

Analisis Perilaku Pengguna Twitter Menggunakan Apache Spark

Disusun untuk memenuhi tugas mata kuliah Big Data

Dosen Pengampu : Dr. Lala Septem Riza, M.T., Ph.D

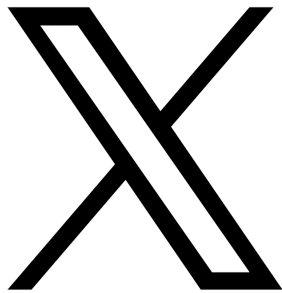


Disusun Oleh :

Nabilla Assyfa Ramadhani	(NIM. 2205297)
Revana Faliha Salma	(NIM. 2202869)
Roshan Syalwan Nurilham	(NIM. 2203142)

**Program Studi Ilmu Komputer
Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam
Universitas Pendidikan Indonesia
2024**

1. Pengertian Twitter/X



Twitter atau yang sekarang beralih nama menjadi “X” adalah platform media sosial yang memungkinkan pengguna untuk berbagi pemikiran, informasi, dan berinteraksi dengan pengguna lain dalam bentuk pesan singkat yang disebut “tweet”. Tweet ini terbatas pada 280 karakter, meskipun awalnya dibatasi hanya 140 karakter. X merupakan salah satu platform yang paling populer di dunia, digunakan oleh individu, organisasi, dan perusahaan untuk berbagai tujuan, termasuk berita, pemasaran, dan interaksi. Dalam X, pengguna memiliki dua konsep utama yang membentuk jaringan sosial mereka, yaitu “pengikut” dan “mengikuti”. “Pengikut” adalah orang-orang atau akun-akun lain yang memilih untuk mengikuti akun pengguna tersebut. Mereka akan melihat postingan pengguna tersebut di feed mereka dan menjadi bagian dari audiensnya. Selain itu, “mengikuti” merujuk kepada tindakan mengikuti akun-akun lain. Ketika pengguna mengikuti akun lain, mereka akan melihat postingan dari akun tersebut di feed mereka.

2. Dataset

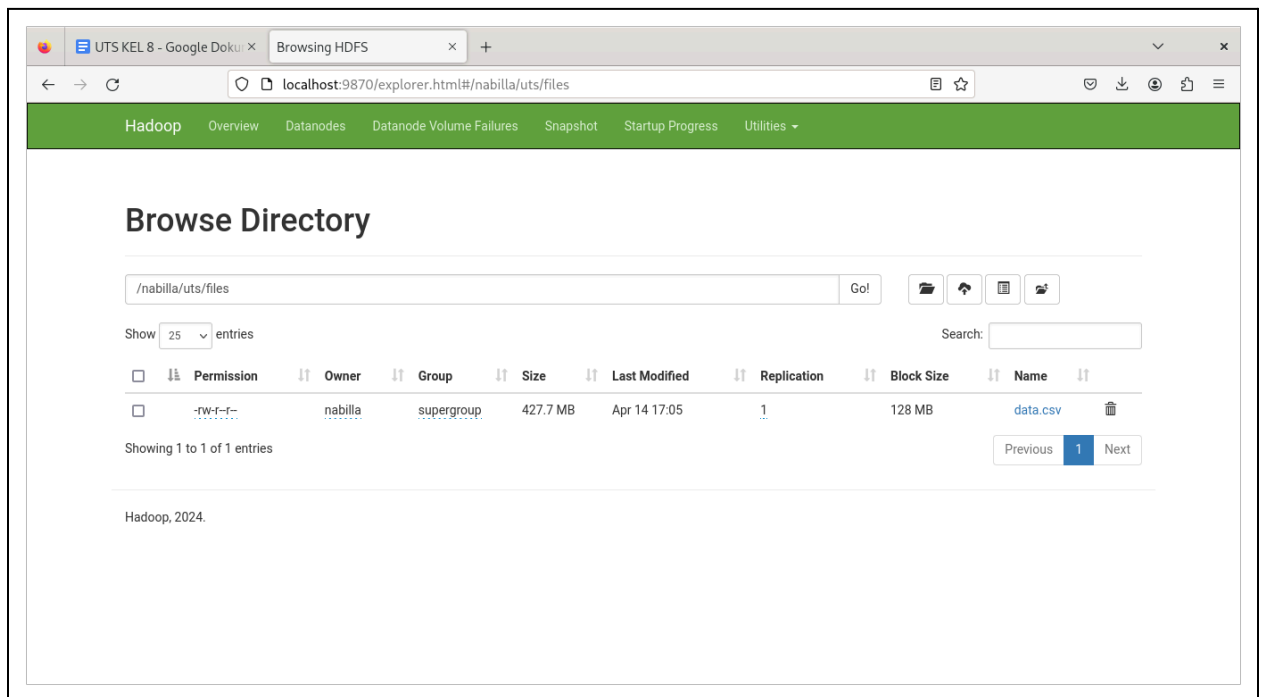
Dataset yang digunakan dalam analisis ini diambil dari situs Kaggle. Dataset ini berisi 40 ribu baris data dengan beberapa kolom, yaitu:

- 1) avatar : url gambar profil
- 2) followerCount : jumlah pengikut
- 3) friendsCount : jumlah yang diikuti
- 4) friendName : nama pengguna (tanpa “@”)
- 5) id : id pengguna
- 6) friends : daftar id yang diikuti pengguna (teman)
- 7) lang : bahasa yang dinyatakan oleh pengguna
- 8) lastSeen : waktu terakhir pengguna melakukan tweet
- 9) tags : tagar yang digunakan pengguna (dengan atau tanpa “#”)
- 10) tweetID : id dari tweet terakhir yang diposting oleh pengguna

3. Tujuan

Tugas ini bertujuan untuk menganalisis perilaku pengguna media sosial, terutama platform Twitter. Analisis dilakukan dengan cara membandingkan dua nilai, yaitu jumlah pengikut (followers) dan jumlah teman (friends). Rasio ini akan memberikan gambaran tentang aktivitas pengguna Twitter. Selain itu, analisis ini akan menghasilkan dua pengelompokan pengguna Twitter, yaitu pengguna biasa dan influencer. Setelahnya, kita bisa mendapatkan informasi mengenai Tag (Topik Popular) apa saja yang sedang marak dibahas oleh para influencer.

