TUGAS 6: MEMBANGUN FITUR DAN MELAKUKAN KALSIFIKASI DENGAN METODE SVM DAN KNN

disusun untuk memenuhi tugas

Teks dan Web Mining

Oleh:

SITI NURRAHMASITA

(2108107010015)

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

DARUSSALAM, BANDA ACEH

2024

A. Deskripsi Tugas

Menggunakan sekumpulan halaman web berkategori POSITIF (+) dan halaman web berkategori NEGATIVE (-) yang sesuai dengan kategori yang telah dikumpulkan pada tugas 4 sebelumnya, anda diminta untuk mengumpulkan minimal 8000 halaman web gabungan dari dua kategori tersebut. Jadi lebih kurang 3000 untuk masing-masing kategori (2 kategori) telah digunakan untuk membangun kamus. Lakukan preproses untuk ke 2000 file baru tersebut dan bangkitkan fitur untuk setiap halaman dari 6 bagian dalam halaman web, yaitu bagian title, bagian content (bagian konten ini dibagi menjadi 5 sub bagian). Pelajari bahan kuliah yang membahas tentang pembangkitan fitur untuk membantu penyelesaian tugas ini. Gunakan kamus POSITIF dan NEGATIF yang telah anda bangkitkan pada tugas sebelumnya. Fitur disusun dalam format ARFF dan format SVM Light atau format lain untuk menjalankan SVM (boleh menggunakan scikit-learn).

Gunakan 80% dari 2000 data yang telah dibangkitkan fiturnya untuk training set dan 20% sisanya untuk menjadi testing set. Gunakan metode SVM dan KNN untuk melakukan klasifikasi dan tentukan kinerjanya.

Buat laporan yang menjelaskan hasil yang diperoleh. Format laporan dapat mengikuti format laporan tugas sebelumnya. File Laporan Tugas 6 dalam format PDF dikumpulkan paling telat tanggal 13 Mei 2024 NAMA_NIM.tar.gz.

B. Pendahuluan

Proses pengolahan dan analisis data memiliki peran yang semakin penting dalam mendukung pengambilan keputusan yang tepat dan efisien di berbagai bidang. Dalam rangka menjawab tantangan tersebut, tugas ini bertujuan untuk mengimplementasikan serangkaian langkah dari preprocessing hingga pembuatan fitur dan klasifikasi pada data teks. Melalui langkah-langkah ini, diharapkan dapat diperoleh informasi yang lebih terstruktur dan dapat dipahami dengan baik, sehingga mendukung pengambilan keputusan yang lebih cerdas.

a. Langkah-langkah Pengolahan Data

Langkah awal dalam proses ini adalah melakukan preprocessing terhadap teks yang telah diambil dari sumbernya. Preprocessing dilakukan untuk membersihkan teks dari karakter-karakter yang tidak relevan dan mengubahnya ke dalam format yang lebih sesuai untuk analisis selanjutnya. Setelah itu, dilakukan penghapusan duplikasi kata-kata berdasarkan observasi eliminasi rasio tertentu. Hal ini bertujuan untuk memastikan bahwa data yang digunakan dalam analisis merupakan representasi yang akurat dari informasi yang ada.

b. Pembuatan Fitur dan Klasifikasi

Selanjutnya, dilakukan pembuatan fitur-fitur yang merepresentasikan informasi dari teks. Fitur-fitur ini dihasilkan melalui penghitungan frekuensi kemunculan kata-kata dalam teks yang telah diproses sebelumnya. Fitur-fitur tersebut kemudian digunakan dalam proses klasifikasi menggunakan algoritma SVM (Support Vector Machine) dan KNN (K-Nearest Neighbors). Dengan demikian, diharapkan dapat diperoleh model klasifikasi yang mampu mengidentifikasi dan mengklasifikasikan data dengan akurasi yang tinggi.

C. Penjelasan Umum

a. Remove-One-Gram.py, Remove-Two-Gram.py, Remove-Three-Gram.py:

Ketiga file tersebut bertanggung jawab untuk menghapus duplikasi kata-kata dari kamus-kamus yang telah dihasilkan sebelumnya. Proses penghapusan duplikasi dilakukan berdasarkan observasi eliminasi rasio, di mana kata-kata yang memiliki frekuensi kemunculan yang rendah dibandingkan dengan kata-kata yang serupa akan dihapus. Ini bertujuan untuk memastikan bahwa kamus-kamus yang digunakan dalam analisis hanya mengandung kata-kata yang relevan dan memiliki frekuensi kemunculan yang cukup signifikan.

b. Create-Feature.py:

File ini bertanggung jawab untuk menghasilkan fitur-fitur yang akan digunakan dalam proses klasifikasi data. Fitur-fitur ini dihasilkan dari analisis teks yang telah diproses sebelumnya, seperti judul dan konten artikel. Proses pembuatan fitur melibatkan perhitungan frekuensi kemunculan kata-kata dalam teks, dengan mempertimbangkan bagian-bagian tertentu dari teks, seperti bagian judul, bagian atas, tengah, dan bawah dari konten artikel. Fitur-fitur ini kemudian akan digunakan sebagai masukan untuk algoritma klasifikasi.

c. Classification.py:

File ini berisi implementasi dari dua algoritma klasifikasi, yaitu Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN). Algoritma SVM dan KNN digunakan untuk melakukan klasifikasi terhadap data berdasarkan fitur-fitur yang telah dihasilkan sebelumnya. Tujuan dari proses klasifikasi ini adalah untuk mengklasifikasikan data ke dalam kategori yang tepat berdasarkan informasi yang terkandung dalam teksnya. Akurasi dari masing-masing model klasifikasi juga dihitung untuk mengevaluasi performanya dalam mengklasifikasikan data.

