BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Pada bab ini penelitian yang berfokus pada analisis perspektif masyarakat tentang Daerah Istimewa Yogyakarta dengan menggunakan *Algoritma K-means*. Penelitian ini dilakukan dengan mengambil data dari *Twitter* dengan topik "Yogyakarta". Berdasarkan pengumpulan data dengan metode *crawling* data dari 1 Mei sampai 16 Juni dengan maksimal 3000 data *tweet*, akan tetapi hasil yang diperoleh jumlah keseluruhan 3006 data *tweet* sudah tercapai dari 11 juni sampai 16 juni 2023. Setelah dilakukan *crawling* data selanjutnya yaitu proses *preprocessing* untuk membersikan data, TF-IDF untuk menghitung bobot kata-kata, dan menerapkan *Algoritma K-Means* mengelompokkan data berdasarkan kemiripan data. Berikut penerapan analisis perspektif mayarakat tentang Daerah Yogyakarta dengan menggunakan *Algoritma K-means* yaitu :

4.2 HASIL CRAWLING DATA

Dengan menggunakan metode crawling dan bantuan *library* snscrape, memudahkan pengambilan data dari Twitter. Data yang diambil merupakan tweet dengan kata kunci "Yogyakarta". Berikut hasil *crawling* data dari Twitter dengan kata kunci "Yogyakarta" :

Tabel 4. 1 Hasil *Crawling* Data

No	Text
1.	Polsek rayon 3 (Gondokusuman, Pakualaman dan Danurejan) melaksanakan Apel Kegiatan Rutin Yang Ditingkatkan antisipasi gangguan Kamtibmas di wilayah Kota Yogyakarta, Sabtu (17/6). #poldadiy #polrestayogyakarta #humaspolsekgk #polsekgondokusuman #kapoldadiy #kapolrestayogyakarta https://t.co/BBPQ8IZKft
2.	20 Destinasi Liburan Terbaik di Yogyakarta Yang Wajib Dikunjungi! - https://t.co/y8etfis6fD
3.	Gubernur Daerah Istimewa Yogyakarta (DIY), Sri Sultan Hamengku Buwono X, menyatakan masyarakat harus bersiap mengahadapi masa

	pandemi ke endemi covid-19. Pemerintah tak lagi membiayai ongkos berobat di fasilitas kesehatan. #GubernurYogyakarta https://t.co/yBOBYl9mMA
4.	Disdikpora Daerah Istimewa Yogyakarta (DIY) sedang menimbang-nimbang untuk memberikan beasiswa kepada siswi berprestasi Putri Ariani. https://t.co/DdxzCAshLs
5.	Jum'at 16/06/2023 Pukul 22.15 Wib. Untuk mengantisipasi maraknya aksi tindak kejahatan menggelar patroli terutama dimalam hari. Unit Patroli malam Personil samapta Polsek Pakualaman Polresta Yogyakarta melaksanakan patroli . https://t.co/JbpMuNPHPA
6.	Kota pilihan untuk menempuh pendidikan yaitu Kota Yogyakarta diikuti oleh Bandung, Malang, Jakarta, Solo, Surabaya, Makassar, Semarang, Medan, dan kota lainnya. #goodstats #gnfi https://t.co/dyjE5CYUaU
7.	Wisata Pantai Widodaren di Gunung Kidul, Yogyakarta, telah menjadi daya tarik utama bagi para pengunjung yang mencari keindahan alam yang menakjubkan dan pesona pantai yang memesona. https://t.co/ZEkgcTMRVj
8.	Dinas Pariwisata Kota Batu kembali menjadi tempat lokus dan studi lapangan Pelatihan Kepemimpinan Pengawas (PKP) Angkatan II Pemerintah Daerah Istimewa Yogyakarta, pada Jum'at,(16/6/2023), bertempat di Kantor Dinas Pariwisata Kota Batu, Balaikota Among Tani Kota Batu. https://t.co/5R7e7s3vqm
9.	Dulu Dicap Miskin, Kampung Unik di Yogyakarta Ini Disulap Jadi Kebun Buah dan Sayur, Punya Singkatan Kece - Malang Network https://t.co/YSVIDIGvPb
10.	Upaya pencegahan tindak kriminalitas dengan sasaran tempat rawan gangguan kamtibmas lainnya Anggota patroli gabungan polresta yogyakarta melaksanakan patroli, tepatnya di wilayah hukum Polresta yogyakarta. Jumat (16/06/2023). #patrolijogja https://t.co/hEAaJuujNm

