free_x, free_y
## Geom Bar
ggplot(
 data = data_vg,
 aes(
  x = Genre,
  # fill = Genre
 )
) + geom_bar(
 color = "Red",
 fill = "LightBlue"
) + labs(
 title = "Jumlah Video Games Pada Setiap Genre",
 x = "Genre",
 y = "Jumlah"
)
## Geom Line
# geom line mengolah data yang berkaitan dengan waktu
new_data_vg = data_vg %>%
 group_by(Year) %>%
 summarize(Total_Sales = sum(NA_Sales + Global_Sales))

ggplot(
 data = new_data_vg,
 aes(
  x = Year,
  y = Total_Sales
 )
) + geom_line(
 color = "Pink",
 linewidth = 3
) + labs(
 title = "Penjualan Video Games Tahun 1984-2015",
 x = "Tahun",
 y = "Penjualan (Juta)"
)
## Pie Chart
new_data_orang = data_orang %>%
 group_by(Sex) %>%
 summarize(Total = n())

ggplot(
 data = new_data_orang,
 aes(
  x = Total,
  y = "",
  fill = Sex
 )
) + geom_col() + coord_polar() + theme_void()
## Data Wraggling
path =
 system.file("extdata",
             package = "dslabs")
filename =
 file.path(path, "fertility-two-countries-example.csv")
wide_data = read_csv(filename)
View(wide_data)
## Gather
# gathered_data = wide_data %>%
#  gather(year, fertility_rate, `1960`:`2015`)

gathered_data = wide_data %>%
 gather(year, fertility_rate, -country, convert = TRUE)

gathered_data
## Spread
spread_data = gathered_data %>%
 spread(year, fertility_rate)

spread_data
# Join
## Left Join
tab1 = slice(murders, 1:6) %>%
 select(state, population)
tab1
tab2 = results_us_election_2016 %>%
 filter(state %in% c("Alabama", "Alaska", "Arizona", "California", "Connecticut", "Delaware")) %>%
 select(state, electoral_votes) %>%
 rename(ev = electoral_votes)
tab2
## Left & Right Join
left_join = left_join(tab1, tab2, by = "state")
left_join


right_join = right_join(tab1, tab2, by = "state")
right_join
## Inner Join
```{r}
inner_join = inner_join(tab1, tab2, by = "state")
inner_join
```

Full Join
```{r}
full_join = full_join(tab1, tab2, by = "state")
full_join
```
```