4. Menyimpan data



Data disimpan kedalam HDFS menggunakan perintah put, dengan sintaks sebagai berikut:

Unset

```
nabilla@localhost:~> hdfs dfs -put  
/home/nabilla/Downloads/data.csv /nabilla/uts/files/
```

```
nabilla@localhost:~$ hdfs fsck /nabilla/uts/files/data.csv -files -blocks -locations
Connecting to namenode via http://localhost:9870/fsck?ugi=nabilla&files=1&blocks=1&locations=1&path=%2Fnabilla%2Futs%2Ffiles%2Fdata.csv
FSCK started by nabilla (auth:SIMPLE) from /127.0.0.1 for path /nabilla/uts/files/data.csv at Sun Apr 14 17:11:13 WIB 2024

/nabilla/uts/files/data.csv 448476867 bytes, replicated: replication=1, 4 block(s): OK
0 BP-1862810082-127.0.0.1-1711081415590:blk_1073741837_1013 len=134217728 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-902eaaca-be91-47cb-8781-e5e48f14b872,DISK]]
1 BP-1862810082-127.0.0.1-1711081415590:blk_1073741838_1014 len=134217728 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-902eaaca-be91-47cb-8781-e5e48f14b872,DISK]]
2 BP-1862810082-127.0.0.1-1711081415590:blk_1073741839_1015 len=134217728 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-902eaaca-be91-47cb-8781-e5e48f14b872,DISK]]
3 BP-1862810082-127.0.0.1-1711081415590:blk_1073741840_1016 len=45823683 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-902eaaca-be91-47cb-8781-e5e48f14b872,DISK]]

Status: HEALTHY
Number of data-nodes: 1
Number of racks: 1
Total dirs: 0
Total symlinks: 0

Replicated Blocks:
Total size: 448476867 B
Total files: 1
Total blocks (validated): 4 (avg. block size 112119216 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Blocks queued for replication: 0

Erasure Coded Block Groups:
```

```
rerevana@localhost.localdomain:~/Hadoop/sbin$ hdfs fsck /rerevana/hadoop/UTS/data.csv -files -blocks -locations
Connecting to namenode via http://localhost:9870/fsck?ugi=rerevana&files=1&blocks=1&locations=1&path=%2Frerevana%2FHadoop%2FUTS%2Fdata.csv
FSCK started by rerevana (auth:SIMPLE) from /127.0.0.1 for path /rerevana/hadoop/UTS/data.csv at Mon Apr 15 06:05:26 WIB 2024

/rerevana/hadoop/UTS/data.csv 448476867 bytes, replicated: replication=1, 4 block(s): Under replicated BP-2120847674-127.0.0.1-1711335088425:blk_1073741825_1001. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
Under replicated BP-2120847674-127.0.0.1-1711335088425:blk_1073741826_1002. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
Under replicated BP-2120847674-127.0.0.1-1711335088425:blk_1073741827_1003. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
Under replicated BP-2120847674-127.0.0.1-1711335088425:blk_1073741828_1004. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
0 BP-2120847674-127.0.0.1-1711335088425:blk_1073741825_1001 len=134217728 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-3f4544f0-027f-49da-934c-70e45a850b4f,DISK]]
1 BP-2120847674-127.0.0.1-1711335088425:blk_1073741826_1002 len=134217728 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-3f4544f0-027f-49da-934c-70e45a850b4f,DISK]]
2 BP-2120847674-127.0.0.1-1711335088425:blk_1073741827_1003 len=134217728 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-3f4544f0-027f-49da-934c-70e45a850b4f,DISK]]
3 BP-2120847674-127.0.0.1-1711335088425:blk_1073741828_1004 len=45823683 Live_rep=1 [DatanodeInfoWithStorage[127.0.0.1:9866,DS-3f4544f0-027f-49da-934c-70e45a850b4f,DISK]]

Status: HEALTHY
Number of data-nodes: 1
Number of racks: 1
Total dirs: 0
Total symlinks: 0

Replicated Blocks:
Total size: 448476867 B
Total files: 1
Total blocks (validated): 4 (avg. block size 112119216 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 4 (100.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 8 (66.666664 %)
Blocks queued for replication: 0

Erasure Coded Block Groups:
Total size: 0 B
Total files: 0
Total block groups (validated): 0
Minimally erasure-coded block groups: 0
Over-erasure-coded block groups: 0
Under-erasure-coded block groups: 0
Unsatisfactory placement block groups: 0
Average block group size: 0.0
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0
Blocks queued for replication: 0
FSCK ended at Mon Apr 15 06:05:26 WIB 2024 in 5 milliseconds

The filesystem under path '/rerevana/hadoop/UTS/data.csv' is HEALTHY
rerevana@localhost:~/Hadoop/sbin$
```

File dataset berukuran 428 MB dan ukuran blok HDFS yang telah ditetapkan adalah 128 MB. HDFS membagi data tersebut kedalam blok - blok kecil yang berukuran 128 MB, sebanyak 4 blok.

Kapasitas Log HD

```
nabilla@localhost:~$ hdfs dfs -du -h /nabilla/uts/files
427.7 M 427.7 M /nabilla/uts/files/data.csv
nabilla@localhost:~$ hdfs dfsadmin -report
Configured Capacity: 52640612352 (49.03 GB)
Present Capacity: 40752066560 (37.95 GB)
DFS Remaining: 40299360256 (37.53 GB)
DFS Used: 452706304 (431.73 MB)
DFS Used%: 1.11%
Replicated Blocks:
  Under replicated blocks: 0
  Blocks with corrupt replicas: 0
  Missing blocks: 0
  Missing blocks (with replication factor 1): 0
  Low redundancy blocks with highest priority to recover: 0
  Pending deletion blocks: 0
Erasure Coded Block Groups:
  Low redundancy block groups: 0
  Block groups with corrupt internal blocks: 0
  Missing block groups: 0
  Low redundancy blocks with highest priority to recover: 0
  Pending deletion blocks: 0
-----
Live datanodes (1):
Name: 127.0.0.1:9866 (localhost.localdomain)
Hostname: localhost.localdomain
Decommission Status : Normal
Configured Capacity: 52640612352 (49.03 GB)
DFS Used: 452706304 (431.73 MB)
Non DFS Used: 11286450176 (10.51 GB)
DFS Remaining: 40299360256 (37.53 GB)
DFS Used%: 0.86%
DFS Remaining%: 76.56%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
```

```
rerevana@localhost.localdomain:~/Hadoop/sbin$ hdfs dfs -du -h /rerevana/hadoop/UTS
427.7 M 1.3 G /rerevana/hadoop/UTS/data.csv
rerevana@localhost:~/Hadoop/sbin$ hdfs dfsadmin -report
Configured Capacity: 8040919040 (7.49 GB)
Present Capacity: 8039632896 (7.49 GB)
DFS Remaining: 7587627008 (7.07 GB)
DFS Used: 452005888 (431.07 MB)
DFS Used%: 5.62%
Replicated Blocks:
  Under replicated blocks: 4
  Blocks with corrupt replicas: 0
  Missing blocks: 0
  Missing blocks (with replication factor 1): 0
  Low redundancy blocks with highest priority to recover: 4
  Pending deletion blocks: 0
Erasure Coded Block Groups:
  Low redundancy block groups: 0
  Block groups with corrupt internal blocks: 0
  Missing block groups: 0
  Low redundancy blocks with highest priority to recover: 0
  Pending deletion blocks: 0
-----
Live datanodes (1):
Name: 127.0.0.1:9866 (localhost)
Hostname: localhost
Decommission Status : Normal
Configured Capacity: 8040919040 (7.49 GB)
DFS Used: 452005888 (431.07 MB)
Non DFS Used: 1286144 (1.23 MB)
DFS Remaining: 7587627008 (7.07 GB)
DFS Used%: 5.62%
DFS Remaining%: 94.36%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 0
Last contact: Mon Apr 15 06:11:57 WIB 2024
Last Block Report: Mon Apr 15 05:51:57 WIB 2024
Num of Blocks: 4
rerevana@localhost:~/Hadoop/sbin$
```

