Lab 4
Data Science and & Analytics
CSCM603234
Feature Engineering & Feature Selection

Pada tutorial berikut, akan dipelajari mengenai meng-ekstrak dan mengolah fitur, serta melakukan seleksi terhadap fitur. Tipe data yang akan diolah adalah citra dan teks.

Teks

Pada tutorial kali ini, kita akan menggunakan data *Twitter Data (Annotated) for Spam and Sentiment Analysis.* Data terdiri atas 8510 *tweet* dari berbagai domain yang dianotasi secara maual menjadi 4 kategori, yaitu positive, negative, objective-not-spam and objective-spam (*readme*).

1. Package yang harus di-*install* adalah "tm". *Package* ini berguna untuk melakukan *cleaning text* sebelum data tersebut kita proses. Silakan melakukan instalasi package tersebut.
2. Data Cleansing
   a. Membaca data

```
tweet_raw <- read.csv("Manually-Annotated-Tweets.csv",
stringsAsFactors = FALSE)
str(tweet_raw)
```

   b. Membuat co*rpus*

```
tweet_corpus <- VCorpus(VectorSource(tweet_raw$tweet))
print(tweet_corpus)
```

   c. Melihat isi corpus

```
as.character(tweet_corpus[[3]])
```

   d. Mengubah dokumen menjadi lowercase

```
tweet_corpus_clean <-
tm_map(tweet_corpus,content_transformer(tolower))
```

   e. Menghilangkan angka dalam dokumen

```
tweet_corpus_clean <- tm_map(tweet_corpus_clean, removeNumbers)
```

   f. Menghilangkan *stop word*
   Stop word merupakan kata-kata yang tidak penting dalam hal pencarian suatu dokumen. Kata-kata ini berjumlah sangat banyak dalam suatu *corpus*. Dalam bahasa Inggris contohnya *the, at, is*. Kata-kata ini perlu dihilangkan sebelum melakukan *text mining.*

```
tweet_corpus_clean <- tm_map(tweet_corpus_clean, removeWords,
stopwords())
```

   g. Menghilangkan tanda titik
   Tanda baca di dalam dokumen perlu dihilangkan, dalam hal ini contohnya menghilangkan tanda titik.

```
tweet_corpus_clean <- tm_map(tweet_corpus_clean, removePunctuation)
```

h. Melakukan *stemming*

Stemming merupakan proses penghilangan imbuhan sehingga suatu kata berubah menjadi kata dasarnya. Contohnya *learned, learning, learns*, akan menjadi *learn*.

```
install.packages("SnowballC")
library(SnowballC)
tweet_corpus_clean <- tm_map(tweet_corpus_clean, stemDocument)
```

3. Preparasi Data dan Visualisasi

a. Tokenisasi Data

Tokenisasi merupakan proses *spliting* data menjadi token-token. Dalam kasus ini, 1 token merupakan sebuah kata. Pemisahan dokumen menjadi token kata dilakukan berdasarkan karakter spasi. Package tm menyediakan sebuah fungsi yang akan memproses corpus kita menjadi sebuah struktur data yang bernama Document Term Matriks (DTM). Pada DTM, setiap baris merupakan data tweet, sedangkan kolom merepresentasikan kata, dan setiap cell akan diisi dengan frekuensi kata yang ada pada sebuah dokumen tweet. Contohnya adalah sebagai berikut:

tweet#	review	star	citizen	see
1	0	0	1	0
2	1	2	0	0

Untuk membuat DTM, perintah yang digunakan adalah

```
tweet_dtm <- DocumentTermMatrix(tweet_corpus_clean)
```

b. Pembagian data training dan testing

Bagilah sebagian data menjadi untuk training dan sebagian testing.

```
tweet_dtm_train <- ……
tweet_dtm_test <- ……
```

Kemudian, kita akan menyimpan label dari data mentah untuk evaluasi di akhir. Ganti x dan y menjadi angka sesuai dengan jumlah data training dan testing. Angka z adalah total jumlah data.

```
tweet_train_labels <- tweet_raw[x:y, ]$class
tweet_test_labels <- tweet_raw[y+1:z, ]$class
```

c. Visualisasi data

Visualisasi dengan menggunakan Wordcloud membantu kita dalam melihat visualisasi frekuensi kata dalam *corpus*. Kata yang sering muncul akan ditampilkan lebih besar daripada kata-kata yang frekuensinya lebih kecil. Kita dapat menggunakan package wordcloud untuk melakukannya.

```
install.packages("wordcloud")
library(wordcloud)
wordcloud(tweet_corpus_clean, min.freq = 50, random.order = FALSE)
```

