

SCHEDULING DAN PIPELINING DATA STREAM PADA KASUS STREAMING DATA TWITTER DENGAN TOPIK WINDOWS 11

Aldi Fianda Putra^{*1}, Riski Darmawan², Hasyir Daffa Ibrahim³

^{1,2,3} Universitas Brawijaya

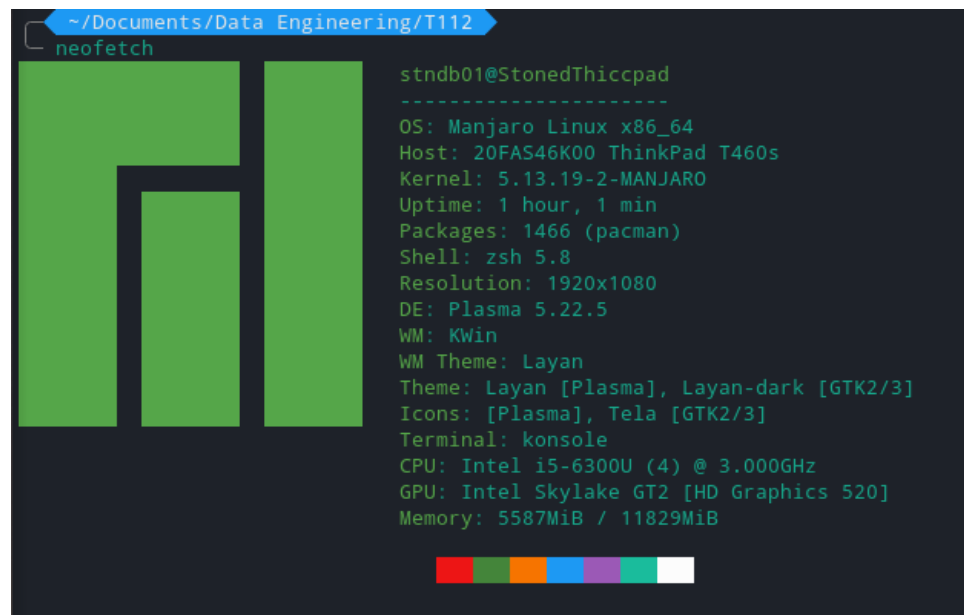
Email: ¹aldifp8492@student.ub.ac.id, ²riskidarmawan@student.ub.ac.id, ³melan_gnrx@student.ub.ac.id

^{*}Penulis Korespondensi

1. INFORMASI PROYEK

Pada proyek akhir Data Engineering ini, kami melakukan proses mekanisme *streaming* data, ETL dan *machine learning* beserta *dashboard* dengan menggunakan data stream dari Twitter. Topik yang dipilih sendiri adalah “Windows 11”. Topik ini kami pilih karena seperti yang kita ketahui Windows 11 baru dirilis pada tahun ini, sehingga pasti muncul pro dan kontra terkait rilisnya sistem operasi baru tersebut pada sosial media terutama pada *platform* Twitter, sehingga *Machine Learning* sangat memungkinkan untuk diimplementasikan terhadap data tersebut, terutama terkait analisis sentimen terhadap rilisnya Windows 11 beserta dengan *dashboard*.

2. INFORMASI ENVIRONMENT IMPLEMENTASI



Gambar 1. Environment Implementasi

Pada proyek akhir Data Engineering kami menggunakan laptop Aldi sebagai *environment* untuk melakukan implementasi proyek akhir. Laptop tersebut memiliki spesifikasi seperti gambar diatas, dimana laptop yang digunakan adalah Lenovo Thinkpad T460s dengan CPU intel i5-6300U 3.00Ghz dengan memori sebesar 12 GB. Untuk lingkungan sistem operasi yang digunakan adalah Manjaro Linux 21.0 “Ornara” dengan *desktop environment* KDE Plasma 5.22.5. Terdapat beberapa perangkat lunak yang digunakan untuk proyek akhir ini, untuk powershell yang digunakan adalah zsh. Sedangkan untuk airflow yang digunakan adalah airflow versi 2.2.2. Kemudian untuk Kafka menggunakan versi 3.0.0 dengan Scala versi 2.12. Terdapat juga aplikasi lain yang digunakan untuk membantu mengerjakan proyek akhir ini seperti sublime text, kate, dan juga *online notebook* yakni Deepnote.

3. INSTALASI DAN KONFIGURASI KAFKA

Berikut ini adalah prosedur instalasi kafka sebelum masuk ke dalam prosedur instalasi airflow :

- Pastikan Java terinstall pada perangkat, Kafka hanya dapat berjalan pada java versi 8 atau 1.8.0, Java 8 dapat diunduh melalui link berikut <https://www.oracle.com/java/technologies/downloads/>.

```
Command Prompt
Microsoft Windows [Version 10.0.19044.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kidar>java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)

C:\Users\kidar>
```

Gambar 2. Versi Java pada Windows

```
~/Documents/Data Engineering/T112
java -version
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

Gambar 3. Versi Java pada Linux

- b. Unduh kafka melalui link <https://kafka.apache.org/downloads>, pilih file tar yang merupakan file binary dan disesuaikan dengan skala yang terinstall pada perangkat.

2.8.0

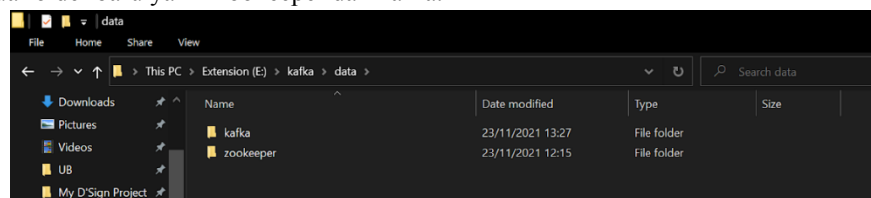
- Released April 19, 2021
- [Release Notes](#)
- Source download: [kafka-2.8.0-src.tgz](#) ([asc](#), [sha512](#))
- Binary downloads:
 - Scala 2.12 - [kafka_2.12-2.8.0.tgz](#) ([asc](#), [sha512](#))
 - Scala 2.13 - [kafka_2.13-2.8.0.tgz](#) ([asc](#), [sha512](#))

We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.13 is recommended).

Kafka 2.8.0 includes a number of significant new features. Here is a summary of some notable changes:

Gambar 4. Halaman untuk mengunduh Kafka

- c. Ekstrak file kafka yang telah terunduh sebelumnya, kemudian buat folder data yang didalamnya terdapat dua folder baru yakni zookeeper dan kafka.



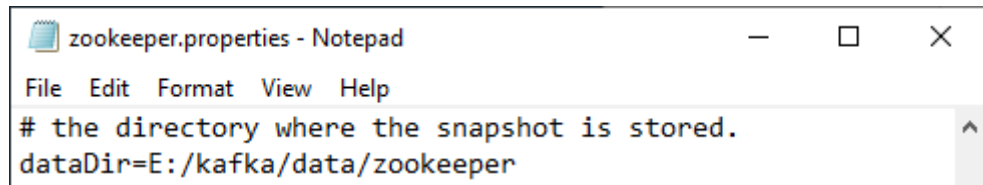
Gambar 5. Tampilan folder Kafka

- d. Setelah diekstrak, kemudian masuk ke dalam folder config dan ubah direktori tempat penyimpanan data pada zookeeper.properties seperti.

