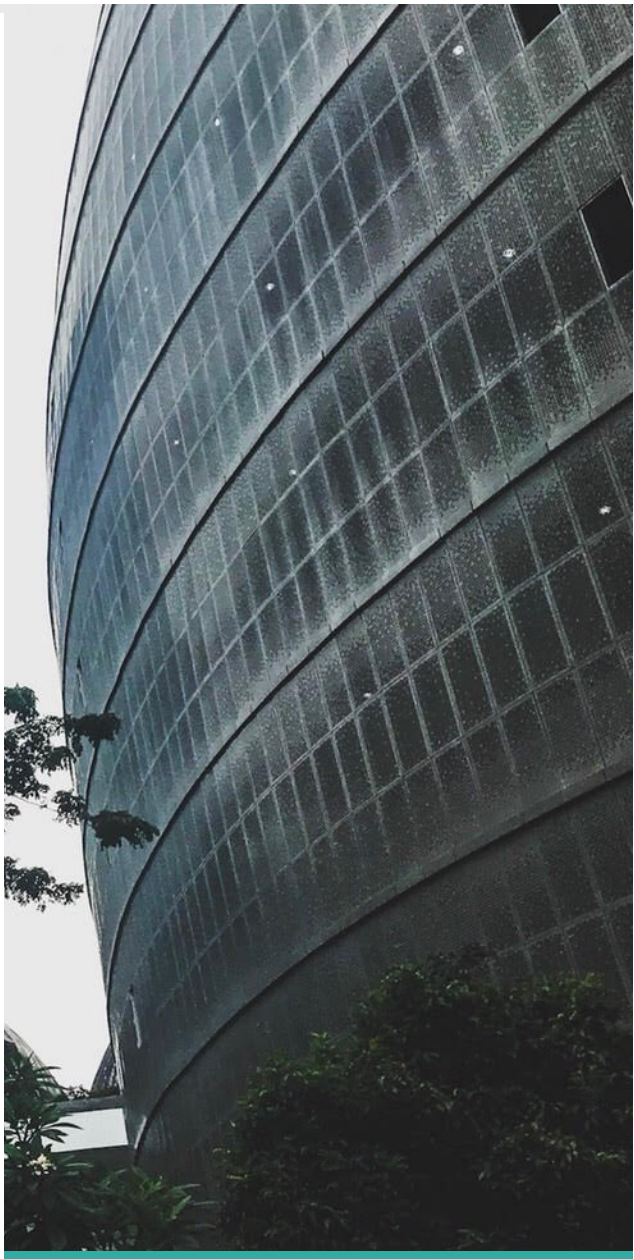
MODUL PRAKTIKUM

IS411 – DATA MODELING
PROGRAM SARJANA S1 SISTEM INFORMASI
FAKULTAS TEKNIK DAN INFORMATIKA



PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNIK DAN INFORMATIKA
UNIVERSITAS MULTIMEDIA NUSANTARA
Gedung B Lantai 5, Kampus UMN
Jl. Scientia Boulevard, Gading Serpong, Tangerang, Banten-15811 Indonesia
Telp: +62-21.5422.0808 (ext. 1803), email: ict.lab@umn.ac.id, web: umn.ac.id

MODUL 8
MACHINE LEARNING

DESKRIPSI TEMA

1. Training and Testing Set on Machine Learning
2. Further Knowledge to Classification Algorithms.

CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Students are able to implement machine learning techniques or methods (C_3)

PENUNJANG PRAKTIKUM

1. Anaconda Navigator
2. Jupyter Notebook

(+ Perlengkapan Apd/Alat Pelindung Diri Yang Harus Digunakan, Jika Ada)

LANGKAH-LANGKAH PRAKTIKUM

Import Library

1. Lakukan langkah yang sama seperti minggu sebelumnya untuk meng-import library Numpy dan Pandas.
2. Import dataset menggunakan file “winequality-red.csv”.

```
dataset = pd.read_csv('winequality-red.csv', delimiter=';')
dataset
```