4. Pembuatan fitur (*feature generating*)

Dari DTM yang sudah dibuat, ambil kata-kata yang muncul minimal dalam 5 tweet. Tujuannya adalah untuk mengabaikan kata-kata yang jarang muncul di dalam *corpus.* Batas frekuensi dapat diubah-ubah sesuai kebutuhan. Kemudian, buat DTM baru untuk data training dan testing sesuai dengan minimal frekuensi yang telah ditetapkan.

```
tweet_freq_words <- findFreqTerms(tweet_dtm_train, 5)
tweet_dtm_freq_train <- tweet_dtm_train[,tweet_freq_words]
tweet_dtm_freq_test <- tweet_dtm_test[, tweet_freq_words]
```

5. Seleksi fitur

Setelah fitur selesai dibuat, akan dilakukan seleksi fitur dengan menggunakan Chi-squared.

```
install.packages('FSelector')
library(FSelector)
training_freq = data.frame(as.matrix(tweet_dtm_freq_train))
view(training_freq)
training_freq$tweetlabel = tweet_train_labels
weights <- chi.squared(tweetlabel ~., training_freq)
```

Citra

1. Install package EBImage (Image processing and analysis toolbox for R)

```
source("http://bioconductor.org/biocLite.R")
biocLite()
biocLite("EBImage")
library(EBImage
```

2. Eksplorasi citra

```
# set working directory dulu dengan command "setwd(...)" yang satu
lokasi dengan berkas citra

# membaca citra
img <- readImage("daun_1.jpg");

# menampilkan properti citra
print(img);

# melihat dimensi citra (lebar x tinggi)
dim_img <- dim(img);

# menampilkan citra ke GUI (jika tidak dengan method="raster", maka
citra akan muncul melalui internet browser)
display(img,  title="Gambar: Daun 1", method="raster");

# menulis citra ke berkas
writeImage(img, 'daunku.jpeg', quality=85);
```

3. Pembuatan fitur (*feature generating*)

Citra memiliki 3 fitur dasar yaitu fitur warna, bentuk, dan tekstur.

a. Warna

Properti RGB (*Red, Green, Blue*)

```
# menampilkan citra dengan ukuran 200x350 pixel dan semua properti
warna (merah, hijau, biru)
display(img[1:200,1:350,], method="raster");

# menampilkan citra dengan hanya warna merah (nilai parameter ke-3: 1)
display(img[,,1], method="raster");
```

Histogram dari properti citra bisa menjadi nilai fitur. Salah satunya adalah membuat histogram dari properti warna RGB. Histogram terdiri atas sejumlah bin yang memiliki rentang nilai. Setiap bin diisi dengan jumlah entri yang memenuhi rentang nilai tersebut. Contoh:

30	200	10
42	32	67
100	51	25
21	0	30

Properti warna hijau pada citra X (3x4 pixel) adalah berikut:

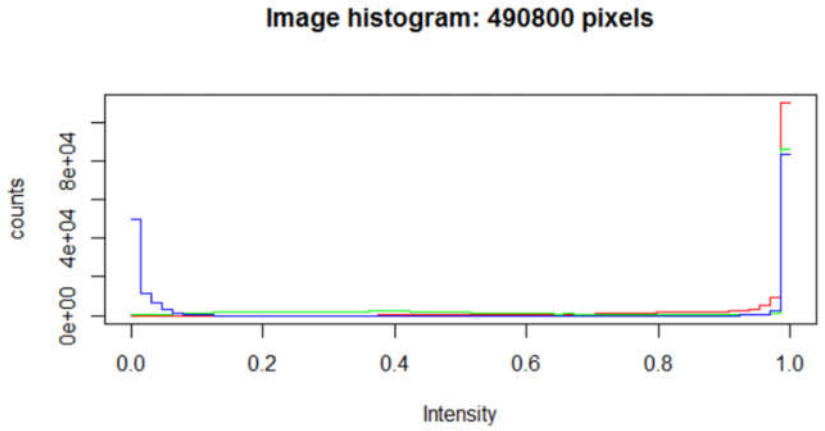
Jika menggunakan 4 bin, maka setiap bin akan memiliki rentang sepanjang 256/4=64. Bin 0-63 terdiri atas 9 entri, yaitu 30, 10, 42, 32, 67, 51, 25, 21, 0, dan 30.