5. Pseudocode

```
class TwitterUser:
    attributes:
        id: int (primary key)
        followers: int
        following: int

function calculate_mean_total_following(users: list of TwitterUser) -> float:
    total_following = 0
    for user in users:
        total_following += user.following
    return total_following / len(users)

function count_unique_tags(users: list of TwitterUser) -> dict:
    unique_tags = {}
    mean_total_following = calculate_mean_total_following(users)
    for user in users:
        if ((user.following >= mean_total_following && user.followers / user.following >= 11) ||
            (user.following < mean_total_following && user.followers - user.following >= 10000)):
            # Check condition and add user tag to unique_tags
            for tag in user.tags:
                if tag not in unique_tags:
                    unique_tags[tag] = 1
                else:
                    unique_tags[tag] += 1
    return unique_tags

function sort_and_display_tags(unique_tags: dict):
    sorted_tags = sorted(unique_tags.items(), key=lambda x: x[1], reverse=True)
    for tag, count in sorted_tags:
        print(tag + ": " + str(count))

# Example usage:
users = [user1, user2, ..., userN] # List of TwitterUser objects
unique_tags = count_unique_tags(users)
sort_and_display_tags(unique_tags)
```

6. Pra Proses

Pra proses adalah langkah yang dilakukan pada data sebelum data dimasukkan kedalam model. Langkah ini bertujuan untuk membersihkan, menormalkan atau mengubah data. Langkah-langkah yang dilakukan untuk membersihkan data adalah sebagai berikut:

1. Pembersihan data

Menghapus nilai null yang ada pada data

2. Pemilihan Fitur

Memilih fitur yang paling relevan dengan tujuan analisis

a. Sebelum

id	screenName	tags	avatar	followersCount	friendsCount	lang	lastSeen	tweetId	friends
1969527638	LingoMakeEmCum_	["#nationaldogday"]	http://pbs.twimg.com/profile_images/5342862	319		112 en	1.47227E+12	7.69311E+17	["1969574754"]
51878493	_notmichelle	["#nationaldogday"]	http://pbs.twimg.com/profile_images/7619776	275		115 en	1.47227E+12	7.69309E+17	["60789485"]
1393409100	jesseayye	["#narcos"]	http://pbs.twimg.com/profile_images/7132825	120		107 en	1.4728E+12	7.71623E+17	["86868062"]
232891415	MrBrianLloyd	["#gloryoutnow"]	http://pbs.twimg.com/profile_images/1334406	492		325 en	1.47227E+12	7.69308E+17	["361335082"]
7.1013E+17	sarahdorat_16	["#nationaldogday"]	http://pbs.twimg.com/profile_images/7671805	128		218 en	1.47227E+12	7.6931E+17	["1571896093"]
3649469655	wanderlustregui	["#veranontv2016"]	http://pbs.twimg.com/profile_images/7624006	479		131 en	1.47273E+12	7.71341E+17	["2401096388"]
99769502	andhesonit	["#felipe massa"]	http://pbs.twimg.com/profile_images/2750403	1875		111 en	1.47273E+12	7.71322E+17	["16874201"]
413415713	Jas_Thxku	["#narcos"]	http://pbs.twimg.com/profile_images/7637257	582		281 en	1.47281E+12	7.71633E+17	["2353374818"]
518053777	KLITZAU	["#nationaldogday"]	http://pbs.twimg.com/profile_images/7609112	688		186 en	1.47227E+12	7.69308E+17	["2432417994"]
3167964531	ThePettyHomo	["#nationaldogday"]	http://pbs.twimg.com/profile_images/7163857	204		226 en	1.47227E+12	7.69308E+17	["386533263"]
634516322	darkgreyxv	["#getwellsoonjackson"]	http://pbs.twimg.com/profile_images/7263620	859		193 en	1.47274E+12	7.71336E+17	["3072390181"]
1357134163	michellemichic	["#justinssexstapeleakedparty"]	http://pbs.twimg.com/profile_images/7539947	138		110 en	1.47238E+12	7.69857E+17	["2395608518"]
1852404296	FreeFX2	["galaxy note 7"]	http://pbs.twimg.com/profile_images/3788000	193		169 en	1.47281E+12	7.71637E+17	["17082836"]
320733081	RichLantos	["#nationaldogday"]	http://pbs.twimg.com/profile_images/6008477	243		122 en	1.47227E+12	7.69309E+17	["634734888"]
611525786	lonestomestereo	["#backtohogwarts"]	http://pbs.twimg.com/profile_images/7435915	384		277 en	1.47274E+12	7.71337E+17	["30892730"]
451819350	topnotchvlt	["#backtohogwarts"]	http://pbs.twimg.com/profile_images/7480120	1046		1059 en	1.47274E+12	7.71336E+17	["2835941210"]
465857810	m_e_lees	["#backtohogwarts"]	http://pbs.twimg.com/profile_images/6804537	111		144 en	1.47274E+12	7.71336E+17	["2218719389"]
1969604874	remi_conway	["#backtohogwarts"]	http://pbs.twimg.com/profile_images/5342771	228		112 en	1.47274E+12	7.71336E+17	["283011482"]