D. Penjelasan Scripts

a. Remove-one-gram.py, remove-two-gram.py, dan remove-three-gram.py

```python
import os

# Fungsi untuk membaca kamus dari file dengan format frekuensi1,kata1
def read_dictionary(file_path):
    dictionary = {}
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            line = line.strip()
            if line:
                # Ambil bagian frekuensi dan kata
                frequency = line.split()[-1]  # Ambil data di bagian akhir
                word = line[:-len(frequency)].strip()  # Ambil sisa sebagai kata
                dictionary[word] = float(frequency)
            else:
                print("Error: Empty line found.")
    return dictionary

# Fungsi untuk menghapus duplikasi kata berdasarkan observasi eliminasi rasio
def remove_duplicates(dictionary_A, dictionary_B, threshold):
    unique_A = {}
    unique_B = {}

    for word, freq_A in dictionary_A.items():
        freq_B = dictionary_B.get(word, 0)

        # Hitung nilai rasio
        max_freq = max(freq_A, freq_B)
        min_freq = min(freq_A, freq_B)
        ratio = min_freq / max_freq if max_freq != 0 else 0

        # Hapus kata duplikat berdasarkan threshold
        if ratio < threshold:
            if freq_A >= freq_B:
                unique_A[word] = freq_A
            else:
                unique_B[word] = freq_B

    return unique_A, unique_B

# Fungsi untuk menyimpan kamus ke file
def save_dictionary(dictionary, output_file):
    with open(output_file, 'w', encoding='utf-8') as file:
        # Urutkan kamus berdasarkan frekuensi dari yang terbesar ke terkecil
        sorted_dict = sorted(dictionary.items(), key=lambda x: x[1], reverse=True)
        for word, frequency in sorted_dict:
            if frequency.is_integer():
                file.write(f"{int(frequency)}\t{word}\n")
            else:
                file.write(f"{frequency:.1f}\t{word}\n")

# Path ke kamus berita dan sepakbola dengan format frekuensi1,kata1
path_news = 'kamus/one-gram/berita_txts.txt'
path_football = 'kamus/one-gram/sepakbola_txts.txt'

# Baca kamus berita dan sepakbola
dict_news = read_dictionary(path_news)
dict_football = read_dictionary(path_football)

# Threshold untuk observasi eliminasi rasio
threshold_1 = 0.50
threshold_2 = 0.55

# Hapus duplikasi kata berdasarkan threshold
unique_news_1, unique_football_1 = remove_duplicates(dict_news, dict_football, threshold_1)
unique_news_2, unique_football_2 = remove_duplicates(dict_news, dict_football, threshold_2)

# Simpan kamus yang telah dihapus duplikasinya
output_news_1 = 'kamus_distinct/berita/one-gram_unique_50.txt'
output_football_1 = 'kamus_distinct/sepakbola/one-gram_unique_50.txt'
output_news_2 = 'kamus_distinct/berita/one-gram_unique_55.txt'
output_football_2 = 'kamus_distinct/sepakbola/one-gram_unique_55.txt'

save_dictionary(unique_news_1, output_news_1)
save_dictionary(unique_football_1, output_football_1)
save_dictionary(unique_news_2, output_news_2)
save_dictionary(unique_football_2, output_football_2)

print("Penghapusan duplikasi selesai.")
```

Gambar D.a.1. Scripts remove-one-gram.py

```python
import os

# Fungsi untuk membaca kamus dari file dengan format frekuensi1,kata1
def read_dictionary(file_path):
    dictionary = {}
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            line = line.strip()
            if line:
                # Ambil bagian frekuensi dan kata
                frequency = line.split()[-1]  # Ambil data di bagian akhir
                word = line[:-len(frequency)].strip()  # Ambil sisa sebagai kata
                dictionary[word] = float(frequency)
            else:
                print("Error: Empty line found.")
    return dictionary

# Fungsi untuk menghapus duplikasi kata berdasarkan observasi eliminasi rasio
def remove_duplicates(dictionary_A, dictionary_B, threshold):
    unique_A = {}
    unique_B = {}

    for word, freq_A in dictionary_A.items():
        freq_B = dictionary_B.get(word, 0)

        # Hitung nilai rasio
        max_freq = max(freq_A, freq_B)
        min_freq = min(freq_A, freq_B)
        ratio = min_freq / max_freq if max_freq != 0 else 0

        # Hapus kata duplikat berdasarkan threshold
        if ratio < threshold:
            if freq_A >= freq_B:
                unique_A[word] = freq_A
            else:
                unique_B[word] = freq_B

    return unique_A, unique_B

# Fungsi untuk menyimpan kamus ke file
def save_dictionary(dictionary, output_file):
    with open(output_file, 'w', encoding='utf-8') as file:
        # Urutkan kamus berdasarkan frekuensi dari yang terbesar ke terkecil
        sorted_dict = sorted(dictionary.items(), key=lambda x: x[1], reverse=True)
        for word, frequency in sorted_dict:
            freq_str = str(int(frequency)) if frequency.is_integer() else f"{frequency:.1f}"
            file.write(f"{freq_str}\t{word}\n")

# Path ke kamus berita dan sepakbola
path_news = 'kamus/two-gram/berita_txts.txt'
path_football = 'kamus/two-gram/sepakbola_txts.txt'

# Baca kamus berita dan sepakbola
dict_news = read_dictionary(path_news)
dict_football = read_dictionary(path_football)

# Threshold untuk observasi eliminasi rasio
threshold_1 = 0.50
threshold_2 = 0.55

# Hapus duplikasi kata berdasarkan threshold
unique_news_1, unique_football_1 = remove_duplicates(dict_news, dict_football, threshold_1)
unique_news_2, unique_football_2 = remove_duplicates(dict_news, dict_football, threshold_2)

# Simpan kamus yang telah dihapus duplikasinya
output_news_1 = 'kamus_distinct/berita/two-gram_unique_50.txt'
output_football_1 = 'kamus_distinct/sepakbola/two-gram_unique_50.txt'
output_news_2 = 'kamus_distinct/berita/two-gram_unique_55.txt'
output_football_2 = 'kamus_distinct/sepakbola/two-gram_unique_55.txt'

save_dictionary(unique_news_1, output_news_1)
save_dictionary(unique_football_1, output_football_1)
save_dictionary(unique_news_2, output_news_2)
save_dictionary(unique_football_2, output_football_2)

print("Penghapusan duplikasi selesai.")
```