0-63	64-127	128-191	192-255
9	2	0	1

Pada R, ada fungsi untuk membuat histogram dari properti warna, yaitu hist(...). Jumlah bin secara default adalah 64, maka rentang berukuran 256/64=4 entri. Nilai properti warna citra otomatis telah dinormalisasi dari awalnya 0-255 menjadi 0-1, yang terlihat di histogram sebagai *intensity*.

```
img_hist <- hist(img)

# menampilkan struktur data dari img_hist
str(img_hist)
```




```
# mengambil nilai properti warna (merah, hijau, atau biru) dari citra
img_ch_r = channel(img, "asred");
img_ch_g = channel(img, "asgreen");
img_ch_b = channel(img, "asblue");

# menampilkan struktur img_ch_r
str(img_ch_r)

# alternatif lain adalah mengambil nilai dan memasukkan ke data frame
channels = sapply(c("red", "green", "blue"),
  function(ch) as.vector(channel(img, ch)),
  simplify = FALSE)
channels = as.data.frame(channels)
# menampilkan channel red pada data frame
channels$red

# menampilkan struktur data channel red pada data frame
str(channels$red)
```

Setelah nilai properti warna didapat, rata-rata dan standar deviasi dari nilai dapat menjadi fitur citra.

```
img_R_mean = mean(channels$red);
img_G_mean = mean(channels$green);
img_B_mean = mean(channels$blue);
```

Properti HSV (*Hue, Saturation, Value* (*brightness*))

Nilai properti HSV bisa didapat dengan fungsi "rgb2hsv" sehingga dihasilkan 3 data H (*hue*), S (*saturation*), dan V (*value* untuk *brightness*).

```
img_hsv <- rgb2hsv(channels$red*255,channels$green*255,
channels$blue*255, maxColorValue = 255);

# menampilkan struktur data nilai properti HSV
str(img_hsv);

print(img_hsv[1,]); # menampilkan nilai H untuk semua pixel
print(img_hsv[2,]); # menampilkan nilai S untuk semua pixel
print(img_hsv[3,]); # menampilkan nilai V untuk semua pixel
```

Fungsi "rgb2hsv" melakukan normalisasi nilai RGB. Karena itu, nilai RGB dikalikan 255 terlebih dahulu. Perhitungan manual HSV adalah sebagai berikut:

- Jika RGB masih dalam rentang 0-255, lakukan normalisasi menjadi 0-1.
- Hitung nilai maksimum, minimum, dan delta dari keduanya.

$$C_{max} = \max(R, G, B)$$

$$C_{min} = \min(R, G, B)$$

$$\Delta = C_{max} - C_{min}$$

- Kalkulasi nilai *hue*:

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

- Kalkulasi nilai *saturation*:

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

- Nilai V sama dengan C_{max}.

Contoh untuk pixel dengan indeks 235-240:

```
channels$red[235:240];
channels$green[235:240];
channels$blue[235:240];
cmax = pmax(channels$red[235:240], channels$green[235:240],
channels$blue[235:240]);
cmin = pmin(channels$red[235:240], channels$green[235:240],
channels$blue[235:240]);
delta = cmax-cmin;
```

Properti YIQ (Y sebagai *luminance* dan I dan Q sebagai informasi *chrominance*)

Perhitungan nilai properti YIQ dihitung sebagai berikut:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

```
# menghitung nilai Y
img_Y <- 0.299 * channels$red + 0.587 * channels$green + 0.114 *
channels$blue;
# menghitung nilai I
img_I <- 0.596 * channels$red - 0.274 * channels$green - 0.322 *
channels$blue;
```

b. Bentuk

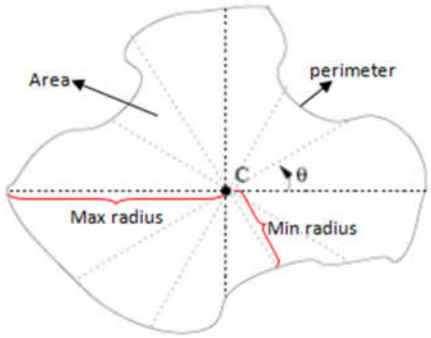
Bentuk merupakan salah satu fitur yang dapat diperoleh dari sebuah citra. Bentuk adalah informasi geometris yang tetap ketika efek lokasi, skala, dan rotasi dilakukan terhadap sebuah objek (D.G. Kendall).