b. Sesudah

id	screenName	followersCount	friendsCount	tags
1974006392	IanTudor	282	101	["#narcos"]
1975455572	TanLibertyFund	296	101	["#cape canaveral"]
2156447599	iGet_Thinspired	553	101	["#emma stone"]
7.29717E+17	bawngtanbois	601	101	["#isac2016"]
3515703093	NotAubri	243	101	["#youtubeisoverparty"]
4885520930	marcela_gironb	194	101	["#louisweloveyou"]
3304938358	KS_Kaybee	132	101	["#nationaldogday"]
7.28669E+17	Tomatotumtum	217	101	["#nationaldogday"]
4911649793	QuippingWithWit	509	101	["#nationaldogday"]
341844807	colorless	3795	101	["#respecttylerjoseph"]
2565806422	adorbsluke	13223	101	["#respecttylerjoseph"]
3851406972	circusbuterah	145	101	["#gloryoutnow"]
283455112	CaptSlyfox	301	101	["#nationaldogday"]
2202164149	alexa4112	113	101	["#nationaldogday"]
2945767804	zaynsbruhh	655	101	["#respecttylerjoseph"]
1512675240	Pablo_faps	124	101	["#nationaldogday"]
3315691328	shmlssmonaghan	1138	101	["#respecttylerjoseph"]
4704835560	chaaaaaants	122	101	["#nationaldogday"]
2228800634	ahgotseven	1114	101	["#respecttylerjoseph"]
7.34828E+17	anaiesss	139	101	["#nationaldogday"]

7. Codingan menggunakan bahasa python

```
class User:
    def __init__(self, user_id, followers, following,
last_post_tag):
        self.user_id = user_id
        self.followers = followers
        self.following = following
        self.last_post_tag = last_post_tag
```

```

def calculate_mean_following(users):
    total_following = sum(user.following for user in users)
    mean_following = total_following / len(users)
    return mean_following

def count_unique_tags(tag_list):
    tag_count = {}
    for tag in tag_list:
        tag_count[tag] = tag_count.get(tag, 0) + 1
    return tag_count

def filter_users(users, mean_following):
    filtered_users = []
    for user in users:
        if ((user.following >= mean_following and user.followers /
user.following >= 11) or
            (user.following < mean_following and user.followers -
user.following >= 10000)):
            filtered_users.append(user)
    return filtered_users

# Contoh data pengguna Twitter dengan atribut tag
users_with_tags = [
    User(1, 2000, 100, "tech"),
    User(2, 50000, 3000, "food"),
    User(3, 15000, 2000, "fashion"),
    User(4, 3000, 250, "travel"),
    User(5, 25000, 1800, "music"),
    User(6, 8000, 600, "tech"),
    User(7, 12000, 900, "food"),
    User(8, 2500, 200, "fashion"),
    User(9, 3500, 300, "travel"),
    User(10, 18000, 1200, "music"),
    User(11, 300, 30, "tech"),
    User(12, 600, 40, "food"),
    User(13, 7500, 500, "fashion"),
    User(14, 20000, 1500, "travel"),
    User(15, 4500, 400, "music"),
    User(16, 500, 50, "tech"),

```



```

    User(17, 1200, 80, "food"),
    User(18, 800, 70, "fashion"),
    User(19, 7000, 600, "travel"),
    User(20, 15000, 1000, "music"),
    User(21, 1000, 100, "tech"),
    User(22, 200, 20, "food"),
    User(23, 8500, 700, "fashion"),
    User(24, 1000, 90, "travel"),
    User(25, 3000, 200, "music"),
    User(26, 700, 60, "tech"),
    User(27, 1500, 100, "food"),
    User(28, 10000, 800, "fashion"),
    User(29, 2000, 150, "travel"),
    User(30, 18000, 1300, "music"),
    User(31, 250, 25, "tech"),
    User(32, 700, 50, "food"),
    User(33, 9000, 700, "fashion"),
    User(34, 1200, 100, "travel"),
    User(35, 4000, 300, "music"),
    User(36, 800, 70, "tech"),
    User(37, 1800, 120, "food"),
    User(38, 6000, 500, "fashion"),
    User(39, 8000, 600, "travel"),
    User(40, 16000, 1100, "music"),
    User(41, 300, 30, "tech"),
    User(42, 600, 40, "food"),
    User(43, 7500, 500, "fashion"),
    User(44, 20000, 1500, "travel"),
    User(45, 4500, 400, "music"),
    User(46, 500, 50, "tech"),
    User(47, 1200, 80, "food"),
    User(48, 800, 70, "fashion"),
    User(49, 7000, 600, "travel"),
    User(50, 15000, 1000, "music")
]

# Menghitung mean total following
mean_total_following = calculate_mean_following(users_with_tags)

# Memfilter pengguna yang sesuai dengan kriteria

```

```

filtered_users = filter_users(users_with_tags, mean_total_following)

# Menyimpan tag dari pengguna yang sesuai kriteria
tag_list = [user.last_post_tag for user in filtered_users]

# Menghitung jumlah tag yang unik
tag_count = count_unique_tags(tag_list)

# Menampilkan counting setiap tag unik yang ada, sort by jumlah
terbanyak
sorted_tag_count = {k: v for k, v in sorted(tag_count.items(),
key=lambda item: item[1], reverse=True)}

for tag, count in sorted_tag_count.items():
    print(f"{tag}: {count}")

```

Hasil code jika di run

```

D:\SEM4\bigdata>python uts-bigdata-code.py
music: 6
travel: 5
fashion: 3
food: 2
tech: 1

```

8. Codingan berdasarkan dataset menggunakan jupyter notebook

Python

```

import pandas as pd
import matplotlib.pyplot as plt

class User:
    def __init__(self, user_id, followers, following,
last_post_tag):
        self.user_id = user_id
        self.followers = followers
        self.following = following
        self.last_post_tag = last_post_tag

```

```

def calculate_mean_following(users):
    total_following = sum(user.following for user in users)
    mean_following = total_following / len(users)
    return mean_following

def count_unique_tags(tag_list):
    tag_count = {}
    for tag in tag_list:
        tag_count[tag] = tag_count.get(tag, 0) + 1
    return tag_count

def plot_users_count(filtered_users_influ, filtered_users_biasa):
    # Hitung jumlah pengguna di masing-masing kelompok
    total_influ_users = len(filtered_users_influ)
    total_biasa_users = len(filtered_users_biasa)

    # Label untuk sumbu x dan y
    labels = ['Influencer', 'Biasa']
    counts = [total_influ_users, total_biasa_users]

    # Buat diagram batang
    plt.bar(labels, counts, color=['blue', 'green'])

    # Tambahkan judul dan label sumbu
    plt.title('Jumlah Pengguna Twitter Influencer dan Biasa')
    plt.xlabel('Kelompok')
    plt.ylabel('Jumlah Pengguna')

    # Tampilkan plot
    plt.show()

def filter_users(users, mean_following):
    filtered_users_influ = []
    filtered_users_biasa = []
    for user in users:
        if ((user.following >= mean_following and user.followers /
user.following >= 11) or
            (user.following < mean_following and user.followers -
user.following >= 10000)):