```
dataDir = E:/kafka/data/zookeeper
```

atau jika menggunakan linux direktorinya menjadi

```
dataDir=/home/USER/kafka/data/zookeeper
```



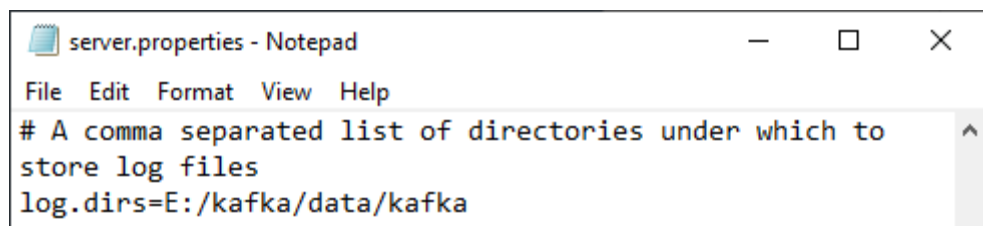
Gambar 6. Perubahan pada file zookeeper.properties

- e. Selain zookeeper konfigurasi server.properties juga diganti menjadi

```
log.dirs = E:/kafka/data/kafka
```

atau jika menggunakan linux direktorinya menjadi

```
log.dirs=/home/USER/kafka/data/kafka
```



Gambar 7. Perubahan pada file server.properties

- f. Pada server.properties, hilangkan comment pada listener dan tambahkan 127.0.0.1 pada IP PLAINTEXT agar broker kafka berjalan

```
# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
# FORMAT:
# listeners = listener_name://host_name:port
# EXAMPLE:
# listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://127.0.0.1:9092
```

Gambar 8. Baris yang perlu ditambahkan pada Linux

- g. Apabila menggunakan linux, tambahkan path kafka ke dalam file konfigurasi powershell yang digunakan seperti .bashrc ataupun .zshrc, tambahkan path tersebut pada baris paling akhir seperti

```
"export PATH=/home/stndb01/kafka_2.13-3.0.0/bin:$PATH"
```

```
# Set personal aliases, overriding those provided by oh-my-zsh libs,
# plugins, and themes. Aliases can be placed here, though oh-my-zsh
# users are encouraged to define aliases within the ZSH_CUSTOM folder.
# For a full list of active aliases, run `alias`.
#
# Example aliases
# alias zshconfig="mate ~/.zshrc"
# alias ohmyzsh="mate ~/.oh-my-zsh"
source ~/powerlevel10k/powerlevel10k.zsh-theme

# To customize prompt, run `p10k configure` or edit ~/.p10k.zsh.
[[ ! -f ~/.p10k.zsh ]] || source ~/.p10k.zsh

export PATH=/home/stndb01/kafka_2.13-3.0.0/bin:$PATH
```

Gambar 9. Konfigurasi Shell yang digunakan

- h. Jalankan zookeeper dengan perintah :

```
zookeeper-server-start.bat ../../config/zookeeper.properties
```

- i. Jalankan kafka dengan perintah:

```
kafka-server-start.bat ../../config/server.properties
```

4. INSTALASI DAN KONFIGURASI AIRFLOW

Berikut ini adalah prosedur instalasi *airflow* sebelum masuk ke dalam pembahasan scheduling data:

- Pastikan menggunakan *environment* linux seperti Ubuntu, Arch dan lain-lain, hal ini dikarenakan proses instalasi dan konfigurasi Airflow jauh lebih mudah dilakukan jika menggunakan environment Linux. Pada percobaan ini kami menggunakan linux dengan distro Manjaro yang merupakan turunan Arch.
- Install python-pip terlebih dahulu dengan menggunakan perintah `sudo apt-get install python-pip`, sedangkan pada perangkat kami karena menggunakan Arch-based maka menggunakan perintah `sudo pacman -S python-pip` ataupun `yay -S python-pip`.

```
~/Documents/Data_Engineering/T122
$ sudo pacman -S python-pip
warning: python-pip-20.3.4-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) python-pip-20.3.4-1

Total Installed Size: 1.60 MiB
Net Upgrade Size: 0.00 MiB

:: Proceed with installation? [Y/n] y
(1/1) checking keys in keyring
(1/1) checking package integrity
(1/1) loading package files
(1/1) checking for file conflicts
(1/1) checking available disk space
:: Running pre-transaction hooks...
(1/1) Creating Timeshift snapshot before upgrade...
=> skipping timeshift-autosnap due skipRsyncAutosnap in /etc/timeshift-autosnap.conf set to TRUE.
:: Processing package changes...
(1/1) reinstalling python-pip
:: Running post-transaction hooks...
(1/2) Arming ConditionNeedsUpdate...
(2/2) Refreshing PackageKit...

[#####] 100%
[#####] 100%
[#####] 100%
[#####] 100%
[#####] 100%
[#####] 100%
```

Gambar 10. Proses instalasi python-pip

- Install dependensi yang dibutuhkan airflow seperti `libmysqlclient-dev`, `libssl-dev`, `libkrb5-dev` ataupun dependensi lainnya mengikuti error yang akan muncul nantinya. Kemudian Install airflow dengan menggunakan perintah `pip install apache-airflow`, jangan lupa menambahkan PATH airflow pada file konfigurasi shell yang digunakan.

```
~/Documents/Data_Engineering/T122
$ pip install apache-airflow
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: apache-airflow in /home/stndb01/.local/lib/python3.9/site-packages (2.2.2)
Requirement already satisfied: sqlalchemy>=1.3.18 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.3.24)
Requirement already satisfied: iso8601>=0.1.12 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.0.2)
Requirement already satisfied: marshmallow-oneofschema>=2.0.1 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (3.0.1)
Requirement already satisfied: wtforms<3.0.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (2.3.3)
Requirement already satisfied: flask<2.0,>=1.1.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.1.4)
Requirement already satisfied: markupsafe<2.0,>=1.1.1 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.1.1)
Requirement already satisfied: flask-wtf<0.15,>=0.14.3 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (0.14.3)
Requirement already satisfied: lockfile<0.12.2 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (0.12.2)
Requirement already satisfied: python-slugify<5.0,>=3.0.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (4.0.1)
Requirement already satisfied: pyyaml<5.1 in /usr/lib/python3.9/site-packages (from apache-airflow) (5.4.1)
Requirement already satisfied: sqlalchemy-jsonfield<=1.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.0.0)
Requirement already satisfied: werkzeug<=1.0.1,~>1.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.0.1)
Requirement already satisfied: itsdangerous<2.0,>=1.1.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.1.0)
Requirement already satisfied: apache-airflow-providers-http in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (2.0.1)
Requirement already satisfied: cattr<1.7.0,~>1.1 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.5.0)
Requirement already satisfied: argcomplete<=1.10 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (1.12.3)
Requirement already satisfied: inflection<=0.3.1 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (0.5.1)
Requirement already satisfied: jsonschema<=3.0 in /home/stndb01/.local/lib/python3.9/site-packages (from apache-airflow) (3.2.0)
```

Gambar 11. Instalasi Airflow

```
# To customize prompt, run `p10k configure` or edit ~/.p10k.zsh.
[[ ! -f ~/.p10k.zsh ]] || source ~/.p10k.zsh

export PATH=/home/stndb01/kafka_2.13-3.0.0/bin:$PATH
export AIRFLOW_HOME=~/.airflow
```

Gambar 12. Konfigurasi Airflow pada Shell yang digunakan

- d. Buat database baru dengan menggunakan perintah `airflow db init` jika baru pertama kali menjalankan Airflow.