4.3 HASIL PREPROCESSING DATA

Setelah mendapatkan data dari Twitter berupa tweet dengan kata kunci "Yogyakarta" selanjutnya proses *preprocessing*, yang dimana tahap ini digunakan untuk membersikan data. Dalam proses *preprocessing* ini terdapat beberapa tahapan yaitu sebagai berikut :

4.3.1 CaseFolding

Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses *casefolding* :

```
def case_folding(text):
    lower_word =  text.lower()
    return lower_word
df['case_folding']= df['text'].str.lower()

print(df['case_folding'])
```

```
0            solo yogyakarta, lo lo lo ga bahaya taaaðÿ¤ª
1       polsek rayon 3 (gondokusuman, pakualaman dan d...
2                           @idextratime ke yogyakarta???
3       kak, rute pasar senen - yogyakarta kelas senja...
4       simak prakiraan cuaca di yogyakarta hari ini y...
                              ...
3001    @widhaswarawidya istimewa yogyakarta jakarta h...
3002    ðÿ«  universitas negeri yogyakarta\nhttps://t....
3003    cara menghitung nilai gabungan ppdb sma smk di...
3004    missing jogja hourâ€¦ ðÿ¥¹\n#yogyakarta https:...
3005    hallo semua nyaâœ¨\nready promo untuk hari ini...
Name: case_folding, Length: 3006, dtype: object
```

Gambar 4. 1 Kode Program dan Hasil Casefolding

Dilihat dari gambar 4. 1 dengan menggunakan perintah casefolding yang menerima 1 parameter ‘text’,berfungsi untuk mengubah semua karakter dalam teks menjadi huruf kecil (*lowercase*).

4.3.2 Cleaning Data

Berdasarkan gambar 4. 2 ada perintah cleaning yaitu untuk membersihkan kata yang tidak penting seperti simbol, angka, *re-tweet*, dll. Di dalam cleaning data terdapat remove_tweet_special yang dimana ada casefolding.replace bertujuan untuk mengganti karakter ‘\t’, ‘\n’, ‘\’ menjadi spasi kosong, text.encode bertujuan untuk mengubah teks menjadi format ASCII dan menggantikan karakter yang tidak terdefinisi dalam ASCII menjadi karakter tanda tanya, dan text.replace untuk menganti “http://” dan “https:// “ menjadi spasi kosong. Remove_number untuk menghapus angka, remove_puctuation untuk menghapus karakter-karakter spesial dan simbol. Remove_whitespace yaitu untuk mengonversi teks menjadi string, remove_singl_char bertujuan untuk menghapus karakter tunggal dalam teks. Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses cleaning data :

```python
def remove_tweet_special(case_folding):
    if isinstance(case_folding, float):
        return ""
    else:
        text = case_folding.replace('\\t'," ").replace('\\n'," ").replace('\\',"")
        text = text.encode('ascii', 'replace').decode('ascii')
        text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
        return text.replace("http://", " ").replace("https://", " ")
df['clean'] = df['case_folding'].apply(remove_tweet_special)

def remove_number(clean):
    text = re.sub(r"\d+", "", clean)
    return text
df['clean'] = df['clean'].apply(remove_number)

def remove_punctuation(clean):
    clean_spcl = re.compile('[/(){}\[\]\|@,;]')
    clean_symbol = re.compile('[^0-9a-z]')
    text = clean_spcl.sub('', clean)
    text = clean_symbol.sub(' ', clean)
    return text
df['clean'] = df['clean'].apply(remove_punctuation)

def remove_whitespace(clean):
    corrected = str(clean)
    corrected = re.sub(r"//t",r"\t", corrected)
    corrected = re.sub(r"( )\1+",r"\1", corrected)
    corrected = re.sub(r"(\n)\1+",r"\1", corrected)
    corrected = re.sub(r"(\r)\1+",r"\1", corrected)
    corrected = re.sub(r"(\t)\1+",r"\1", corrected)
    return corrected.strip(" ")
df['clean'] = df['clean'].apply(remove_whitespace)

def remove_singl_char(clean):
    singl = re.sub(r"\b[a-zA-Z]\b", "",clean)
    return singl
df['clean'] = df['clean'].apply(remove_singl_char)

def remove_repeated_letters(clean):
    # Mencocokkan pola kata-kata dengan huruf yang diulang minimal 3 kali
    pattern = r'\b\w*(\w)\1{2,}\w*\b'
    processed_text = re.sub(pattern, '', clean)
    return processed_text
df['filtering'] = df['clean'].apply(remove_repeated_letters)

print(df['filtering'])
```