```

```

        filtered_users_influ.append(user)
    else:
        filtered_users_biasa.append(user)

    # Menghitung jumlah pengguna di masing-masing kelompok
    total_influ_users = len(filtered_users_influ)
    total_biasa_users = len(filtered_users_biasa)
    plot_users_count(filtered_users_influ, filtered_users_biasa)

    # Membandingkan jumlah terbanyak
    if total_influ_users > total_biasa_users:
        print( "Kelompok influencer memiliki jumlah pengguna
        terbanyak dengan total {} pengguna.".format(total_influ_users))
        return filtered_users_influ
    elif total_influ_users < total_biasa_users:
        print ( "Kelompok biasa memiliki jumlah pengguna terbanyak
        dengan total {} pengguna.".format(total_biasa_users))
        return filtered_users_biasa
    else:
        print ( "Jumlah pengguna di kedua kelompok sama, dengan
        total masing-masing {} pengguna.".format(total_influ_users))
        return filtered_users_biasa, filtered_users_influ

df = pd.read_excel("data.xlsx")
df.head(5)

```

Membuat objek User dari DataFrame

```
In [3]: df = pd.read_excel("data.xlsx")
```

```
In [4]: df.head(5)
```

Out[4]:

	id	screenName	followersCount	friendsCount	tags
0	1974006392	IanTudor	282	101	["#narcos"]
1	1975455572	TanLibertyFund	296	101	["cape canaveral"]
2	2156447599	iGet_Thinspired	553	101	["emma stone"]
3	729716562056248960	bawngtanbois	601	101	["#isac2016"]
4	3515703093	NotAubri	243	101	["#youtubeisoverparty"]

```

users_with_tags = []
for index, row in df.iterrows():

```

```

        user = User(row['id'], row['followersCount'],
row['friendsCount'], row['tags'])
        users_with_tags.append(user)

# Menghitung mean total following
mean_total_following = calculate_mean_following(users_with_tags)

# Memfilter pengguna yang sesuai dengan kriteria
filtered_users = filter_users(users_with_tags,
mean_total_following)

# Menyimpan tag dari pengguna yang sesuai kriteria
tag_list = [user.last_post_tag for user in filtered_users]

# Menghitung jumlah tag yang unik
tag_count = count_unique_tags(tag_list)

```

9. Codingan berdasarkan dataset menggunakan Apache Spark

Di bawah ini merupakan code untuk mengkategorikan user twitter apakah seorang influencer atau bukan di Apache Spark dengan menggunakan library Spark SQL.

Code

```

Python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf, mean
from pyspark.sql.types import StringType

# Inisialisasi sesi Spark
spark = SparkSession.builder.appName("TwitterUserAnalysis") .getOrCreate()

# Membaca data dari file Excel ke dalam DataFrame Spark
df = spark.read.format("csv") .option("header", True) .option("separator",
True) .load("hdfs:///BigData/UAS/uas.csv")

```

```

# Kelas User
class User:
    def __init__(self, user_id, followers, following, last_post_tag):
        self.user_id = user_id
        self.followers = followers
        self.following = following
        self.last_post_tag = last_post_tag

# Fungsi untuk menghitung rata-rata following
def calculate_mean_following(df):
    mean_following = df.select(mean(col("friendsCount"))).first()[0]
    return mean_following

# Fungsi untuk mengklasifikasikan pengguna
def classify_user(followers, following, mean_following):
    followers = float(followers)
    following = float(following)
    if ((following >= mean_following and followers / following >= 11) or
        (following < mean_following and followers - following >= 10000)):
        return "influencer"
    else:
        return "biasa"

# Fungsi untuk menghitung jumlah tag unik
def count_unique_tags(tags_df):
    tags_rdd = tags_df.rdd.flatMap(lambda x: x)
    tag_counts = tags_rdd.map(lambda tag: (tag, 1)).reduceByKey(lambda a, b: a
+ b)
    return tag_counts

# Menghitung rata-rata following
mean_following = calculate_mean_following(df)

# Mendefinisikan UDF untuk mengklasifikasikan pengguna
classify_user_udf = udf(lambda followers, following: classify_user(followers,
following, mean_following), StringType())

# Menambahkan kolom klasifikasi pengguna
df_with_classification = df.withColumn(
    "classification",
    classify_user_udf(col("followersCount"), col("friendsCount"))
)

```

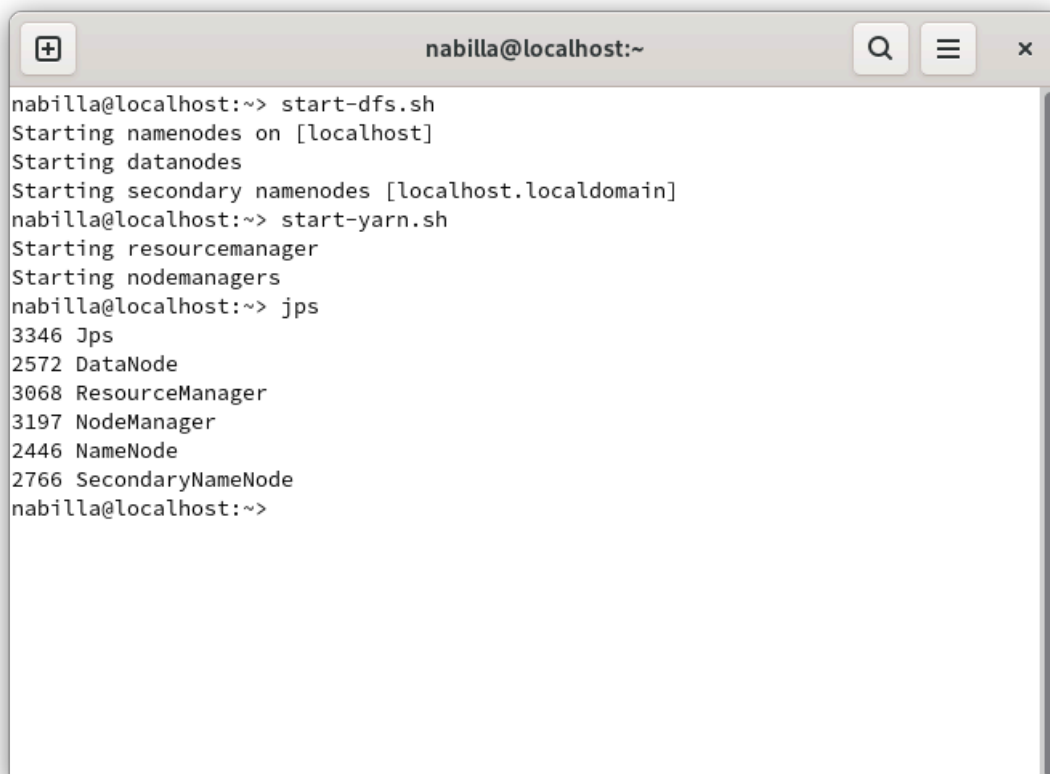
```
# Menampilkan DataFrame dengan klasifikasi
df_with_classification.show()

# WordCount untuk influencer dan biasa
classification_rdd =
df_with_classification.select("classification").rdd.flatMap(lambda x: x)

# Menghitung jumlah influencer dan biasa menggunakan wordcount
classification_counts = classification_rdd.map(lambda classification:
(classification, 1)).reduceByKey(lambda a, b: a + b)
print(classification_counts.collect())
```