```
~/Documents/Data_Engineering/T122
airflow db init
DB: sqlite:///home/stndb01/airflow/airflow.db
[2021-11-28 20:34:17,979] {db.py:910} INFO - Creating tables
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
WARNI [airflow.models.crypto] empty cryptography key - values will not be stored encrypted.
WARNI [unusual_prefix_c6a75ab906088349581314b33bcc79fb87b48191_example_kubernetes_executor_config] Could not import DAGs in example_kubernetes_executor_config.py: No module named 'kubernetes'
WARNI [unusual_prefix_c6a75ab906088349581314b33bcc79fb87b48191_example_kubernetes_executor_config] Install kubernetes dependencies with: pip install apache-airflow[cncf.kubernetes]
Initialization done
```

Gambar 13. Membuat database baru untuk Airflow

- e. Jalankan perintah `airflow webserver -p 8080`, kemudian jalankan juga `airflow scheduler` namun pada terminal yang berbeda.

```
~/Documents/Data_Engineering/T122
airflow webserver -p 8080
[2021-11-28 20:38:13,354] {dagbag.py:500} INFO - Filling up the DagBag from /dev/null
[2021-11-28 20:38:13,641] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG Runs.can_create Admin
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: - -
Access Logformat:
[2021-11-28 20:38:41+0700] [7315] [INFO] Starting gunicorn 20.1.0
[2021-11-28 20:38:41+0700] [7315] [INFO] Listening at: http://0.0.0.0:8080 (7315)
[2021-11-28 20:38:41+0700] [7315] [INFO] Using worker: sync
[2021-11-28 20:38:41+0700] [7316] [INFO] Booting worker with pid: 7316
[2021-11-28 20:38:41+0700] [7320] [INFO] Booting worker with pid: 7320
[2021-11-28 20:38:41+0700] [7321] [INFO] Booting worker with pid: 7321
[2021-11-28 20:38:41+0700] [7325] [INFO] Booting worker with pid: 7325
[2021-11-28 20:38:42,974] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG Runs.can_create Admin
[2021-11-28 20:38:43,288] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG Runs.can_create Admin
[2021-11-28 20:38:43,300] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG Runs.can_create Admin
[2021-11-28 20:38:43,431] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG Runs.can_create Admin
```

Gambar 14. Penjalanan Airflow Webserver

```
~/Documents/Data_Engineering/T122
airflow scheduler
[2021-11-28 20:38:48,366] {scheduler_job.py:596} INFO - Starting the scheduler
[2021-11-28 20:38:48,366] {scheduler_job.py:601} INFO - Processing each file at most -1 times
[2021-11-28 20:38:48+0700] [7343] [INFO] Starting gunicorn 20.1.0
[2021-11-28 20:38:48,370] {manager.py:163} INFO - Launched DagFileProcessorManager with pid: 7344
[2021-11-28 20:38:48+0700] [7343] [INFO] Listening at: http://0.0.0.0:8793 (7343)
[2021-11-28 20:38:48+0700] [7343] [INFO] Using worker: sync
[2021-11-28 20:38:48,372] {scheduler_job.py:1114} INFO - Resetting orphaned tasks for active dag runs
[2021-11-28 20:38:48+0700] [7345] [INFO] Booting worker with pid: 7345
[2021-11-28 20:38:48,376] {settings.py:52} INFO - Configured default timezone Timezone('UTC')
[2021-11-28 20:38:48,380] {scheduler_job.py:1137} INFO - Marked 1 SchedulerJob instances as failed
[2021-11-28 20:38:48,387] {manager.py:431} WARNING - Because we cannot use more than 1 thread (parsing_processes = 2) when using sqlite. So we set parallelism to 1.
[2021-11-28 20:38:48+0700] [7346] [INFO] Booting worker with pid: 7346
```

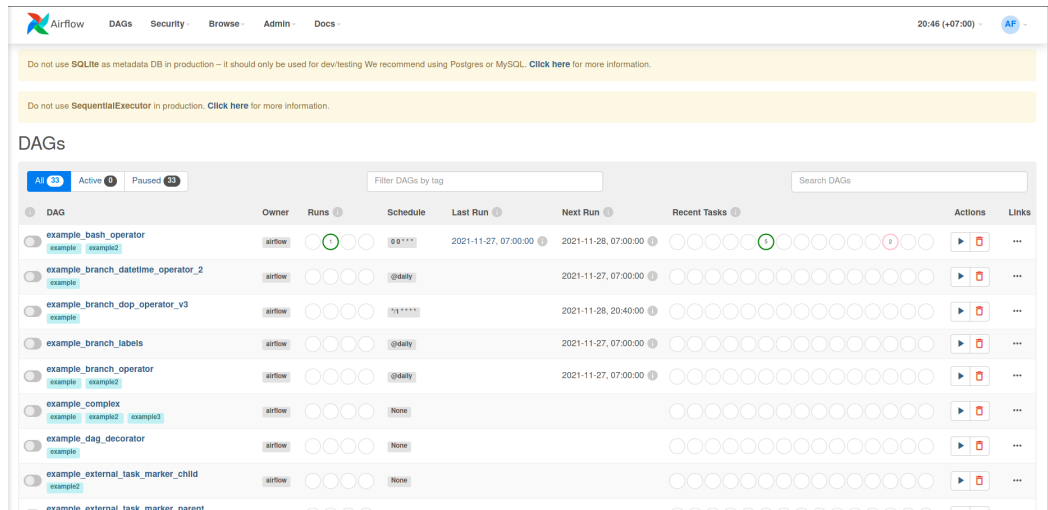
Gambar 15. Penjalanan Airflow Scheduler

- f. Buat sebuah akun untuk airflow dengan urutan `airflow users create [-h] -e EMAIL -f FIRSTNAME -l LASTNAME [-p PASSWORD] -r ROLE [--use-random-password] -u USERNAME`.

```
~/Documents/Data_Engineering/T122
airflow users create -e haisayaaldifp@gmail.com -f Aldi -l Fianda -p admin123 -r Admin -u admin
```

Gambar 16. Pembuatan akun Airflow

- g. Buka web browser pada alamat `localhost:8080`, kemudian login dengan menggunakan akun yang sudah dibuat sebelumnya, dan apabila diperoleh tampilan UI Airflow seperti gambar dibawah maka Airflow siap digunakan.



Gambar 17. Tampilan halaman Airflow ketika login berhasil

5. STREAMING DATA

- a. Buatlah topik menggunakan perintah:

```
kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1
--partitions 1 --topic windows_11
```

Sebagai alternatif, topik dapat dibuat langsung pada modul python untuk proses streaming.

- b. Buat sebuah modul python yang digunakan untuk ingesting data dengan menggunakan library python. Pastikan dalam modul tersebut terdapat access token dan secret, consumer key dan secret, topic, keyword, dan file csv untuk dump. Secara lengkap berikut adalah kode program untuk proses streaming data.