Gambar 4. 2 Kode Program Cleaning

```
0                          solo yogyakarta lo lo lo ga bahaya
1       polsek rayon gondokusuman pakualaman dan danur...
2                                                 ke yogyakarta
3       kak rute pasar senen yogyakarta kelas senja ut...
4       simak prakiraan cuaca di yogyakarta hari ini y...
                                ...
3001                    istimewa yogyakarta jakarta hehehe
3002                         universitas negeri yogyakarta
3003    cara menghitung nilai gabungan ppdb sma smk di...
3004                                    missing jogja hour
3005    hallo semua nya ready promo untuk hari ini slo...
Name: filtering, Length: 3006, dtype: object
```

Gambar 4. 3 Hasil Cleaning

4.3.3 Tokenizing

Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses *tokenizing* data :

```
from nltk.tokenize import word_tokenize

def tokenizing(filtering):
    tokens = word_tokenize(filtering)
    return tokens

df['token'] = df['filtering'].apply(tokenizing)

print(df['token'])
```

```
0              [solo, yogyakarta, lo, lo, lo, ga, bahaya]
1       [polsek, rayon, gondokusuman, pakualaman, dan,...
2                                          [ke, yogyakarta]
3       [kak, rute, pasar, senen, yogyakarta, kelas, s...
4       [simak, prakiraan, cuaca, di, yogyakarta, hari...
                              ...
3001             [istimewa, yogyakarta, jakarta, hehehe]
3002                   [universitas, negeri, yogyakarta]
3003    [cara, menghitung, nilai, gabungan, ppdb, sma,...
3004                                [missing, jogja, hour]
3005    [hallo, semua, nya, ready, promo, untuk, hari,...
Name: token, Length: 3006, dtype: object
```

Gambar 4. 4 Kode Program dan Hasil *Tokenizing*

Berdasarkan gambar 4.4 *tokenizing* berfungsi untuk memisahkan kata-kata dalam teks.

4.3.4 Stopword

Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses *stopword* data :

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory

factory = StopWordRemoverFactory()
stopword = factory.create_stop_word_remover()

def stopword_removal(tokens):
    if isinstance(tokens, str):
        stopwords = stopword.remove(tokens)
    else:
        stopwords = [stopword.remove(token) for token in tokens]
    return stopwords
df['stopword'] = df['token'].apply(stopword_removal)

print(df['stopword'])
```

```
0              [solo, yogyakarta, lo, lo, lo, ga, bahaya]
1       [polsek, rayon, gondokusuman, pakualaman, , da...
2                                          [, yogyakarta]
3       [kak, rute, pasar, senen, yogyakarta, kelas, s...
4       [simak, prakiraan, cuaca, , yogyakarta, hari, ...
                              ...
3001             [istimewa, yogyakarta, jakarta, hehehe]
3002                   [universitas, negeri, yogyakarta]
3003    [cara, menghitung, nilai, gabungan, ppdb, sma,...
3004                                [missing, jogja, hour]
3005    [hallo, semua, nya, ready, promo, , hari, , sl...
Name: stopword, Length: 3006, dtype: object
```

Gambar 4. 5 Kode Program dan Hasil *Stopword*

Pada gambar 4. 5 *stopword* ini digunakan untuk membersihkan kata-kata umum yang sering muncul dalam teks seperti yang, dengan, dan, pada, dan sebagainya.

4.3.5 Stemming

Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses *stemming* data :

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def stemming(stopwords):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    return stemmer.stem(stopwords)
df['stemmer'] = df['stopword'].apply(lambda x: [stemming(token) for token in x])

print(df['stemmer'])
```

```
0              [solo, yogyakarta, lo, lo, lo, ga, bahaya]
1       [polsek, rayon, gondokusuman, pakualaman, , da...
2                                              [, yogyakarta]
3       [kak, rute, pasar, senen, yogyakarta, kelas, s...
4       [simak, prakira, cuaca, , yogyakarta, hari, , ...
                              ...
3001              [istimewa, yogyakarta, jakarta, hehehe]
3002                    [universitas, negeri, yogyakarta]
3003    [cara, hitung, nilai, gabung, ppdb, sma, smk, ...
3004                                [missing, jogja, hour]
3005    [hallo, semua, nya, ready, promo, , hari, , sl...
Name: stemmer, Length: 3006, dtype: object
```