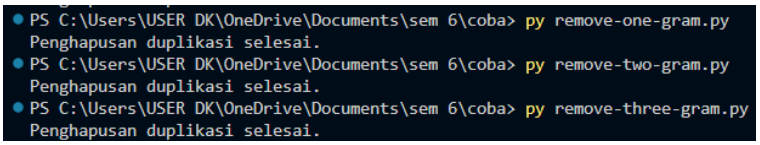
Gambar D.a.2. Scripts remove-two-gram.py

```python
import os

# Fungsi untuk membaca kamus dari file dengan format frekuensi1,kata1
def read_dictionary(file_path):
    dictionary = {}
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            line = line.strip()
            if line:
                # Ambil bagian frekuensi dan kata
                frequency = line.split()[-1]  # Ambil data di bagian akhir
                word = line[:-len(frequency)].strip()  # Ambil sisa sebagai kata
                dictionary[word] = float(frequency)
            else:
                print("Error: Empty line found.")
    return dictionary

# Fungsi untuk menghapus duplikasi kata berdasarkan observasi eliminasi rasio
def remove_duplicates(dictionary_A, dictionary_B, threshold):
    unique_A = {}
    unique_B = {}

    for word, freq_A in dictionary_A.items():
        freq_B = dictionary_B.get(word, 0)

        # Hitung nilai rasio
        max_freq = max(freq_A, freq_B)
        min_freq = min(freq_A, freq_B)
        ratio = min_freq / max_freq if max_freq != 0 else 0

        # Hapus kata duplikat berdasarkan threshold
        if ratio < threshold:
            if freq_A >= freq_B:
                unique_A[word] = freq_A
            else:
                unique_B[word] = freq_B

    return unique_A, unique_B

# Fungsi untuk menyimpan kamus ke file
def save_dictionary(dictionary, output_file):
    with open(output_file, 'w', encoding='utf-8') as file:
        # Urutkan kamus berdasarkan frekuensi dari yang terbesar ke terkecil
        sorted_dict = sorted(dictionary.items(), key=lambda x: x[1], reverse=True)
        for word, frequency in sorted_dict:
            freq_str = str(int(frequency)) if frequency.is_integer() else f"{frequency:.1f}"
            file.write(f"{freq_str}\t{word}\n")

# Path ke kamus berita dan sepakbola
path_news = 'kamus/three-gram/berita_txts.txt'
path_football = 'kamus/three-gram/sepakbola_txts.txt'

# Baca kamus berita dan sepakbola
dict_news = read_dictionary(path_news)
dict_football = read_dictionary(path_football)

# Threshold untuk observasi eliminasi rasio
threshold_1 = 0.50
threshold_2 = 0.55

# Hapus duplikasi kata berdasarkan threshold
unique_news_1, unique_football_1 = remove_duplicates(dict_news, dict_football, threshold_1)
unique_news_2, unique_football_2 = remove_duplicates(dict_news, dict_football, threshold_2)

# Simpan kamus yang telah dihapus duplikasinya
output_news_1 = 'kamus_distinct/berita/three-gram_unique_50.txt'
output_football_1 = 'kamus_distinct/sepakbola/three-gram_unique_50.txt'
output_news_2 = 'kamus_distinct/berita/three-gram_unique_55.txt'
output_football_2 = 'kamus_distinct/sepakbola/three-gram_unique_55.txt'

save_dictionary(unique_news_1, output_news_1)
save_dictionary(unique_football_1, output_football_1)
save_dictionary(unique_news_2, output_news_2)
save_dictionary(unique_football_2, output_football_2)

print("Penghapusan duplikasi selesai.")
```

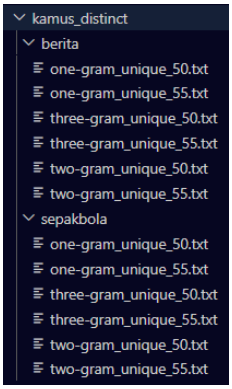
Gambar D.a.3. Scripts remove-three-gram.py

File Remove-One-Gram.py, Remove-Two-Gram.py, dan Remove-Three-Gram.py, memiliki tujuan yang serupa, yaitu menghapus duplikasi kata-kata dari kamus-kamus yang telah dibuat sebelumnya, namun dengan tingkat n-gram berbeda. Berikut adalah tahapan-tahapannya:

1. Pada tahap awal, masing-masing skrip membaca kamus yang berisi daftar kata dan frekuensi kemunculannya dari file teks. Fungsi ‘read_dictionary(file_path)’ digunakan untuk ini. Fungsi ini membuka file, membaca setiap baris, dan memisahkan kata dari frekuensinya untuk dimasukkan ke dalam sebuah dictionary.
2. Setelah membaca kamus-kamus, dilakukan proses penghapusan duplikasi kata-kata. Hal ini dilakukan dengan membandingkan frekuensi kemunculan kata-kata antara kamus satu dengan kamus lainnya. Fungsi ‘remove_duplicates(dictionary_A, dictionary_B, threshold)’ melakukan ini dengan membandingkan frekuensi kemunculan kata yang sama di kedua kamus, maka kata tersebut dianggap sebagai duplikasi dan akan dihapus.
3. Proses penghapusan duplikasi didasarkan pada observasi eliminasi rasio. Rasio ini merupakan perbandingan antara frekuensi kemunculan kata yang akan dihapus dengan frekuensi kemunculan kata yang serupa di kamus lain. Jika rasio ini lebih kecil dari threshold tertentu, maka kata tersebut dihapus.
4. Setelah proses penghapusan duplikasi, kamus yang telah dibersihkan disimpan kembali ke file teks. Fungsi ‘save_dictionary(dictionary, output_file)’ digunakan untuk menyimpan dictionary ke dalam file. Fungsi ini juga mengurutkan kata-kata berdasarkan frekuensinya dari yang terbesar hingga terkecil sebelum menyimpannya. Kamus akan ditampilkan sebagai ouput dengan format frekuensi dan data.



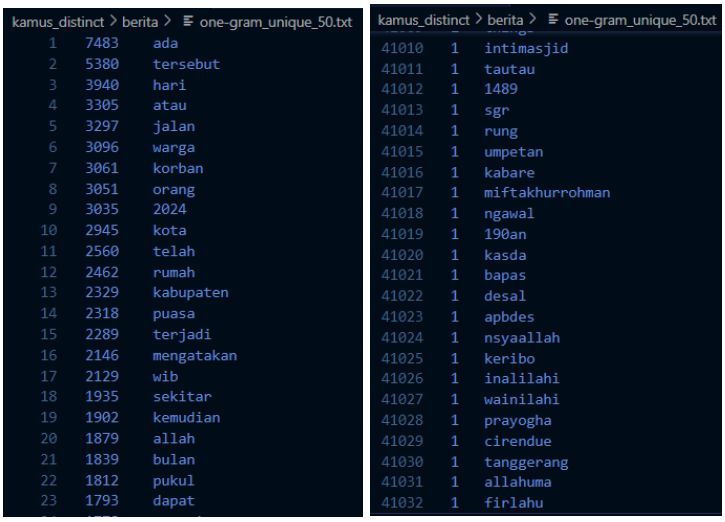
Gambar D.a.4. Output sukses membangun kamus one-gram, two-gram, dan three-gram



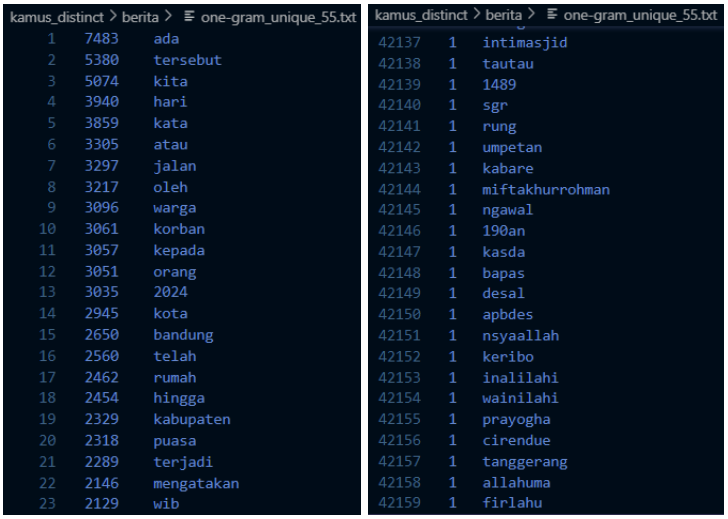
Gambar D.a.5. Directory yang dihasilkan

Output yang dihasilkan:

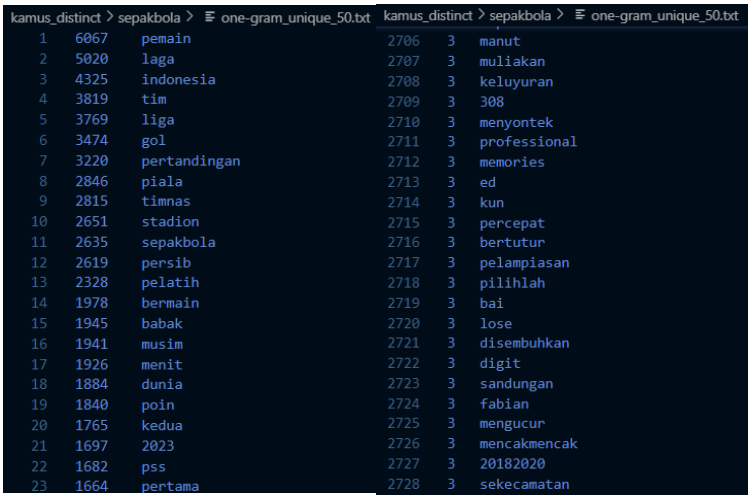
1. Remove-one-gram.py



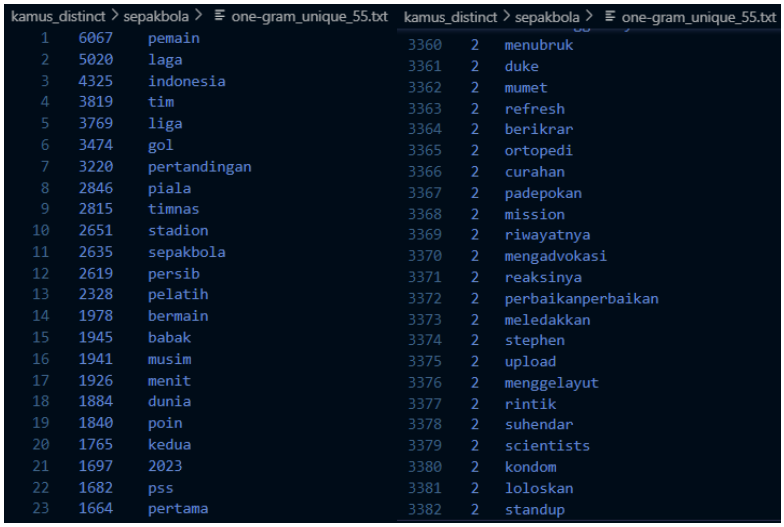
Gambar D.a.6. Hasil kamus one-gram_unique_50.txt kategori berita



Gambar D.a.7. Hasil kamus one-gram_unique_55.txt kategori berita

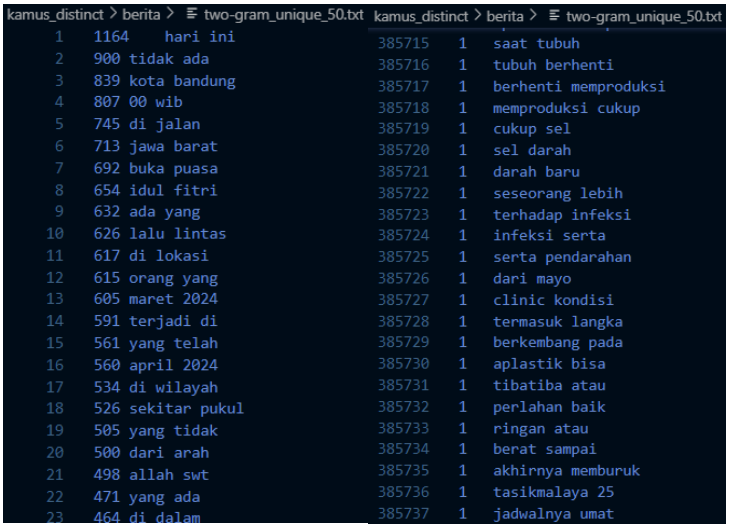


Gambar D.a.8. Hasil kamus one-gram_unique_50.txt kategori sepakbola

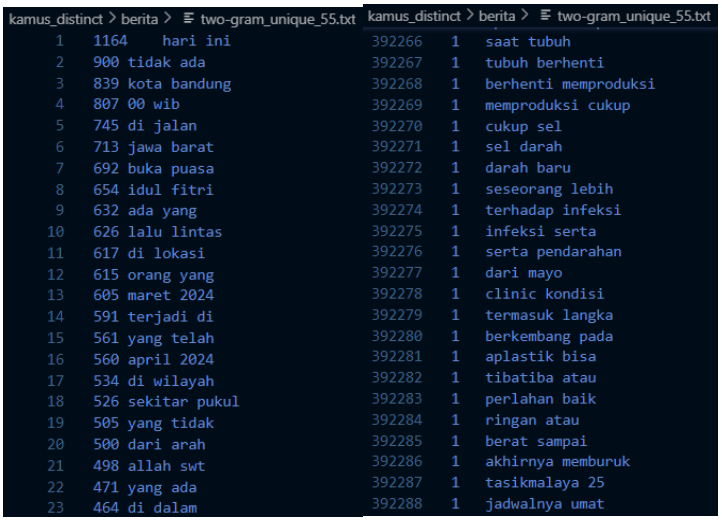


Gambar D.a.9. Hasil kamus one-gram_unique_55.txt kategori sepakbola

2. Remove-two-gram.py



Gambar D.a.10. Hasil kamus two-gram_unique_50.txt kategori berita



Gambar D.a.11. Hasil kamus two-gram_unique_55.txt kategori berita

kamus_distinct > sepakbola > two-gram_unique_50.txt

```
1    1544    di stadion
2    1308    liga 1
3    1288    piala dunia
4    1066    timnas indonesia
5    988 piala asia
6    803 di laga
7    788 di liga
8    765 pss sleman
9    722 di babak
10   681 musim ini
11   609 1 20232024
12   606 pemain yang
13   586 para pemain
14   571 dengan skor
15   544 liga 2
16   533 persib bandung
17   532 di piala
18   528 mencetak gol
19   508 shin taeyong
20   490 di posisi
21   485 tuan rumah
22   466 persis solo
23   452 lolos ke
```

kamus_distinct > sepakbola > two-gram_unique_50.txt

```
7870   3   sejak era
7871   3   tidak terkejut
7872   3   penting daripada
7873   3   memiliki opsi
7874   3   dan ceo
7875   3   pernah mengungkapkan
7876   3   akan berperan
7877   3   telah menganalisis
7878   3   bisa berfoto
7879   3   kok ujar
7880   3   tak bikin
7881   3   kini semua
7882   3   dirinya mau
7883   3   rp 95
7884   3   ikut di
7885   3   disayangkan oleh
7886   3   sebagai pahlawan
7887   3   mereka kami
7888   3   tidak membantu
7889   3   untuk diganti
7890   3   gaji mereka
7891   3   aspirasi mereka
7892   3   ayah kami
```