3. Lakukan pembagian data training dan testing.

```
from sklearn.model_selection import train_test_split

#membagi data ke X dan Y
redwine = dataset.copy()
Y = redwine['quality']
X = redwine.drop(columns = 'quality')

#membagi data training 80% dan testing 20%
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

Cross Validation

Pada praktikum sebelumnya telah dilakukan klasifikasi menggunakan algoritma LogisticRegression terhadap data wine, dimana setelah melakukan testing dan training tidak ada validasi terhadap hasil klasifikasi.

Sebagai perbandingan terhadap hasil dari klasifikasi, berikut adalah contoh klasifikasi LogisticRegression biasa.

Berikut ini adalah contoh data redwine dengan label yang sudah diberikan.

4. Pemodelan dengan algoritma LogisticRegression.

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'.format(logreg.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

5. Selanjutnya adalah melakukan Cross Validation menggunakan library **cross_val_score**, sistem akan melakukan pengulangan pada data train yang akan menghasilkan akurasi yang berbeda.

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(logreg, X_train, y_train, cv=10)
```

```
print('Cross-Validation Accuracy Scores', scores)
scores = pd.Series(scores)
scores.min(), scores.mean(), scores.max()
```

6. Catatlah berapa hasil Cross-Validation score yang diperoleh.

Natural Language Processing (NLP)

Natural language processing umumnya digunakan pada kumpulan teks, seperti yang dihasilkan / didapatkan dari Tweet, *post* media sosial, percakapan, *review* film, berita, dan sebagainya. Kumpulan teks disebut dengan ***corpus***, atau dalam bentuk jamak disebut ***corpora***.

Ada beberapa cara dan *tools* untuk melakukan NLP, pada praktikum kali ini kita akan menggunakan TextBlob. **TextBlob** adalah *object-oriented NLP text-processing library* yang dibangun pada NLTK dan *pattern NLP libraries*.

7. Membuat TextBlob.

```
#install TextBlob

#Cara 1: menggunakan Anaconda Prompt
#conda install -c conda-forge textblob

#Cara 2: menggunakan Jupyter Notebook
#!pip install textblob

!pip install textblob
```

```
#download the NLTK cropora used by TextBlob

import nltk
nltk.download('averaged_perceptron_tagger')
```

```
import nltk
nltk.download('brown')
```

```
#Create a TextBlob

from textblob import TextBlob

text = 'Today is a beautiful day. Tomorrow looks like bad weather.'

blob = TextBlob(text)

blob
```

8. Melakukan **Tokenisasi** teks ke dalam kalimat (*sentences*) dan kata (*words*).

```
#sentence property to get a list of Sentence objects

blob.sentences
```

```
#words property returns a WordList object containing a list of Word objects

blob.words
```

9. Memecah teks menjadi ***Part-of-Speech Tagging***

Part-of-Speech Tagging memecah teks / kalimat ke dalam *tuples*, biasanya per suku kata.

```
#tags property returns a list of tuples, each containing a word and a string representing its part-of-speect tag

blob.tags
```

10. Kita juga dapat mengekstrak kalimat atau teks ke dalam frasa.

```
#noun_phrases property returns a WordList object containing a list of Word objects - one for each noun phrase in the text

blob.noun_phrases
```

11. Melakukan analisis sentimen (*sentiment analysis*) menggunakan **Sentiment Analyzer** dari TextBlob

```
#getting the sentiment of a TextBlob

blob.sentiment
```

```
#getting the polarity and subjectivity from the Sentiment Object

%precision 3
```

```
blob.sentiment.polarity
```

```
blob.sentiment.subjectivity
```

```
#getting the sentiment of a sentence
#get the sentiment at the individual sentence level

for sentence in blob.sentences:
    print(sentence.sentiment)