```
# import library
import tweepy
from kafka import KafkaProducer
from datetime import datetime, timedelta
import csv
import pandas as pd

consumer_key = "Gg0FYnVBila9MeVR4cd7iBIHH"
consumer_secret = "i3whWfXsPi5uNjAXwskSQKmZobfgezo4SmK4R8hfL6WkWZGfdM"
access_token = "818938632782299136-VM0HUdQzBHdWsaWkXIfStkzSoKzRa0L"
access_token_secret = "IcKhsxWvCnLVqp2iigRw4tdNVqMsxFNWkGt5kXfQKhcfH"

# setup autentikasi
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

# instansiasi API
api = tweepy.API(auth, wait_on_rate_limit=True)

# menyesuaikan waktu dengan waktu lokal (GMT +7 / UTC +7)
def normalize_time(time):
    mytime = datetime.strptime(time, "%Y-%m-%d %H:%M:%S")
    mytime += timedelta(hours=7)
    return (mytime.strftime("%Y-%m-%d %H:%M:%S"))

# instansiasi Kafka Producer
producer = KafkaProducer(bootstrap_servers=['localhost:9092'], api_version=(0, 10, 1))

# inisialisasi topik, kata kunci, serta batas maksimum query dan tweet
topic_name = 'windows_11'
search_key = "windows 11"
maxId = -1
maxTweets = 3000
tweetCount = 0
tweetsPerQuery = 500
```

```

# deklarasi file csv
csvFile = open(search_key+".csv", "a+", newline="", encoding="utf-8")
csvWriter = csv.writer(csvFile)

# deklarasi list
tweet_id = []
tweet_username = []
tweet_text = []

# perulangan untuk mendapatkan tweet dengna API Twitter hingga limit yang
ditentukan
while tweetCount < maxTweets:
    # mengambil data tweet pertama kali
    if maxId <= 0:
        # newTweets = api.search_tweets(q=search_key, lang="en", count=tweetCount,
max_id=maxId)
        newTweets = api.search_tweets(q=search_key, lang="en", count=tweetCount)
    # mengambil data tweet kedua dan seterusnya
    newTweets = api.search_tweets(q=search_key, lang="en", count=tweetsPerQuery)

    # mengambil atribut tertentu dari suatu tweet
    for i in newTweets:
        record = str(i.user.id_str)
        record += ';'
        record = str(i.user.name)
        record += ';'
        # record += str(normalize_timestamp(str(i.created_at)))
        # record += ';'
        # record += str(i.full_text.encode('utf-8'))
        record += str(i.text.encode('utf-8'))
        record += ';'
        print(str.encode(record))
        producer.send(topic_name, str.encode(record))

        tweet_id.append(str(i.user.id_str))
        tweet_username.append(str(i.user.name))
        tweet_text.append(str(i.text.encode('utf-8')))
        tweets = [str(i.user.id_str), str(i.user.name),
str(i.text.encode('utf-8'))]
        csvWriter.writerow(tweets)

    # menambah jumlah TweetCount dan MaxId
    tweetCount += len(newTweets)
    maxId = newTweets[-1].id

# mencoba mencetak
dictTweets = {"id":tweet_id, "username":tweet_username, "text":tweet_text}
df = pd.DataFrame(dictTweets)
# df

```

Pada kode diatas menggunakan library tweepy, namun terlihat kode diatas juga membutuhkan KafkaProducer untuk menyimpan record. Dengan demikian untuk proses streaming ini tetap membutuhkan library Kafka. Karena pada proyek ini menggunakan penjadwalan maka modul python ini tidak dijalankan langsung, simpan modul ini di direktori yang sama dengan modul untuk proses ETL nantinya.

6. PROSES ETL

- Dari modul untuk streaming data sebelumnya, akan dihasilkan sebuah file CSV. File CSV yang diperoleh kemudian diekstrak ke dalam dataframe pandas untuk dilakukan perbaikan pada data hasil *streaming*, karena disini semua proses dilakukan dengan jadwal yang teratur dengan menggunakan Airflow, maka untuk proses ETL sama seperti *streaming data*, dibuat dalam modul yang terpisah. Berikut adalah kode program untuk proses ekstraksi data.

```

import pandas as pd
kolom = ['id', 'username', 'tweet']

```

```
#Direktori dapat disesuaikan dengan tempat path tempat file python, csv dan db
disimpan
df_windows = pd.read_csv('/home/stndb01/Documents/Data_Engineering/Proyek/windows
11.csv', names = kolom)
df_windows
df_windows.duplicated().value_counts()

df_windows.to_csv('/home/stndb01/Documents/Data_Engineering/Proyek/windows_11_e.cs
v')
```

Sebagai tambahan, karena Airflow berjalan di direktori instalasi Airflow itu sendiri, sedangkan direktori modul stream dan ETL bisa jadi berbeda, maka kita dapat menjalankan modul-modul yang digunakan pada *virtual environment* python ataupun dengan menuliskan direktori lengkap dari file-file yang dipengaruhi oleh modul python yang digunakan.

- b. Dari data hasil *streaming* pada file csv tersebut, akan diperoleh sekitar 3000 data namun hanya sedikit data yang berbeda. Sehingga kita menjalankan perintah `df_windows.duplicated().value_counts()` untuk melihat jumlah data yang berbeda.
- c. Data yang sudah terekstrak tadi disimpan ke dalam file CSV baru dengan menambahkan *suffix* “_e” pada bagian akhir namanya untuk menandakan bahwa file CSV ini adalah file hasil ekstraksi. Sama seperti modul *streaming*, modul untuk proses ekstraksi ini tidak dijalankan dahulu.
- d. Dari file CSV hasil ekstraksi sebelumnya, dilakukan proses transformasi dimana data yang dihasilkan diperbaiki dengan cara menghapus data duplikat. Sehingga dari dataframe tersebut, dibuat sebuah modul baru untuk ekstraksi yang menjalankan perintah `df_windows.drop_duplicates(inplace=True, ignore_index=True)` untuk menghapus data yang duplikat, berikut adalah kode lengkap untuk proses transformasi.

```
import pandas as pd
kolom = ['id', 'username', 'tweet']
df_windows =
pd.read_csv('/home/stndb01/Documents/Data_Engineering/Proyek/windows_11_e.csv',
names = kolom)
#df_windows.duplicated().value_counts()

df_windows.drop_duplicates(inplace=True, ignore_index=True)
df_windows
df_windows.to_csv('/home/stndb01/Documents/Data_Engineering/Proyek/windows_11_t.cs
v')
```

- e. Data yang sudah ditransformasi tadi disimpan ke dalam file CSV baru dengan menambahkan *suffix* “_t” pada bagian akhir namanya untuk menandakan bahwa file CSV ini adalah file hasil transformasi, meskipun sudah hasil transformasi tetapi data ini masih belum dapat langsung digunakan untuk proses *machine learning*. Sama seperti sebelumnya, modul ini tidak langsung dijalankan.
- f. Dari file CSV hasil transformasi sebelumnya, dilakukan proses load data ke dalam database SQLite3. Sehingga pada tahapan ini kita membuat connection dengan perintah `connect` dan menyimpannya ke variable `conn`, kemudian membuat cursor yang digunakan untuk mengeksekusi perintah database yang tidak dapat dieksekusi dengan menggunakan query saja, kemudian me-load data ke database menggunakan perintah `to_sql`. Berikut adalah kode lengkap untuk proses load data.