Gambar D.a.12. Hasil kamus two-gram_unique_50.txt kategori sepakbola

kamus_distinct > sepakbola > two-gram_unique_55.txt

```
1    1544    di stadion
2    1308    liga 1
3    1288    piala dunia
4    1066    timnas indonesia
5    988 piala asia
6    803 di laga
7    788 di liga
8    765 pss sleman
9    722 di babak
10   681 musim ini
11   609 1 20232024
12   606 pemain yang
13   586 para pemain
14   571 dengan skor
15   544 liga 2
16   533 persib bandung
17   532 di piala
18   528 mencetak gol
19   508 shin taeyong
20   490 di posisi
21   485 tuan rumah
22   466 persis solo
23   452 lolos ke
```

kamus_distinct > sepakbola > two-gram_unique_55.txt

```
12798   2   burung dan
12799   2   lihat kita
12800   2   sampai dewasa
12801   2   kebetulan berada
12802   2   hadiah itu
12803   2   itu balik
12804   2   prestasi para
12805   2   mereka kita
12806   2   meninggalkan motornya
12807   2   bantul untuk
12808   2   oleh pemangku
12809   2   ikut main
12810   2   yang memunculkan
12811   2   abad ke10
12812   2   ke12 di
12813   2   wib karena
12814   2   persidangan dan
12815   2   persidangan dengan
12816   2   masa persiapan
12817   2   2020 dengan
12818   2   proyek apa
12819   2   ada 44
12820   2   serius yang
```

Gambar D.a.13. Hasil kamus two-gram_unique_55.txt kategori sepakbola

3. Remove-three-gram.py

kamus_distinct > berita > three-gram_unique_50.txt

```
1    343 jadwal buka puasa
2    313 yang ada di
3    240 di media sosial
4    209 arus lalu lintas
5    206 selengkapnya di sini
6    205 yang berada di
7    168 di bulan ramadhan
8    165 di kota bandung
9    150 kabupaten bandung barat
10   147 saat ditemui di
11   145 ke rumah sakit
12   144 gibran rakabuming raka
13   138 di lokasi kejadian
14   137 hari raya idul
15   137 raya idul fitri
16   137 أكبر الله أكبر
17   131 saat ini masih
18   129 di jawa barat
19   128 artikel ini ditulis
20   126 dan wakil presiden
21   126 ini ditulis oleh
22   126 kampus merdeka di
23   126 merdeka di detikcom
```

kamus_distinct > berita > three-gram_unique_50.txt

```
811851   1   aplastik bisa terjadi
811852   1   terjadi secara tibatiba
811853   1   secara tibatiba atau
811854   1   tibatiba atau bisa
811855   1   bisa juga terjadi
811856   1   juga terjadi secara
811857   1   terjadi secara perlahan
811858   1   secara perlahan baik
811859   1   perlahan baik ringan
811860   1   baik ringan atau
811861   1   ringan atau berat
811862   1   atau berat sampai
811863   1   berat sampai akhirnya
811864   1   sampai akhirnya memburuk
811865   1   akhirnya memburuk seiring
811866   1   memburuk seiring berjalannya
811867   1   waktu imsak garut
811868   1   dan tasikmalaya 25
811869   1   tasikmalaya 25 maret
811870   1   2024 ini jadwalnya
811871   1   ini jadwalnya umat
811872   1   jadwalnya umat islam
811873   1   tasikmalaya hari senin
```

Gambar D.a.14. Hasil kamus three-gram_unique_50.txt kategori berita

kamus_distinct > berta > three-gram_unique_55.txt

1	343	jadwal buka puasa
2	313	yang ada di
3	240	di media sosial
4	209	arus lalu lintas
5	206	selengkapnya di sini
6	205	yang berada di
7	168	di bulan ramadhan
8	165	di kota bandung
9	150	kabupaten bandung barat
10	147	saat ditemui di
11	145	ke rumah sakit
12	144	gibran rakabuming raka
13	138	di lokasi kejadian
14	137	hari raya idul
15	137	raya idul fitri
16	137	أكبر الله أكبر
17	131	saat ini masih
18	129	di jawa barat
19	128	artikel ini ditulis
20	126	dan wakil presiden
21	126	ini ditulis oleh
22	126	kampus merdeka di
23	126	merdeka di detikcom

kamus_distinct > berta > three-gram_unique_55.txt

815185	1	aplastik bisa terjadi
815186	1	terjadi secara tibatiba
815187	1	secara tibatiba atau
815188	1	tibatiba atau bisa
815189	1	bisa juga terjadi
815190	1	juga terjadi secara
815191	1	terjadi secara perlahan
815192	1	secara perlahan baik
815193	1	perlahan baik ringan
815194	1	baik ringan atau
815195	1	ringan atau berat
815196	1	atau berat sampai
815197	1	berat sampai akhirnya
815198	1	sampai akhirnya memburuk
815199	1	akhirnya memburuk seiring
815200	1	memburuk seiring berjalannya
815201	1	waktu imsak garut
815202	1	dan tasikmalaya 25
815203	1	tasikmalaya 25 maret
815204	1	2024 ini jadwalnya
815205	1	ini jadwalnya umat
815206	1	jadwalnya umat islam
815207	1	tasikmalaya hari senin