```

12. Melakukan analisis sentiment menggunakan algoritma Naïve Bayes, dengan **NaiveBayesAnalyzer**

```
from textblob.sentiments import NaiveBayesAnalyzer

blob = TextBlob(text, analyzer=NaiveBayesAnalyzer())

blob
```

Kita bisa mencoba menggunakan dataset lain, misalnya dataset ulasan film berikut.

```
import nltk
nltk.download('movie_reviews')
```

```
blob.sentiment
```

```
#get the sentiment of each sentences

for sentence in blob.sentences:
    print(sentence.sentiment)
```

13. *Inflection: Pluralization* dan *Singluarization*

Inflection adalah bentuk berbeda dari kata yang sama, biasanya dalam bahasa Inggris seringkali terdapat dua kata yang berbeda karena terdapat bentuk tunggal (*singular*) dan jamak (*plural*), meskipun memiliki arti yang sama. Misalnya "person" dan "people" sama – sama memiliki arti "orang", atau "run" dan "ran" sama – sama memiliki arti "lari".

```
#Words and WordLists each support converting words to their singluar or plurar forms.

from textblob import Word

index = Word('index')
index.pluralize()
```

```
cacti = Word('cacti')
cacti.singularize()
```

```
from textblob import TextBlob

animals = TextBlob('dog cat fish bird').words
animals.pluralize()
```

14. *Spell Checking and Correction*

Kita dapat melakukan pengecekan atau koreksi terhadap ejaan yang keliru.

```
#spellcheck method returns a list of tuples containing possible correct spellings and a confidence value

from textblob import Word

word = Word('theyr')
word.spellcheck()
```

```
#correct method chooses word woth the highest confidence value

word.correct()
```

```
from textblob import Word

sentence = TextBlob('Ths sentense has misspelled wrds.')
sentence.correct()
```

15. *Normalization: Stemming and Lemmatization*

Stemming dilakukan untuk menghilangkan awalan (*prefix*) atau akhiran (*suffix*) dari suatu kata sehingga hanya tersisa sebagai kata dasar. Hal ini perlu dilakukan untuk mengurangi risiko adanya kata yang redudansi.

```
#Stemming removes a prefix or suffix from a word leaving only a stem, which may or may not a real word.
#Lemmatization is similar, but factors in the word's part of speech and meaning and results in a real word.

from textblob import Word

word = Word('strawberries')
word.lemmatize()
```

```
word.stem()
```

16. Menghitung Frekuensi Kata

Untuk melakukan perhitungan frekuensi kata, kita menggunakan dataset **RomeoAndJuliet.txt**.

```
#Various techniques for detecting similarity between documents rely on word frequencies.

#use Path class form the Python Standard Library's pathlib module

from pathlib import Path
from textblob import TextBlob

#load the e-book RomeoAndJuliet.txt (untuk memudahkan, letakan file di path yang sama dengan tempat menyimpan ipynb ini)
blob = TextBlob(Path('RomeoAndJuliet.txt').read_text())
```

```
blob.word_counts['juliet']
```

```
blob.word_counts['romeo']
```

Pertanyaan: Berapa jumlah kata "juliet" dan "romeo" yang Anda dapatkan?

17. Mendapatkan Arti Kata, Sinonim, dan Antonim menggunakan **WordNet**

    WordNet adalah database yang dibuat oleh Princeton University. TextBlob library menggunakan antarmuka library NLTK yaitu WordNet, untuk mendapatkan arti kata, sinonim, dan antonim dari suatu kata.

```
#getting definitions

from textblob import Word

happy = Word('happy')

#definitions property returns a list of all the word's definitions in the WordNet database

happy.definitions
```

```
#getting synonims

happy.synsets
```

18. Menghapus *Stop Word*

    Stop word adalah kata umum pada teks yang biasanya dihilangkan sebelum melakukan analisis, karena dianggap tidak memberikan arti/makna penting untuk analisis.

```
#download NLTK's stop-words lists

import nltk

nltk.download('stopwords')
```

```
#load the english stop words list

from nltk.corpus import stopwords

stops = stopwords.words('english')
```

```
#create a TextBlob from wich we'll remove stop words

from textblob import TextBlob

blob = TextBlob('Today is a beautiful day.')

[word for word in blob.words if word not in stops]
```