Langkah - langkah membuat Analisa menggunakan Apache Spark

1. Mengaktifkan HDFS untuk menjalankan file inputan pada program



```
nabilla@localhost:~> start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [localhost.localdomain]
nabilla@localhost:~> start-yarn.sh
Starting resourcemanager
Starting nodemanagers
nabilla@localhost:~> jps
3346 Jps
2572 DataNode
3068 ResourceManager
3197 NodeManager
2446 NameNode
2766 SecondaryNameNode
nabilla@localhost:~>
```

2. Memasukkan dataset yang telah di pra proses kedalam HDFS

```
nabilla@localhost:~  
nabilla@10:~> hadoop fs -mkdir -p /nabilla/BigData/UAS  
nabilla@10:~> hadoop fs -put '/home/nabilla/Documents/BigData/UAS/uas.csv' /nabilla/BigData/UAS  
nabilla@10:~>
```

3. Membuat Aplikasi Apache Spark

The screenshot displays the Hadoop web interface at localhost:8088/cluster. The interface includes a sidebar with navigation links (About, Nodes, Node Labels, Applications, NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED, Scheduler, Tools) and a main content area with various metrics and application details.

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
1	0	1	0	3

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Application Details

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1717826869736_0001	nabilla	uas	SPARK		root.default	0	Sat Jun 8 13:19:21 +0700 2024	Sat Jun 8 13:19:35 +0700 2024	N/A

Showing 1 to 1 of 1 entries

4. Memasukkan Codingan

[illegible]

```
nabilla@localhost:....1-bin-hadoop3/bin
```

```
>>> classify_user_udf = udf(lambda followers, following: classify_user(followers, following, mean_following), StringType())
>>> df_with_classification = df.withColumn(
...     "classification",
...     classify_user_udf(col("followersCount"), col("friendsCount"))
... )
>>>
>>> df_with_classification.show(5, False)
```

id	screenName	followersCount	friendsCount	tags	classification
1974006392	lanTudor	282	101	"[""#narcos""]"	biasa
1975455572	TanLlbertyFund	296	101	"[""cape canaveral""]"	biasa
2156447599	iGet_Thinspired	553	101	"[""emma stone""]"	biasa
7,29717E+17	bawngtanbois	601	101	"[""#isac2016""]"	biasa
3515703093	NotAubri	243	101	"[""#youtubeisoverparty""]"	biasa

```
only showing top 5 rows

>>> def count_unique_tags(tags_df):
...     tags_rdd = tags_df.rdd.flatMap(lambda x: x)
...     tag_counts = tags_rdd.map(lambda tag: (tag, 1)).reduceByKey(lambda a, b: a + b)
...     return tag_counts
...
>>> classification_rdd = df_with_classification.select("classification").rdd.flatMap(lambda x: x)
>>> classification_counts = classification_rdd.map(lambda classification: (classification, 1)).reduceByKey(lambda a,
b: a + b)
>>> print(classification_counts.collect())
[('biasa', 38659), ('influencer', 599)]
>>>
```

```
nabilla@localhost:....1-bin-hadoop3/bin
>>> class User:
...     def __init__(self, user_id, followers, following, last_post_tag):
...         self.user_id = user_id
...         self.followers = followers
...         self.following = following
...         self.last_post_tag = last_post_tag
...
>>> def calculate_mean_following(df):
...     mean_following = df.select(mean(col("friendsCount"))).first()[0]
...     return mean_following
...
>>> def classify_user(followers, following, mean_following):
...     followers = float(followers)
...     following = float(following)
...     if ((following >= mean_following and followers / following >= 11) or
...         (following < mean_following and followers - following >= 10000)):
...         return "influencer"
...     else:
...         return "biasa"
...
>>> mean_following = calculate_mean_following(df)
>>> print (mean_following)
911.09381527332
>>> classify_user_udf = udf(lambda followers, following: classify_user(followers, following, mean_
following), StringType())
>>> df_with_classification = df.withColumn(
...     "classification",
...     classify_user_udf(col("followersCount"), col("friendsCount"))
... )
>>> _
```

Penjelasan Code

`from pyspark.sql import SparkSession`

Import SparkSession: SparkSession adalah titik masuk utama ke dalam API DataFrame dan SQL di PySpark. Ini digunakan untuk membuat aplikasi Spark.