Gambar 4. 6 Kode Program dan Hasil *Stemming*

Dilihat dari gambar 4. 6 *stemming* bertujuan untuk membuang kaya yang berimbuhan menjadi bentuk dasar contohnya seperti mengitung menjadi hitung, gabungan menjadi gabung, dan sebagainya.

4.3.6 Normalization

Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses *normalization* data untuk mengubah kata yang sebelumnya singkatan dan kata tidak baku menjadi kata baku yaitu seperti berikut :

```python
from nltk.stem import WordNetLemmatizer

norm_dict = {
    'ga' : 'tidak',
    'gk' : 'tidak',
    'gak' : 'tidak',
    'gw' : 'saya',
    'gua' : 'saya',
    'gue' : 'saya',
    'ane' : 'saya',
    'bgt' : 'banget',
    'yg' : 'yang',
    'utk' : 'untuk',
    'trs' : 'terus',
    'bgm' : 'bagaimana',
    'dr' : 'dari',
    'udh' : 'sudah',
    'sdh' : 'sudah',
    'dah' : 'sudah',
    'tp' : 'tapi',
    'td' : 'tadi',
    'pgn' : 'pengen',
    'jd' : 'jadi',
    'ttg' : 'tentang',
    'dikit' : 'sedikit',
    'dpt' : 'mendapat',
    'dgn' : 'dengan',
    'kl' : 'kalau',
    'klo' : 'kalau',
    'otak' : 'pikiran',
    'spt' : 'seperti',
    'jg' : 'juga',
    'mo' : 'mau',
    'ato' : 'atau',
    'janlup' : 'jangan lupa',
    'naro' : 'menaruh',
    'kalo' : 'kalau',
    'naro' : 'menaruh',
    'sabi' : 'bisa'
}

def normalizing(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    normalized_tokens = [norm_dict.get(token, token) for token in lemmatized_tokens]
    normalized_text = " ".join(normalized_tokens)
    return normalized_text


df['normalizing'] = df['stemmer'].apply(normalizing)

print(df['normalizing'])
```

Gambar 4. 7 Kode Program *Normalization*

```
0                    solo yogyakarta lo lo lo tidak bahaya
1       polsek rayon gondokusuman pakualaman  danureja...
2                                            yogyakarta
3       kak rute pasar senen yogyakarta kelas senja ut...
4       simak prakira cuaca  yogyakarta hari   rilis  ...
                              ...
3001                  istimewa yogyakarta jakarta hehehe
3002                       universitas negeri yogyakarta
3003    cara hitung nilai gabung ppdb sma smk diy guna...
3004                                  missing jogja hour
3005    hallo semua nya ready promo  hari  slot bata b...
Name: normalizing, Length: 3006, dtype: object
```

Gambar 4. 8 Hasil *Normalization*

4.4 FITUR EKSTRAKSI

4.4.1 *Term Frequency* (TF)

TF adalah metode perhitungan yang digunakan untuk mengukur seberapa sering sebuat *term* (*t*) muncul dalam suatu dokumen. Dapat dilihat dari gambar 4.9 merupakan program perhitungan tf :

```
dt_fiturs['words'] = dt_fiturs['normalizing']
tf1 = (dt_fiturs['words'][1:2]).apply(lambda x:pd.value_counts(x.split(" "))).sum(axis = 0).reset_index()
tf1.columns = ['words','tf']
tf1
```

	words	tf
0		3
1	polsek	1
2	tingkat	1
3	yogyakarta	1
4	kota	1
5	wilayah	1
6	kamtibmas	1
7	ganggu	1
8	antisipasi	1
9	rutin	1
10	rayon	1
11	giat	1
12	apel	1
13	laksana	1
14	danurejan	1
15	pakualaman	1
16	gondokusuman	1
17	sabtu	1

Gambar 4. 9 Kode Program dan Hasil TF

4.4.2 *Inverse Document Frequency* (IDF)