Gambar D.a.15. Hasil kamus three-gram_unique_55.txt kategori berita

kamus_distinct > sepakbola > three-gram_unique_50.txt

1	608	liga 1 20232024
2	375	piala asia u23
3	375	piala asia 2023
4	269	di piala asia
5	260	di stadion manahan
6	240	dini hari wib
7	230	stadion manahan solo
8	220	asia u23 2024
9	186	david da silva
10	185	timnas indonesia u23
11	151	kualifikasi piala dunia
12	149	di piala dunia
13	147	berlangsung di stadion
14	131	piala dunia 2026
15	120	dalam lanjutan liga
16	118	lanjutan liga 1
17	115	berada di posisi
18	115	si jalak harupat
19	113	bandung lautan api
20	111	akan digelar di
21	110	di liga 2
22	106	pukul 1900 wib
23	106	sesi jumpa pers

kamus_distinct > sepakbola > three-gram_unique_50.txt

2533	3	bukan hanya dari
2534	3	nothing to lose
2535	3	pada senin 294
2536	3	dalam pernyataannya yang
2537	3	saya bisa melihat
2538	3	yang sedang terjadi
2539	3	situasi yang kurang
2540	3	kabag ops polresta
2541	3	ia juga berterima
2542	3	merupakan kali kedua
2543	3	besar dan itu
2544	3	sarung tangan dan
2545	3	tepatnya di rt
2546	3	tidak bisa berbuat
2547	3	hasil penelitian yang
2548	3	lebih penting daripada
2549	3	keras lagi untuk
2550	3	oleh sejumlah pejabat
2551	3	selama 2 hari
2552	3	pungkasnya seperti diketahui
2553	3	tidak pernah melihat
2554	3	halhal di luar
2555	3	dengan cara seperti

Gambar D.a.16. Hasil kamus three-gram_unique_50.txt kategori sepakbola

kamus_distinct > sepakbola > three-gram_unique_55.txt

1	608	liga 1 20232024
2	375	piala asia u23
3	375	piala asia 2023
4	269	di piala asia
5	260	di stadion manahan
6	240	dini hari wib
7	230	stadion manahan solo
8	220	asia u23 2024
9	186	david da silva
10	185	timnas indonesia u23
11	151	kualifikasi piala dunia
12	149	di piala dunia
13	147	berlangsung di stadion
14	131	piala dunia 2026
15	120	dalam lanjutan liga
16	118	lanjutan liga 1
17	115	berada di posisi
18	115	si jalak harupat
19	113	bandung lautan api
20	111	akan digelar di
21	110	di liga 2
22	106	pukul 1900 wib
23	106	sesi jumpa pers

kamus_distinct > sepakbola > three-gram_unique_55.txt

5052	2	kami meminta agar
5053	2	karena jika tidak
5054	2	jika tidak bisa
5055	2	saat ini posisi
5056	2	pernah mengungkapkan bahwa
5057	2	oleh menteri dalam
5058	2	dengan misi yang
5059	2	europa untuk pertama
5060	2	bersama jokowi di
5061	2	jokowi akan mengunjungi
5062	2	ini beberapa kali
5063	2	berjalan di bawah
5064	2	yang kebetulan berada
5065	2	kebetulan berada di
5066	2	taiwan dan korea
5067	2	di hari valentine
5068	2	masa lalu dan
5069	2	pernah melihat orang
5070	2	pidana yang dilakukan
5071	2	sudah lama di
5072	2	dia belum bisa
5073	2	kami bisa membawa
5074	2	sudah tiga bulan