```
import sqlite3 as s3
import pandas as pd
import os

# membuat koneksi ke database
conn =
s3.connect('/home/stndb01/Documents/Data_Engineering/Proyek/windows11_data.db')

# Object cursor untuk menjalankan perintah SQL
cur = conn.cursor()

# load dataframe ke datamart
```



```
df_windows =
pd.read_csv('/home/stndb01/Documents/Data_Engineering/Proyek/windows_11_t.csv')
df_windows.to_sql('windows11_table',conn,if_exists='replace',index=False)
os.remove("/home/stndb01/Documents/Data_Engineering/Proyek/windows_11_e.csv")
os.remove("/home/stndb01/Documents/Data_Engineering/Proyek/windows_11_t.csv")
```

- g. Data yang sebelumnya telah ditransformasikan disimpan ke dalam file database baru, kemudian file hasil ekstraksi dan transformasi tadi dihapus dengan menggunakan perintah `remove` dari *library* OS. Dengan demikian keempat modul yang telah dibuat sebelumnya siap untuk digunakan melalui penjadwalan Airflow.

7. SCHEDULING

- Untuk proses scheduling dapat dilakukan dengan cara membuat modul python layaknya proses-proses sebelumnya. Import library yang akan digunakan seperti `datetime`, `BashOperator`, `DAG`, dan `days_ago`.
- Inisialisasi default argument, yang perlu diperhatikan diisi disini adalah owner saja agar DAG yang dibuat nantinya muncul pada akun yang sesuai dengan owner yang telah didefinisikan. Untuk email tidak masalah tidak diisi dengan benar. Pada blok kode berikutnya definisikan DAG yang kita buat, hal ini meliputi nama atau id dari DAG yang dibuat, default argumennya, deskripsi, interval (dalam hal ini kami mengaturnya dengan `days = 1` yang berarti jalan setiap 24 jam, dapat juga ditulis dalam notasi `0 0 * * *` ataupun `@daily`), start date dan tag yang sifatnya opsional.
- Proses berikutnya adalah mendefinisikan operasi yang akan dijalankan, dapat memberikan nama operasinya terlebih dahulu, kemudian mendefinisikan operasi apa yang akan dijalankan oleh `BashOperator`. Dalam hal ini yang perlu didefinisikan pada `BashOperator` adalah task id dan bash command dimana command atau perintah yang dijalankan adalah perintah untuk eksekusi modul python yang telah dibuat untuk proses stream dan ETL.
- Setelah mendefinisikan operasi yang akan dibuat, kemudian proses berikutnya adalah mendefinisikan urutan eksekusinya dengan menggunakan syntax `set_downstream` yang mana urutan yang akan diperoleh dengan syntax tersebut akan bersifat `sequential`. Dengan demikian berikut adalah kode lengkap untuk proses scheduling.

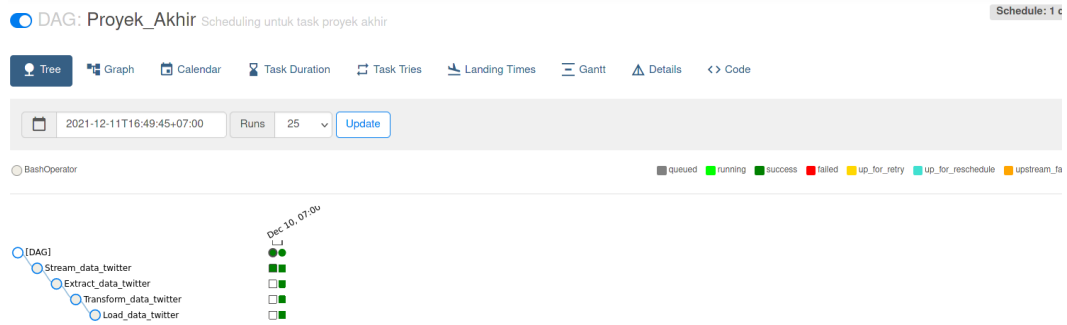
```
from datetime import timedelta
from airflow.operators.bash import BashOperator
from airflow.utils.dates import days_ago
from airflow import DAG

default_args = {
    'owner': 'admin',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
}

with DAG(
    'Proyek_Akhir',
    default_args=default_args,
    description='Scheduling untuk task proyek akhir',
    schedule_interval=timedelta(days=1),
    start_date=days_ago(2),
    tags=['proyek'],
) as dag:

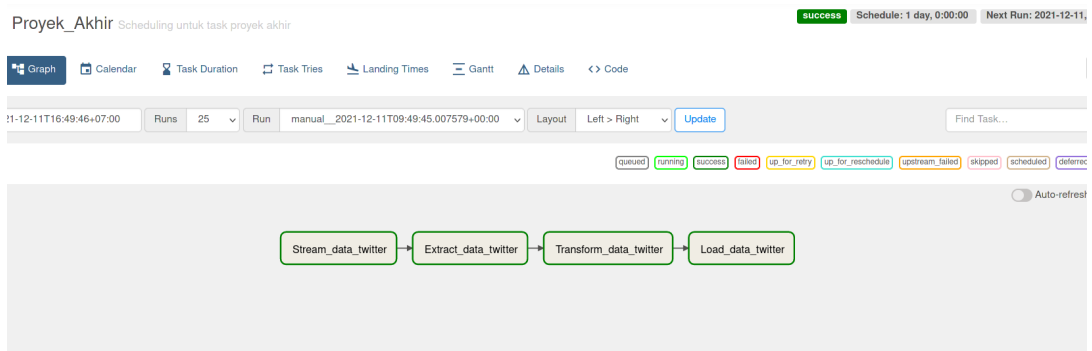
    stream = BashOperator(
        task_id='Stream_data_twitter',
        bash_command='python
/home/stndb01/Documents/Data_Engineering/Proyek/stream.py'
        #bash_command='python stream.py',
    )