Hasil perhituga nilai IDF akan mengindikasikan hubungan antara suatu term dengan kumpulan dokumen secara keseluruhan. Jika jumlah term yang terkait dengan suatu dokumen lebih sedikit, maka nilai IDF semakin besar. Dilihat dari gambar 4.10 merupakan kode program perhitungan IDF :

```
for i, word in enumerate(tf1['words']):
    matches = dt_fiturs[dt_fiturs['words'].str.contains(word, na=False)]
    if len(matches) != 0:
        tf1.loc[i, 'idf'] = np.log(dt_fiturs.shape[0] / len(matches))
    else:
        tf1.loc[i, 'idf'] = 0

tf1.columns = ['words', 'tf', 'idf']
tf1
```

	words	tf	idf
0		3	0.002331
1	polsek	1	2.628468
2	tingkat	1	3.833978
3	yogyakarta	1	0.269006
4	kota	1	2.089472
5	wilayah	1	3.045521
6	kamtibmas	1	3.017933
7	ganggu	1	4.224176
8	antisipasi	1	3.947923
9	rutin	1	4.872871
10	rayon	1	4.641070
11	giat	1	2.619294
12	apel	1	4.676161
13	laksana	1	2.384348
14	danurejan	1	5.523459
15	pakualaman	1	4.116545
16	gondokusuman	1	5.175152
17	sabtu	1	4.370779

Gambar 4. 10 Kode Program dan Hasil IDF

4.4.3 *Term Frequency – Inverse Document Frequency* (TF-IDF)

Setelah melakukan perhitugan TF dan IDF, Langkah selanjutnya adalah menghitung nilai TF-IDF. Nilai TF-IDF dapat diperoleh dengan megalikan nilai TF dan nilai IDF. Berikut adalah kode program dan yang digunakan untuk mengitung nilai TF-IDF :

```
tf1['tfidf'] = tf1['tf'] * tf1['idf']
tf1
```

	words	tf	idf	tfidf
0		3	0.002331	0.006994
1	polsek	1	2.628468	2.628468
2	tingkat	1	3.833978	3.833978
3	yogyakarta	1	0.269006	0.269006
4	kota	1	2.089472	2.089472
5	wilayah	1	3.045521	3.045521
6	kamtibmas	1	3.017933	3.017933
7	ganggu	1	4.224176	4.224176
8	antisipasi	1	3.947923	3.947923
9	rutin	1	4.872871	4.872871
10	rayon	1	4.641070	4.641070
11	giat	1	2.619294	2.619294
12	apel	1	4.676161	4.676161
13	laksana	1	2.384348	2.384348
14	danurejan	1	5.523459	5.523459
15	pakualaman	1	4.116545	4.116545
16	gondokusuman	1	5.175152	5.175152
17	sabtu	1	4.370779	4.370779

Gambar 4. 11 Kode Program dan Hasil TF-IDF

4.4.4 Konversi TF-IDF ke Vektor

Konversi TF-IDF ke vektor ini digunakan untuk mengubah TF-IDF menjadi bentuk bilangan vektor dan hasil konversi tersebut akan digunakan dalam menentukan jumlah kluster dengan menggunakan metode *elbow*. Berikut merupakan perintah dan hasil yang digunakan untuk melakukan proses konversi TF-IDF ke vektor :

```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
stopwords = stopwords.words('english')

dt_fiturs = dt_fiturs.dropna()
dt_fiturs['normalizing'] = dt_fiturs['normalizing'].astype(str)

tfidf = TfidfVectorizer(
    min_df=5,
    max_df=0.95,
    max_features = 10000,
    stop_words=stopwords)

text = tfidf.fit(dt_fiturs.normalizing)
text = tfidf.transform(dt_fiturs.normalizing)

print(text)
```

```
  (0, 1763)     0.059958093787433886
  (0, 1605)     0.2148662163641439
  (0, 1471)     0.20463398288001022
  (0, 880)      0.9144734190821749
  (0, 130)      0.2685159862979385
  (1, 1763)     0.06381641436788445
  (1, 1742)     0.2033123276165407
  (1, 1616)     0.24337618079452644
  (1, 1321)     0.26910533979418555
  (1, 1319)     0.29570293052192487
  (1, 1278)     0.28402759493384433
  (1, 1194)     0.18830046599095118
  (1, 1092)     0.2565908896154078
  (1, 828)      0.1705436681099161
  (1, 793)      0.15886833252183555
  (1, 678)      0.2322921494543508
  (1, 471)      0.3080492076147261
  (1, 461)      0.18219520601626069
  (1, 447)      0.2618976662012436
  (1, 327)      0.32444001554073737
  (1, 79)       0.28579506724714715
  (1, 75)       0.2482542927295014
  (2, 1763)     1.0
  (3, 1763)     0.0643531096923523
  (3, 1760)     0.30528849145645653
```