`from pyspark.sql.functions import col, udf, mean`

col: Digunakan untuk menunjuk ke kolom dalam DataFrame.

udf: Digunakan untuk mendefinisikan fungsi yang dapat diterapkan pada kolom DataFrame. udf memungkinkan kita menulis fungsi Python yang dapat digunakan pada kolom DataFrame.

mean: Digunakan untuk menghitung rata-rata dari nilai kolom tertentu dalam DataFrame.

```
from pyspark.sql.types import StringType
```

StringType: Digunakan untuk menentukan tipe data dari kolom dalam DataFrame. Dalam konteks ini, StringType digunakan saat mendefinisikan UDF untuk menunjukkan bahwa fungsi tersebut mengembalikan tipe data string. (“influ”, “biasa”)

```
# Membaca data dari file Excel ke dalam DataFrame Spark
```

```
df = spark.read.format("csv") .option("header", True) .option("separator", True) .load("hdfs:///BigData/UAS/uas.csv")
```

Membaca data berformat file CSV, yang bersumber dari file HDFS, dan menyimpannya kedalam dataframe.

```
# Kelas User
```

```
class User:
```

```
    def __init__(self, user_id, followers, following, last_post_tag):
        self.user_id = user_id
        self.followers = followers
        self.following = following
        self.last_post_tag = last_post_tag
```

Mendefinisikan kelas User dengan atribut user_id, followers, following, dan last_post_tag.

```
# Fungsi untuk menghitung rata-rata following
```

```
def calculate_mean_following(df):
    mean_following = df.select(mean(col("friendsCount"))).first()[0]
    return mean_following
```

Fungsi diatas digunakan untuk menghitung rata-rata jumlah following dari dataframe yang kemudian mengembalikan nilai rata-rata tersebut.

```
# Fungsi untuk mengklasifikasikan pengguna
```

```
def classify_user(followers, following, mean_following):
    followers = float(followers)
    following = float(following)
    if ((following >= mean_following and followers / following >= 11) or
        (following < mean_following and followers - following >= 10000)):
        return "influencer"
    else:
        return "biasa"
```

Fungsi ini digunakan untuk mengklasifikasikan pengguna menjadi “influencer” atau “masyarakat biasa” berdasarkan syarat yang telah dibuat.

- Jika following lebih besar atau sama dengan rata-rata following (mean_following) dan rasio followers ke following lebih besar atau sama dengan 11, atau
- Jika following kurang dari rata-rata following (mean_following) dan selisih followers dan following lebih besar atau sama dengan 10.000.

Mengembalikan "influencer" jika salah satu kondisi terpenuhi, dan "biasa" jika tidak.

```
# Fungsi untuk menghitung jumlah tag unik
def count_unique_tags(tags_df):
    tags_rdd = tags_df.rdd.flatMap(lambda x: x)
    tag_counts = tags_rdd.map(lambda tag: (tag, 1)).reduceByKey(lambda a, b: a
+ b)
    return tag_counts
```

Fungsi ini digunakan untuk menghitung jumlah tags yang digunakan menggunakan map reduce.

```
# Menghitung rata-rata following
mean_following = calculate_mean_following(df)
```

Menghitung rata-rata following menggunakan fungsi calculate_mean_following dan menyimpannya dalam variabel mean_following.

```
# Mendefinisikan UDF untuk mengklasifikasikan pengguna
classify_user_udf = udf(lambda followers, following: classify_user(followers,
following, mean_following), StringType())
Mendefinisikan User Defined Function (UDF) classify_user_udf untuk
mengklasifikasikan pengguna menggunakan fungsi classify_user.
```

```
# Menambahkan kolom klasifikasi pengguna
df_with_classification = df.withColumn(
    "classification",
    classify_user_udf(col("followersCount"), col("friendsCount"))
)
```

Menambahkan kolom baru classification ke DataFrame df dengan menerapkan UDF classify_user_udf pada kolom followersCount dan friendsCount.

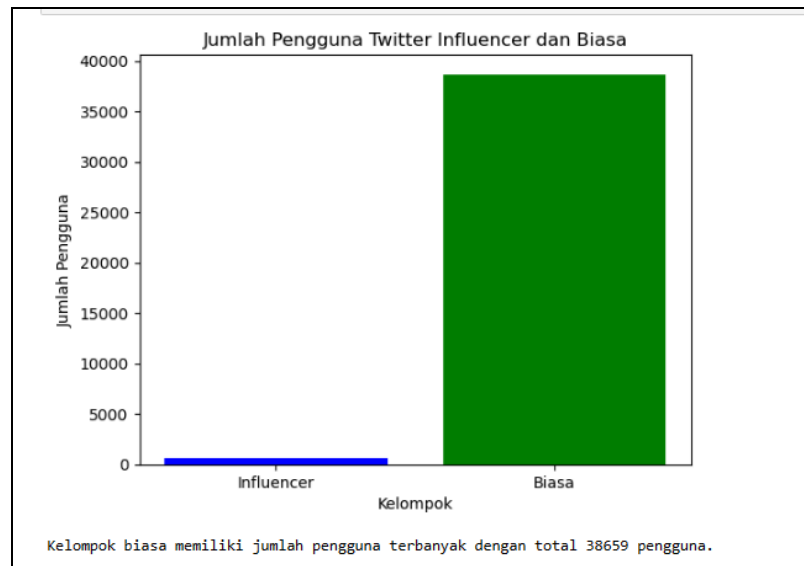
```
# WordCount untuk influencer dan biasa
classification_rdd =
df_with_classification.select("classification").rdd.flatMap(lambda x: x)
```

Mengambil kolom classification dari df_with_classification dan mengubahnya menjadi RDD.

```
# Menghitung jumlah influencer dan biasa menggunakan wordcount
classification_counts = classification_rdd.map(lambda classification:
(classification, 1)).reduceByKey(lambda a, b: a + b)
print(classification_counts.collect())
```

Menghitung jumlah pengguna yang diklasifikasikan sebagai "influencer" dan "biasa" menggunakan operasi map dan reduceByKey.

10. Hasil Analisis Menggunakan Jupyter Notebook



Hasil analisis menunjukkan bahwa jumlah masyarakat biasa yang menggunakan X/Twitter secara signifikan lebih tinggi daripada jumlah influencer. Temuan ini mengindikasikan bahwa Twitter telah menjadi platform yang luas digunakan oleh masyarakat umum untuk berbagi pemikiran, informasi, dan interaksi sosial, dengan dampak yang mungkin mencakup ragamnya opini publik dan potensi pengaruh terhadap tren dan peristiwa yang sedang berlangsung.

Selain itu, temuan bahwa jumlah masyarakat biasa yang menggunakan X/Twitter secara signifikan lebih tinggi daripada jumlah influencer mengindikasikan adanya perbedaan yang menarik dalam preferensi pengguna di platform tersebut. Hal ini mungkin menunjukkan bahwa X/Twitter memiliki daya tarik yang lebih luas di kalangan masyarakat umum, dengan penggunaan yang mungkin lebih beragam dan inklusif. Potensi penyebabnya bisa mencakup tujuan penggunaan yang berbeda, dengan masyarakat biasa menggunakan platform tersebut untuk interaksi sosial, mendapatkan berita, atau hanya untuk bersenang-senang, sementara influencer mungkin lebih fokus pada membangun merek pribadi atau mengiklankan produk.