    extract = BashOperator(
        task_id='Extract_data_twitter',
        bash_command='python
/home/stndb01/Documents/Data_Engineering/Proyek/extract.py'
```

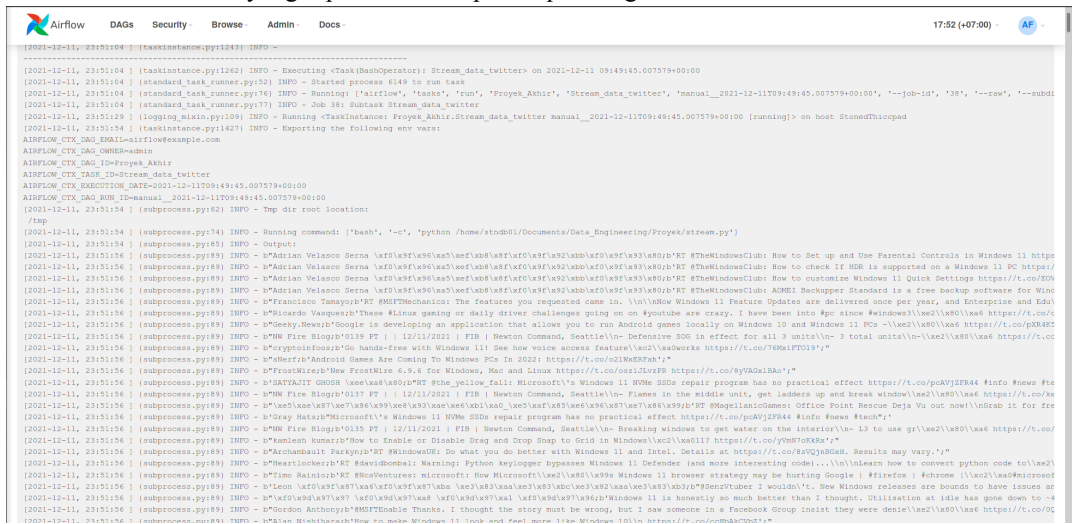
Gambar 21. Proses scheduling berhasil

Hal yang sama juga dapat diperhatikan pada tampilan model graph, proses yang berhasil dijalankan akan berwarna hijau tua juga.



Gambar 22. Proses scheduling berhasil pada tampilan Graph

- g. Proses dengan menggunakan scheduling ini tidak dapat diamati secara langsung, hal yang dapat diketahui hanyalah proses berjalan dengan sukses atau tidak. Untuk itu agar dapat melihat prosesnya kita dapat melihat pada bagian log dari salah satu proses, berikut adalah tampilan log ketika melakukan stream data dimana data yang diperoleh ditampilkan pada log terminal.



Gambar 23. Log hasil streaming data

Great expectation digunakan pada proses pipeline. Great expectation merupakan metode validasi data (data validation) yang bertujuan untuk memastikan data yang diproses pada tahap pipeline benar dan bebas dari permasalahan kualitas data yang dapat memicu error akibat kesalahan input atau transformasi. Terdapat dua pertimbangan mengapa kami tidak menggunakan Great Expectation dalam project kami.

Pertama. Pada project kami, pipeline dilakukan dengan streaming data yang bersumber dari twitter dimana dalam prosesnya memanfaatkan API khusus yang diberikan oleh twitter. API sendiri bekerja sebagai perantara antar aplikasi yang digunakan untuk "komunikasi" seperti pengambilan data. Singkatnya, kami

memperoleh informasi yang sudah ditentukan oleh twitter itu sendiri, pada project kami berupa id twitter, nama akun, dan isi tweet yang berupa data text. Tentunya data text yang diberikan oleh twitter sudah sepatutnya konsisten sehingga bebas dari error.

Kedua. Great Expectation bekerja dengan cara membuat pernyataan atau konteks dari data yang disebut dengan *Expectation* dan memvalidasi data apapun menggunakan Expectation tersebut. Data utama yang kami stream adalah text data atau isi tweet yang mengandung kata kunci yang diberikan, sehingga menurut kami validasi teks tidak diperlukan dalam hal ini karena tidak membutuhkan spesifikasi tertentu.

8. PERSIAPAN MODEL MACHINE LEARNING

Berikut ini adalah prosedur persiapan data untuk model machine learning agar data tersebut siap digunakan dalam sebuah model machine learning nantinya:

- a. Read Data, Data didapatkan melalui database yang sudah di-upload sebelumnya pada datamart SQLite3. Pertama yang dilakukan adalah dengan membuat koneksi ke database serta membuat query. Data dibaca dengan menggunakan method `read_sql` yang disimpan dalam variable `df` untuk data frame nya

```
# membuat koneksi ke database
conn = s3.connect('windows11_data.db')

# Object cursor untuk menjalankan perintah SQL
cur = conn.cursor()
query = '''select * from windows11_table'''
df = pd.read_sql(query, conn)
df
```

id	username	tweet
60951889	60951889	b'RT @lockheimer: We're bringing Play Games to Windows 10 and 11! Prett...
1408089628764586000	theolreverendsburner	b'Macaulay-Tonna @Halo My 2 cents: do a clean Windows 11 installation ...
1082920971114332200	Ricoh eShop UK	b'*PRODUCT*\n\nThe Lenovo notebook Laptops with NEW windows 11 are now ...
2198733733	Cleodesktop	b'After Dark Gray Theme For Windows 11 - \nGet it here: ...
1052557647856189400	Beyte Fyr	b'Should I upgrade to Windows 11? I'm angsty about this."
2281129274	Mehmet	b'Windows 11 Features #1 Color Filters\n#Windows11 #ColorFilters ...
2571317438	鳥見人(とりむぎやう)	b'RT @lockheimer: We're bringing Play Games to Windows 10 and 11! Prett...
1274775491006922000	Ash Collins	b'So I had a look at windows 11 the other day and uh... Why is ...
3064146644	SATYAJIT GHOSH	b'RT @androidcodex: So I had a look at windows 11 the other day and uh... ..
1469317591445524500	Soumyashree Biswal	b'When you don't wanna risk downloading windows 11 prematurely so you ...

Gambar 24. Data yang diperoleh dari proses ETL

- b. Berikutnya melakukan Pre-Processing kembali. Tujuan dari preprocessing yang dilakukan pada dataset ini adalah dengan menghapus text - text yang tidak diperlukan seperti emoji, link http, simbol - simbol dan lainnya yang sebelumnya tidak dibersihkan dengan baik pada proses ETL. Hal ini perlu dilakukan karena data stream yang diperoleh sebelumnya disimpan dalam bentuk byte, sehingga terdapat beberapa karakter yang dapat merusak proses analisis sentimen seperti emoji yang dirubah menjadi UTF-8, username, URL, escape character dan lain-lain. Proses pre-processing dilakukan dengan memanfaatkan fungsi regex untuk menghapus kata yang tidak digunakan.

```
def preprocess_text(review):
    # using regex to replace
    review = codecs.decode(review, 'unicode_escape') #remove escape character
    review = review[2:-1]
    review = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL', review)
    review = re.sub('[^x00-x7f]', '', review)
    review = re.sub('@[^\s]+','USER', review)
    review = review.lower().replace("ë", "e")
    review = re.sub('[^a-zA-Za-zA-яA-Я1-9]+', ' ', review)
    review = re.sub(' +', ' ', review)
    return review.strip()

df['tweet'] = [preprocess_text(review) for review in df['tweet']]
```

9. MODEL MACHINE LEARNING

Berikut ini adalah prosedur membuat model machine learning berupa Sentiment Analysis yang diimplementasikan menggunakan Text Blob:

- Membuat fungsi `sentiment_analysis` dengan menggunakan library `Text Blob`, yang pertama kali dicari adalah mencari nilai dari `subjectivity` dan `polarity` dari tweet yang diperoleh, kemudian menentukan nilai analisisnya apakah negatif, netral ataupun positif. Ketiga hasil tersebut nantinya ditambahkan menjadi sebuah kolom tabel baru pada dataframe sebelumnya.