Gambar 4. 12 Kode Program dan Hasil Konversi TF-IDF ke Vektor

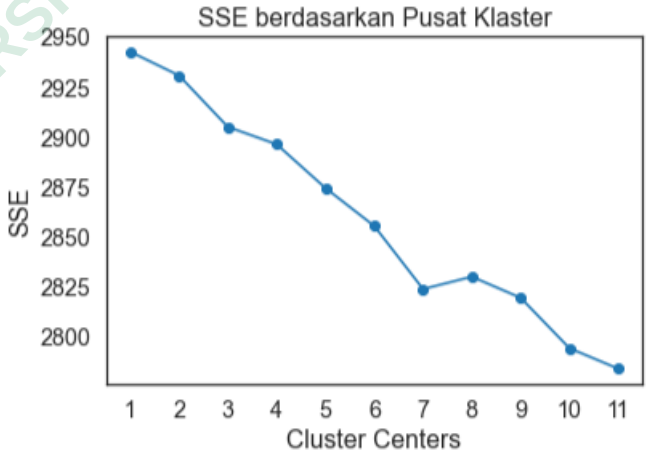
4.5 K-MEANS

Algoritma K-Means memiliki beberapa tahapan, dimulai dengan menentukan jumlah kluster yang akan digunakan, dalam hal ini menggunakan metode *elbow method*.. Berikut adalah perintah dan hasil yang digunakan untuk menjalankan proses *elbow method* dalam menentukan jumlah kluster :

```
from sklearn.cluster import KMeans
from sklearn.cluster import MiniBatchKMeans
def find_optimal_clusters(data, max_k):
    iters = range(1, max_k+1, 1)

    sse = []
    for k in iters:
        sse.append(MiniBatchKMeans(n_clusters=k,
                                   init_size=1025,
                                   batch_size=2049,
                                   random_state=11).fit(data).inertia_)
        print('Fit {} klaster'.format(k))

    f, ax = plt.subplots(1, 1)
    ax.plot(iters, sse, marker='o')
    ax.set_xlabel('Cluster Centers')
    ax.set_xticks(iters)
    ax.set_xticklabels(iters)
    ax.set_ylabel('SSE')
    ax.set_title('SSE berdasarkan Pusat Klaster')

# Contoh penggunaan
find_optimal_clusters(text, 11)
```

Gambar 4. 13 Kode Program *Elbow Method*

```
Fit 1 klaster
Fit 2 klaster
Fit 3 klaster
Fit 4 klaster
Fit 5 klaster
Fit 6 klaster
Fit 7 klaster
Fit 8 klaster
Fit 9 klaster
Fit 10 klaster
Fit 11 klaster
```



Gambar 4. 14 Hasil *Elbow Method*

Berdasarkan gambar 4.14, *elbow method* diterapkan menggunakan perhitungan *Sum of Squared Error* (SSE) untuk menentukan jumlah kluster yang optimal. Dalam gambar tersebut, terlihat bahwa proses perhitungan SSE mengalami penurunan yang signifikan pada nilai K tertentu, dan nilai K yang membentuk titik siku adalah 7. Setelah menemukan nilai kluster terbaik, langkah selanjutnya adalah membagi semua data *tweet* yang telah diproses menjadi 7 kluster. Berikut adalah perintah dan hasil yang digunakan untuk melakukan pembagian data tweet menjadi 7 kluster:

```
from sklearn.cluster import MiniBatchKMeans

# Menghasilkan kluster menggunakan MiniBatchKMeans
clusters = MiniBatchKMeans(n_clusters=7, init_size=1025, batch_size=2049,
                           random_state=11).fit_predict(text)

# Membuat DataFrame dari data teks dan kluster
text_clusters = {'text': text, 'cluster': clusters}
df = pd.DataFrame(text_clusters, columns = ['text', 'cluster'])
text = tfidf.inverse_transform(text) # mengubah kembali ke bentuk string
text = [' '.join(t) for t in text] # menggabungkan token menjadi kalimat
text_clusters = {'text': text, 'cluster': clusters}
df = pd.DataFrame(text_clusters, columns = ['text', 'cluster'])

print(df)
df['cluster'].value_counts()
```

```
                                                   text  cluster
0                          yogyakarta tidak solo lo bahaya        3
1     yogyakarta wilayah tingkat sabtu rutin rayon p...        2
2                                             yogyakarta        4
3     yogyakarta yk utama senja senen rute pasar kur...        0
4        yogyakarta simak rilis prakira hari cuaca bmkg        3
...                                                 ...      ...
2994                       yogyakarta jakarta istimewa        4
2995                    yogyakarta universitas negeri        3
2996     smk sma ppdb nilai guna gabung diy cara artikel        0
2997                                             jogja        5
2998  tunggu slot semua ready promo nya inc hari hal...        3

[2999 rows x 2 columns]

3    1402
0     574
2     351
1     224
5     212
4     139
6      97
```