19. N-Grams

N-Grams adalah urutan n item teks, seperti huruf dalam kata atau kata dalam kalimat. Dalam *Natural Language Processing*, n-gram dapat digunakan untuk mengidentifikasi huruf atau kata yang sering muncul berdekatan satu sama lain. Untuk input pengguna berbasis teks, ini dapat membantu memprediksi huruf atau kata berikutnya yang akan diketik pengguna.

```
from textblob import TextBlob

text = 'Today is a beautiful dat. Tomorrow looks like bad weather.'

blob = TextBlob(text)
blob.ngrams()
```

```
#n-grams consisting of 5 words

blob.ngrams(n=5)
```

20. Visualisasi Frekuensi Kata dengan Bar Charts dan Word Clouds

    Untuk membuat visualisasi jumlah/frekuensi suatu kata dari dataset, kitab isa menggunakan Pandas.

    a. Menggunakan Bar Charts

```
#load the data

from pathlib import Path
from textblob import TextBlob

blob = TextBlob(Path('RomeoAndJuliet.txt').read_text())
```

```
#load the NLTK stopwords

from nltk.corpus import stopwords

stop_words = stopwords.words('english')
```

```
#getting the word frequencies

items = blob.word_counts.items()
```

```
#eliminating the stop words

items = [item for item in items if item[0] not in stop_words]
#item[0] gets the word from each tuple so we can check the whether it's in stop_words
```

```
#sorting the words by requency

from operator import itemgetter

sorted_items = sorted(items, key=itemgetter(1), reverse=True)

#As sorted orders item's elements,
#it accesses the element at index 1 in each tuple via the expression itemgetter(1)
#The reverse=True keyword argument indicates that the tuples should be sorted in descending order
```

```
#getting the top 20 words

top20 = sorted_items[1:21]
```

```
#convert top20 to a DataFrame

import pandas as pd

df = pd.DataFrame(top20, columns=['word', 'count'])
df
```

```
#visualizing the DataFrame

axes = df.plot.bar(x='word', y='count', legend=False)
```

b. Menggunakan Word Clouds

```
#installing the wordcloud module

#Cara 1: menggunakan Anaconda Prompt
#conda install -c conda-forge wordcloud

#Cara 2: via Jupyter Notebook
#!pip install wordcloud

!pip install wordcloud
```

```
#loading the text

from pathlib import Path

text = Path('RomeoAndJuliet.txt').read_text()
```

```
#configuring the WordCloud object

from wordcloud import WordCloud

wordcloud = WordCloud(colormap='prism', background_color='white')
```

```
#generating the Word Cloud

wordcloud = wordcloud.generate(text)
```

```
#saving the Word Cloud as an image file

wordcloud = wordcloud.to_file('RomeoAndJulietWordCloud.png')
```

Challenge:

1. Ambil kata-kata dari satu dokumen sampel (Shakespeare-hamlet.txt) di corpus Gutenberg dan tampilkan.
2. Tampilkan 500 kata pertama dari teks corpus tersebut dalam sebuah paragraf.
3. Lakukan proses Tokenization,Stemming,Lemmatization dan POS Tags pada paragraf tersebut.

PENGUMPULAN

1. File yang dikumpulkan terdiri dari:
    a. File project (.ipynb)
    b. File PDF berisi screenshot output dan jawaban (jika ada pertanyaan)
2. File di-compress (.zip) dan diberi nama **KODEMATAKULIAH_KELAS_NIM_NAMA_WEEK KE-XX.zip** (contoh: IS5411_A_13110310017_Monika Evelin Johan_Week-01.zip).

REFERENSI

Deitel, P., & Deitel, H. (2020). Intro to Python for Computer Science and Data Science. Pearson Education.