```
from textblob import TextBlob

def sentiment_analysis(data):
    def getSubjectivity(text):
        return TextBlob(text).sentiment.subjectivity

    #Create a function to get the polarity
    def getPolarity(text):
        return TextBlob(text).sentiment.polarity

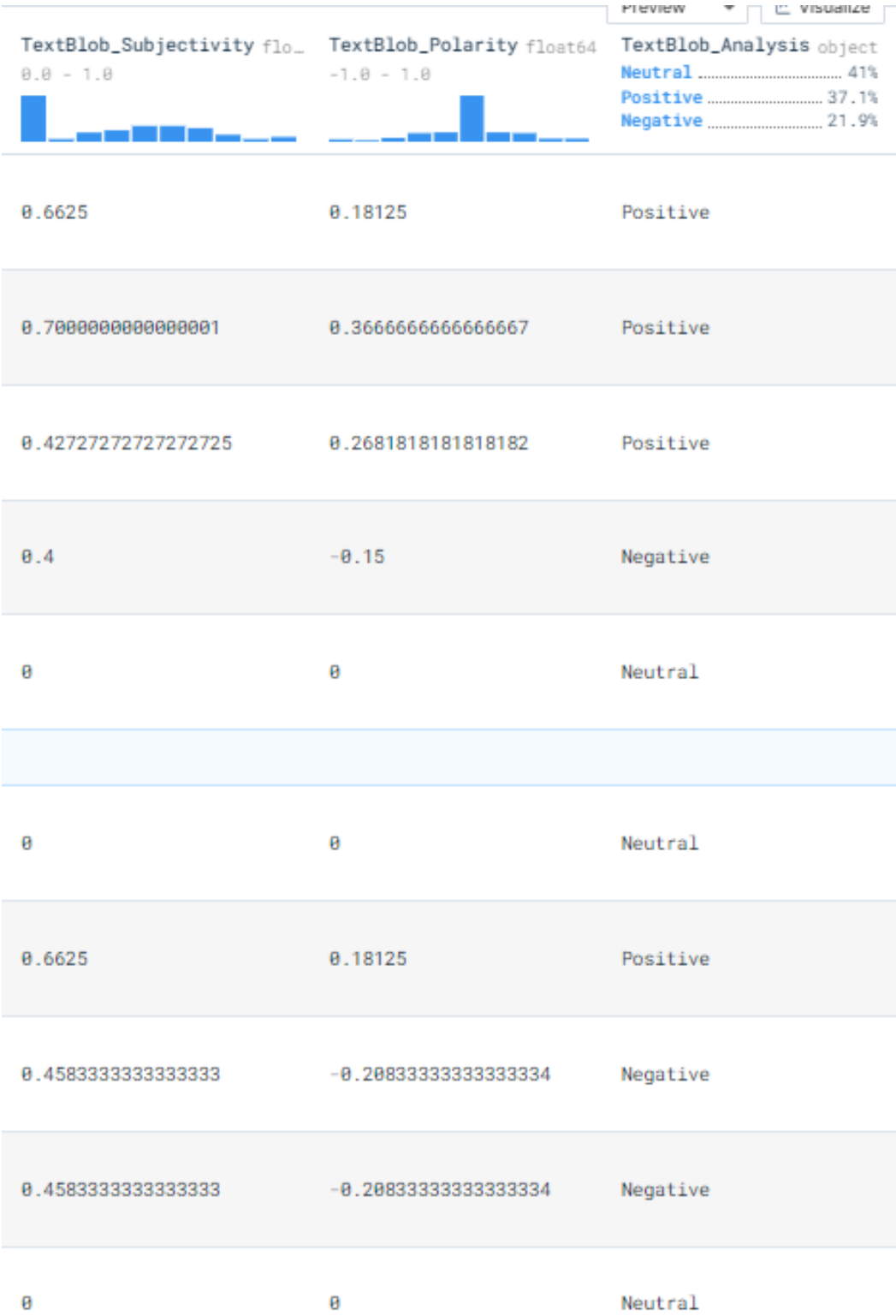
    def getAnalysis(score):
        if score < 0:
            return "Negative"
        elif score == 0:
            return "Neutral"
        else:
            return "Positive"

    #Create two new columns 'Subjectivity' & 'Polarity'
    data["TextBlob_Subjectivity"] = data["tweet"].apply(getSubjectivity)
    data["TextBlob_Polarity"] = data["tweet"].apply(getPolarity)

    data["TextBlob_Analysis"] = data["TextBlob_Polarity"].apply(getAnalysis)
    return data
```

- b. Memanggil fungsi sentiment_analysis terhadap dataframe yang digunakan kemudian menampilkan hasil TextBlob_Subjectivity, TextBlob_Polarity dan TextBlob_Analysis, sehingga diperoleh tampilan seperti gambar dibawah.

```
df = sentiment_analysis(df)
df
```



Gambar 25. Tampilan tiga kolom baru pada dataframe

- c. Selanjutnya yang dilakukan adalah mencari 20 kata yang paling sering muncul pada data tweet yang mana dapat dikatakan juga sebagai salah satu bentuk exploratory data analysis. Membuat sebuah fungsi bernama `getMostFrequentWord` yang digunakan untuk mencari kata yang sering muncul tersebut.

```
from collections import Counter
```

```
def getMostFrequentWord(df):
    most_freq = Counter(" ".join(df["tweet"]).split()).most_common(30)

    most_freq_filtered = {}
    for i in most_freq:
        if(i[0] == 'user' or i[0] == 'url' or i[0] == 'rt'):
            continue
        else:
            most_freq_filtered[i[0]] = i[1]
            if(len(most_freq_filtered) == 20): # ambil 20 kata paling banyak
                break