Gambar 4. 15 Kode Program dan Hasil *Clustering*

Setelah data *tweet* dikelompokkan ke dalam kluster-kluster, Langkah selanjutnya yaitu menampilkan kata-kata dalam setiap kluster menjadi sebanyak 7 kluster. Berikut adalah perintah dan hasil yang digunakan :

```
def get_top_keywords(data, clusters, labels):
    data_matrix = tfidf.transform(data)
    df = pd.DataFrame(data_matrix.todense()).groupby(clusters).mean()
    for i in df.index:
        print('\nCluster {}'.format(i))
        cluster_text = df.loc[i, :]
        cluster_keywords = [labels[t] for t in np.argsort(cluster_text)]
        print(','.join(cluster_keywords))

get_top_keywords(text, clusters, tfidf.get_feature_names())
```

Gambar 4. 16 Kode Program *Cluster*

Tabel 4. 2 Hasil Sampel

Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
yogyakarta budaya ramai sekolah rekomendasi mahasiswa masyarakat sejahtera universitas kuliah	fasilitas fungsi transportasi nyaman jogja tiket jadwal pariwisata pantau kondusif	patroli polsek polresta jogja personil aman upaya solusi masalah sosial	jalan padat cegah operasi kawasan parkir lokasi kunjung prakira jogja	yogyakarta info tidak bahaya waspada kriminalitas kasus risiko kondisi situasi	loker lowongan admin informasi kerja pegawai administrasi yogyakarta daftar industri	harap himbauan izin pesantren ponpes pondok yogyakarta ibadah komunitas hadir

Berdasarkan hasil pada tabel 4.2 penentuan jenis konten setiap kluster diperoleh hasil sebagai berikut :

1. Kluster 0, membahas Kota Yogyakarta sebagai pusat budaya yang ramai dengan keberadaan universitas, rekomendasi sekolah, dan mahasiswa yang menjadikan masyarakatnya sejahtera.
2. Kluster 1, membahas pariwisata di Yogyakarta dengan fokus pada fasilitas transportasi yang nyaman, tiket dan jadwal, serta upaya menjaga kondisi pariwisata yang kondusif

3. Kluster 2, membahas keamanan dan kepolisian di Yogyakarta dengan upaya patroli dari polsek dan polresta untuk memberikan solusi pada masalah sosial dan menjaga keamanan.
4. Kluster 3, membahas kondisi umum, resiko yang ada dan isu-isu terkait kriminalitas dan situasi kota agar lebih waspada.
5. Kluster 4, membahas lalu lintas dan transportasi di Yogyakarta dengan fokus pada penanganan jalan padat, operasi kepolisian, dan prakiraan kondisi lalu lintas di berbagai lokasi.
6. Kluster 5, membahas informasi lowongan pekerjaan dan industri di Yogyakarta dengan fokus pada lowongan kerja, administrasi, dan pendaftaran.
7. Kluster 6, membahas himbauan terkait izin dan kehadiran dalam kegiatan ibadah di pesantren, pondok, dan komunitas di Yogyakarta.

Secara keseluruhan, hasil analisis menunjukkan bahwa masyarakat memiliki beragam pandangan dan minat terhadap Yogyakarta, termasuk dalam hal kesempatan kerja, budaya, pariwisata, keamanan, kehadiran pihak terkait, dan kegiatan ibadah.