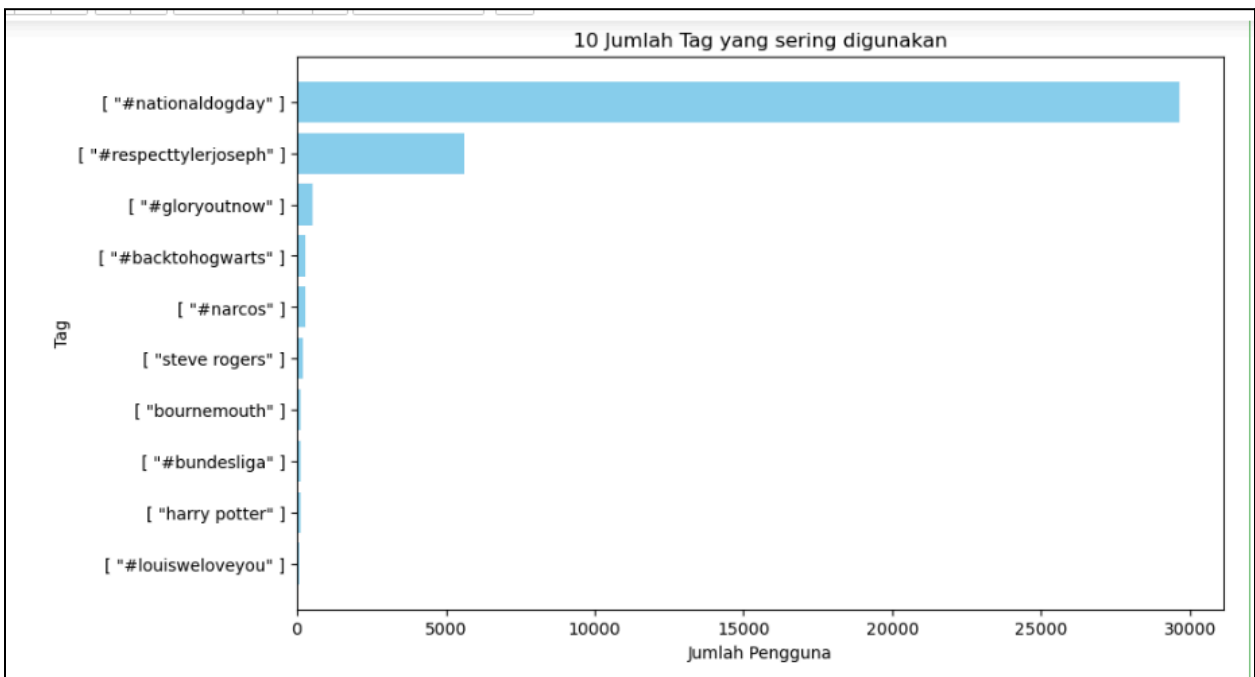
Python

```
# Mengambil 10 tags yang sering digunakan
top_10_tags = dict(list(sorted_tag_count.items())[:10])

labels = list(top_10_tags.keys())
counts = list(top_10_tags.values())

# Buat plot
```

```
plt.figure(figsize=(10, 6)) # Atur ukuran plot
plt.barh(labels, counts, color='skyblue')
plt.xlabel('Jumlah Pengguna')
plt.ylabel('Tag')
plt.title('10 Jumlah Tag yang sering digunakan')
plt.gca().invert_yaxis() # Inversi sumbu y agar tag paling banyak muncul di atas
plt.show()
```



```
In [12]: for tag, count in sorted_tag_count.items():
          print(f"{tag}: {count}")
```

```
["#nationaldogday"] : 29646
["#respecttylerjoseph"] : 5615
["#gloryoutnow"] : 540
["#backtohogwarts"] : 277
["#narcos"] : 260
["steve rogers"] : 188
["bournemouth"] : 114
["#bundesliga"] : 112
["harry potter"] : 100
["#louisweloveyou"] : 95
["#whatsapp"] : 85
["#isac2016"] : 83
["labour"] : 81
["#getwellsoonjackson"] : 77
["#belgiangp"] : 77
["#weekend"] : 73
["#journalssstansparty"] : 68
["halsey"] : 66
["#happybirthdaydylanobrien"] : 57
```

Analisis kami menemukan bahwa pengguna Twitter cenderung menggunakan tagar tertentu. Tagar yang paling sering muncul adalah #nationaldogday, diikuti oleh #respecttylerjoseph dan #gloryoutnow. Temuan ini mengindikasikan minat dan tren yang dominan di kalangan pengguna Twitter yang kami amati.

11. Hasil Analisis menggunakan Apache Spark

Hasil analisis menunjukkan bahwa jumlah masyarakat biasa yang menggunakan X/Twitter secara signifikan lebih tinggi daripada jumlah influencer. Seorang pengguna dikategorikan sebagai influencer jika memenuhi salah satu dari dua kondisi utama. Pertama, jika jumlah following seorang pengguna lebih dari atau sama dengan rata-rata (mean) following, maka jumlah followers pengguna harus lebih dari atau sama dengan 11 kali lipat dari jumlah following, atau selisih antara jumlah followers dan following harus lebih dari atau sama dengan 10.000. Kedua, jika jumlah following seorang pengguna kurang dari rata-rata (mean) following, maka selisih antara jumlah followers dan following harus lebih dari atau sama dengan 10.000. Pengguna yang tidak memenuhi salah satu dari kondisi tersebut dikategorikan sebagai masyarakat

12. Kesimpulan

Twitter adalah platform yang banyak digunakan oleh masyarakat umum untuk berbagai keperluan, seperti berbagi pemikiran, informasi, dan berinteraksi sosial. Hal ini menunjukkan bahwa Twitter memiliki pengaruh yang besar dalam membentuk beragam opini publik dan mempengaruhi tren serta peristiwa yang sedang berlangsung. Selain itu, perbedaan yang signifikan antara jumlah pengguna biasa dan influencer menunjukkan bahwa Twitter menarik bagi beragam kalangan, dengan penggunaan yang lebih inklusif. Ini bisa disebabkan oleh perbedaan dalam tujuan penggunaan, di mana pengguna biasa lebih fokus pada interaksi sosial dan mendapatkan informasi, sementara influencer lebih berorientasi pada membangun merek atau mempromosikan produk. Selain itu, analisis tagar yang dominan seperti #nationaldogday dan lainnya menunjukkan minat dan tren yang populer di kalangan pengguna Twitter yang diteliti.