Dalam package EBImage telah tersedia beberapa fungsi untuk membuat fitur, salah satunya adalah fungsi `computeFeatures.shape` untuk membuat fitur bentuk dari sebuah citra. Perlu diingat bahwa, untuk membuat fitur bentuk dari sebuah citra, citra masukan harus berukuran 2D sehingga untuk citra 3D atau citra berwarna harus dikonversi terlebih dahulu menjadi citra *grayscale* atau citra biner.

Lakukan langkah berikut untuk memperoleh fitur bentuk dari citra daun:

```
img_gray = channel(img,"gray")#convert image to grayscale
fts = computeFeatures.shape(img_gray)#generate shape features
fts
s.area s.perimeter s.radius.mean s.radius.sd s.radius.min s.radius.max
1  74158        1270      98.63257    40.86266     12.87985     176.1535
```

Output dari fungsi `computeFeatures.shape` terdiri dari 5 fitur yaitu: area, perimeter, radius mean, radius standar deviasi, radius minimal dan radius maksimal.

***Area** merupakan luas suatu objek yang dinyatakan dalam jumlah piksel yang terdapat pada objek tersebut. ***Perimeter** atau keliling menyatakan panjang tepi suatu objek. ***Radius minimal** merupakan jarak terpendek antara pusat massa dan titik dalam kontur. ***Radius maksimal** merupakan jarak terpanjang antara pusat massa dan titik dalam kontur. ***Radius mean** merupakan jarak rata-rata antara pusat massa dan titik dalam kontur. ***Radius standar deviasi** merupakan standar deviasi dari jarak antara pusat massa dan titik dalam kontur.	

c. Tekstur

Tekstur pada citra merupakan hubungan mutual antara nilai intensitas piksel-piksel yang bertetangga yang berulang di suatu area yang lebih luas daripada jarak hubungan tersebut (Kulkarni, 2004).
Beberapa jenis image texture feature extraction
Pada tutorial ini kita akan menggunakan GLCM sebagai fitur ekstraksi pada citra.

```
install.packages("glcm")
install.packages("raster") # needed for plotRGB function
library(glcm)
library(raster)
r <- raster("daun_1.jpg") # need "rgdal" package

textures<- glcm(r, window = c(3, 3), shift = c(1, 1), statistics =
c("mean", "mean ENVI", "variance", "variance ENVI", "homogeneity",
"contrast", "dissimilarity", "entropy", "second_moment",
"correlation"), na_opt="any", na_val=NA)

names(textures)
plot(textures$glcm mean)
plot(textures$glcm variance)
```

4. Seleksi fitur

Untuk melakukan seleksi fitur, lakukan perhitungan fitur (no.3) untuk citra daun_1.jpg, daun_2.jpg, dan dan daun_3.jpg. Simpan data dalam data frame "df". Daun_1.jpg diberi kelas A, Daun_2.jpg diberi kelas B, dan daun_3.jpg diberi kelas C.
Fitur warna: rata-rata dari nilai RGB, HSV, YIQ
Fitur bentuk: area, perimeter, radius minimal, radius maksimal, radius mean, radius standar deviasi.
Fitur tekstur: mean, variance, homogeneity, contrast, dissimiliarity, entropy, second_moment, correlation

Kelas	R_mean	G_mean	...	area	perimeter	glcm_mean	glcm_variance	...
A								
B								
C								

Metode univariat

Chi-square

```
library(FSelector)
weights <- chi.squared(Kelas~., df)

# Menampilkan bobot
print(weights)
```

T-Test

```
label <- df$Kelas
ttest <- lapply(df[,-1], function(x) { t.test(x ~ label)$statistic })
ttest
```