    return most_freq_filtered
```

```
most_freq = getMostFrequentWord(df)
most_freq
```

```
{'windows': 184,
 '11': 92,
 'to': 57,
 'and': 39,
 'the': 39,
 'is': 29,
 '1': 25,
 'a': 22,
 'on': 22,
 'it': 22,
 'i': 20,
 'microsoft': 17,
 'you': 16,
 'like': 16,
 'games': 15,
 'with': 15,
 'so': 15,
 'of': 15,
 'google': 15,
 'play': 14}
```

Gambar 26. 20 data yang sering muncul pada data

10. DASHBOARD

Berikut ini adalah prosedur Dashboarding yang ditujukan untuk mengidentifikasi perspektif pengguna twitter terhadap kata kunci “Windows 11” serta mengidentifikasi kata - kata yang sering muncul berdasarkan kata kunci tersebut:

- Mendefinisikan fungsi untuk membuat dashboard dengan memanfaatkan library seaborns. Pada proses ini dashboarding dilakukan pada dua data yaitu data hasil analisis sentimen sebagai figur satu yang ditampilkan dalam bentuk pie chart, dan data hasil EDA atau pengecekan terhadap kata yang sering muncul sebagai figur dua yang ditampilkan dalam bentuk histogram. Berikut adalah kode lengkap untuk proses dashboarding.

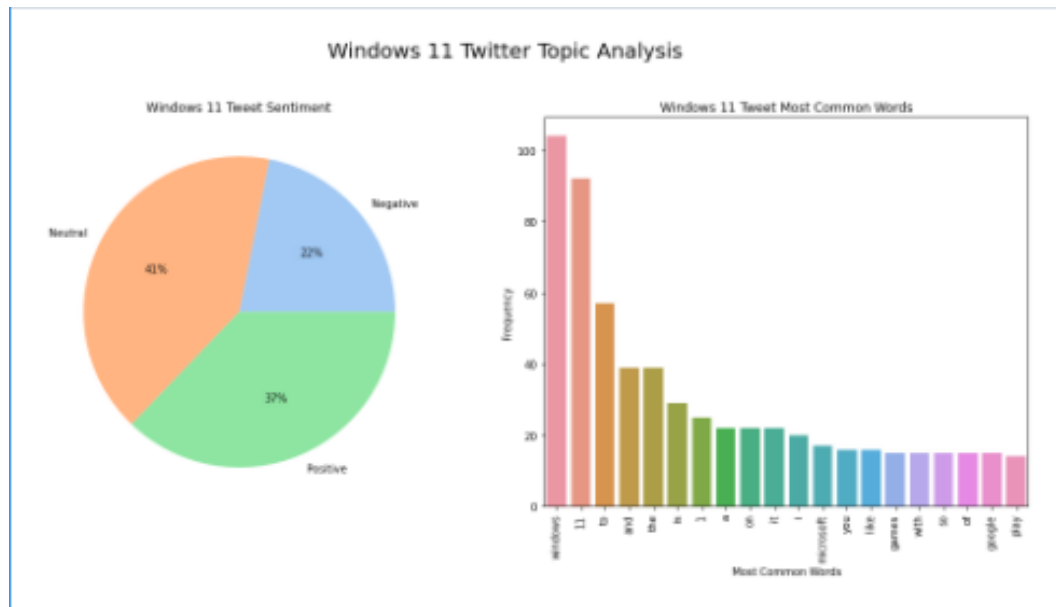
```
f, axes = plt.subplots(1, 2, figsize = (15, 7))
# sns.set_style("white")
colors = sns.color_palette('pastel')[0:5]
# colors = sns.color_palette('bright')[0:5]
f.suptitle("{} Twitter Topic Analysis".format(TOPIC), fontsize=20)
f.tight_layout(pad=3.0)

# figure 1
```

```
ax1 = plt.subplot2grid((1,2),(0,0))
ax1.title.set_text("{} Tweet Sentiment".format(TOPIC))
fig1 = plt.pie(df.groupby("TextBlob_Analysis").size(), labels =
df.groupby("TextBlob_Analysis").size().index, colors = colors, autopct='%0.0f%%')

# figure 2
ax2 = plt.subplot2grid((1,2),(0,1))
ax2.title.set_text("{} Tweet Most Common Words".format(TOPIC))
fig2 = sns.barplot(x=list(most_freq.keys()), y=list(most_freq.values()))
fig2.set(xlabel='Most Common Words', ylabel='Frequency')
for item in fig2.get_xticklabels():
    item.set_rotation(90)
```

b. Hasil Dashboard dapat dilihat pada gambar dibawah.



Gambar 27. Dashboard


DAFTAR PUSTAKA

- 5, J. and Rowe, W., 2017. Working with streaming Twitter data using Kafka. [online] BMC Blogs. Available at: <<https://www.bmc.com/blogs/working-streaming-twitter-data-using-kafka/>> [Accessed 11 Dec. 2021].
- Airflow, A., 2015. Tutorial Airflow. [online] Tutorial - Airflow Documentation. Available at: <<https://airflow.apache.org/docs/apache-airflow/stable/tutorial.html>> [Accessed 11 Dec. 2021].
- Alayil, M. and Admin, 2021. Setting up and running Apache Kafka on windows. [online] Goavega. Available at: <<https://www.goavega.com/install-apache-kafka-on-windows/>> [Accessed 11 Dec. 2021].
- Bull, J., 2020. How to install Apache Airflow. [online] Medium. Available at: <<https://medium.com/@jacksonbull1987/how-to-install-apache-airflow-6b8a2ae60050>> [Accessed 11 Dec. 2021].
- DigitalOcean, 2021. How to test your data with Great Expectations. [online] DigitalOcean. Available at: <<https://www.digitalocean.com/community/tutorials/how-to-test-your-data-with-great-expectations>> [Accessed 11 Dec. 2021].
- ES, S., 2021. Sentiment Analysis in Python: TextBlob vs Vader Sentiment vs Flair vs Building It From Scratch. [online] neptune.ai. Available at: <<https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair>> [Accessed 11 Dec. 2021].
- Heronius, A., 2019. Twitter sentiment analysis Bahasa Indonesia dengan textblob. [online] Medium. Available at:

- <<https://medium.com/@albertusheronius/twitter-sentiment-analysis-bahasa-indonesia-dengan-textblob-f34e1ffdcdaa>> [Accessed 11 Dec. 2021].
- Shah, P., 2020. Sentiment Analysis using TextBlob. [online] Medium. Available at: <<https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524>> [Accessed 11 Dec. 2021].
- Sharma, A., 2020. Exploratory data analysis for text data: EDA using python. [online] Analytics Vidhya. Available at: <<https://www.analyticsvidhya.com/blog/2020/04/beginners-guide-exploratory-data-analysis-text-data/>> [Accessed 11 Dec. 2021].
- SuperDataWorld, 2021. How to create a python dashboard (Seaborn / Jupyter Notebook). [online] YouTube. Available at: <<https://www.youtube.com/watch?v=t95bqksypic>> [Accessed 11 Dec. 2021].
- Susanto, G.K., 2021. Data Pipeline using Apache airflow to import data from public API. [online] Medium. Available at: <<https://medium.com/jakartasmartcity/data-pipeline-using-apache-airflow-to-import-data-from-public-api-7ff719118ac8>> [Accessed 11 Dec. 2021].
- Yunus, M., 2020. Apache airflow: Membuat Pipeline Sederhana Untuk pertama kali. [online] Medium. Available at: <<https://yunusmuhammad007.medium.com/apache-airflow-membuat-pipeline-sederhana-untuk-pertama-kali-8a2e771a45ff>> [Accessed 11 Dec. 2021].

PENULIS

	<p>Peran dalam project:</p> <ul style="list-style-type: none"> - Ketua kelompok - Membantu penyusunan laporan, power point dan video presentasi - Menyediakan environment implementasi proyek - Menyesuaikan kode program dengan environment yang digunakan - Merencanakan proses ETL - Membuat kode sumber proses ETL - Merencanakan proses streaming - Membuat kode sumber streaming - Merencanakan proses scheduling - Membuat kode sumber scheduling - Menentukan proses machine learning - Menentukan konten dashboard <p>No whatsapp: 0823-3488-8196</p>
	<p>Peran dalam project:</p> <ul style="list-style-type: none"> - Membantu penyusunan laporan, power point dan video presentasi - Merencanakan proses ETL - Membuat kode sumber proses ETL - Merencanakan proses streaming - Membuat kode sumber streaming - Merencanakan proses scheduling - Menentukan proses machine learning - Membuat kode sumber machine learning - Menentukan konten dashboard - Membuat kode sumber dashboard <p>No whatsapp: 0821-3192-9796</p>

	<p>Peran dalam project:</p> <ul style="list-style-type: none">- Membantu penyusunan laporan, power point dan video presentasi- Merencanakan proses ETL- Membuat kode sumber proses ETL- Merencanakan proses streaming- Membuat kode sumber streaming- Merencanakan proses scheduling- Menentukan proses machine learning- Menentukan konten dashboard <p>No whatsapp: 0813-8899-3033</p>
---	---