Gambar D.a.17. Hasil kamus three-gram_unique_55.txt kategori sepakbola

b. Create_feature.py

```python
import os
import re
import random
from collections import Counter

# Fungsi untuk membersihkan teks
def clean_text(text):
    cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)
    cleaned_text = cleaned_text.lower()
    return cleaned_text

# Fungsi untuk membaca daftar stopwords dari file eksternal
def read_stopwords(file_path):
    stopwords = set()
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            word = line.strip()
            if word:
                stopwords.add(word)
    return stopwords

# Fungsi untuk menghitung jumlah kata dalam teks (tidak termasuk stopwords)
def count_words(text, stopwords):
    words = text.split()
    filtered_words = [word for word in words if word not in stopwords]
    return len(filtered_words)

# Fungsi untuk menghitung fitur untuk bagian judul
def calculate_title_feature(title, dictionary, stopwords):
    title_words = clean_text(title).split()
    total_title_words = count_words(title, stopwords)
    word_counter = Counter(title_words)
    feature = sum(word_counter[word] for word in word_counter if word in dictionary)
    return feature / total_title_words if total_title_words > 0 else 0

# Fungsi untuk menghitung fitur untuk bagian konten (bagian atas, tengah, atau bawah)
def calculate_content_feature(content, dictionary, weight, stopwords):
    content_words = clean_text(content).split()
    total_content_words = count_words(content, stopwords)
    word_counter = Counter(content_words)
    feature = sum(word_counter[word] for word in word_counter if word in dictionary)
    return weight * feature / total_content_words if total_content_words > 0 else 0

# Fungsi untuk menghitung semua fitur untuk satu halaman web
def calculate_all_features(title, content_top, content_middle, content_bottom, dictionaries, weights,
stopwords):
    features = []
    for category_dict, weight in zip(dictionaries, weights):
        category_features = []
        for dictionary in category_dict:
            title_feature = calculate_title_feature(title, dictionary, stopwords)
            top_feature = calculate_content_feature(content_top, dictionary, weight[0], stopwords)
            middle_feature = calculate_content_feature(content_middle, dictionary, weight[1],
stopwords)
            bottom_feature = calculate_content_feature(content_bottom, dictionary, weight[2],
stopwords)
            category_features.extend([title_feature, top_feature, middle_feature, bottom_feature])
        features.extend(category_features)
    return features

# Direktori tempat menyimpan file teks
directories = [r"content_splitting/berita", r"content_splitting/sepakbola"]

# Kamus-kamus untuk setiap kategori (1-kata, 2-kata, dan 3-kata)
positive_dictionaries = [
    [
        "kamus_distinct/berita/one-gram_unique_55.txt",
        "kamus_distinct/berita/two-gram_unique_55.txt",
        "kamus_distinct/berita/three-gram_unique_55.txt"
    ],
    [
        "kamus_distinct/sepakbola/one-gram_unique_55.txt",
        "kamus_distinct/sepakbola/two-gram_unique_55.txt",
        "kamus_distinct/sepakbola/three-gram_unique_55.txt"
    ]
]
```

```
# Bobot untuk bagian konten (bagian atas, tengah, dan bawah)
weights = [(0.5, 0.4, 0.3), (0.5, 0.4, 0.3)]

# Mendefinisikan path untuk file stopwords
stopwords_path = "stopword.txt"
# Membaca stopwords dari file eksternal
stopwords = read_stopwords(stopwords_path)

# Menyimpan data fitur dan label
data = {0: [], 1: []}  # 0 untuk berita, 1 untuk sepak bola

# Proses setiap folder
for label, directory in enumerate(directories):
    # Mendapatkan daftar file dalam direktori
    txt_files = [file for file in os.listdir(directory) if file.endswith(".dat")]

    # Filter file yang memiliki urutan lebih dari 3000
    txt_files = [file for file in txt_files if int(re.findall(r'\d+', file)[0]) > 3000]

    # Proses setiap file teks
    for txt_file in txt_files:
        file_path = os.path.join(directory, txt_file)
        with open(file_path, "r", encoding="utf-8") as file:
            content = file.read()

            # Pisahkan teks menjadi bagian judul, atas, tengah, dan bawah (masing-masing 30%, 40%, 30%)
            content_words = content.split()
            total_content_words = len(content_words)
            top_end = int(total_content_words * 0.3)
            middle_start = int(total_content_words * 0.3)
            middle_end = int(total_content_words * 0.7)
            title = " ".join(content_words[:top_end])
            content_top = " ".join(content_words[:top_end])
            content_middle = " ".join(content_words[middle_start:middle_end])
            content_bottom = " ".join(content_words[middle_end:])

            # Menghitung fitur untuk setiap halaman web
            features = calculate_all_features(title, content_top, content_middle, content_bottom,
positive_dictionaries, weights, stopwords)

            # Simpan fitur dan label ke dalam data
            data[label].append(features)

# Pembagian data menjadi training set (80%) dan testing set (20%)
training_data = []
testing_data = []

for label in data:
    # Acak data
    random.shuffle(data[label])
    # Hitung jumlah data untuk training set
    train_size = int(len(data[label]) * 0.8)
    # Pisahkan data
    training_data.extend([(features, label) for features in data[label][:train_size]])
    testing_data.extend([(features, label) for features in data[label][train_size:]])

# Gabungkan data training dan testing ke dalam satu file CSV
combined_data = training_data + testing_data

def save_to_csv(file_path, data):
    with open(file_path, "w", encoding="utf-8") as file:
        for features, label in data:
            file.write(",".join(map(str, features)) + "," + str(label) + "\n")

save_to_csv("combined_features.csv", combined_data)
save_to_csv("training_features.csv", training_data)
save_to_csv("testing_features.csv", testing_data)

print("Features extracted and saved to CSV files.")
```

Gambar D.b.1. Scripts create_feature.py

Berikut adalah tahapan dalam kode `create_feature.py`:

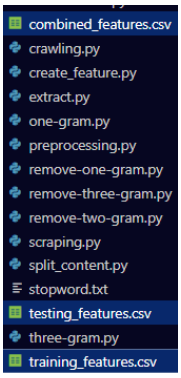
1. Fungsi `clean_text` digunakan untuk membersihkan teks dari karakter khusus dan angka, serta mengubahnya menjadi huruf kecil. Kemudian, fungsi `read_stopwords` membaca daftar stopwords dari file eksternal dan menyimpannya dalam sebuah set.
2. Fungsi `count_words` digunakan untuk menghitung jumlah kata dalam teks setelah menghapus stopwords. Selanjutnya, fungsi `calculate_title_feature` dan `calculate_content_feature` digunakan untuk menghitung fitur untuk bagian judul dan

konten (bagian atas, tengah, atau bawah) dari teks. Ini dilakukan dengan memperhitungkan kamus yang telah dibentuk, bobot bagian konten, dan stopwords.

3. Setiap file teks diproses dengan membaginya menjadi bagian judul, atas, tengah, dan bawah. Fitur-fitur diekstraksi menggunakan fungsi-fungsi yang telah disebutkan sebelumnya dalam looping, dengan memperhitungkan kamus-kamus yang relevan dan stopwords. Hasil ekstraksi fitur disimpan dalam bentuk list.
4. Data untuk masing-masing kategori diacak dan dibagi menjadi 80% untuk training set dan 20% untuk testing set. Setiap data fitur dan label disimpan dalam satu file CSV terpisah, yaitu `combined_features.csv`, `training_features.csv`, dan `testing_features.csv`. Di mana, combined_features.csv ialah penggabungan data training dan testing. Proses penyimpanan dilakukan dengan fungsi `save_to_csv`.
5. Pesan "Features extracted and saved to CSV files." ditampilkan untuk memberi informasi bahwa ekstraksi fitur telah selesai dan data telah disimpan dalam file CSV.

```
PS C:\Users\USER DK\OneDrive\Documents\sem 6\coba> py create_feature.py
Features extracted and saved to CSV files.
```

Gambar D.b.2 Output sukses membangun model hasil perhitungan fitur dari setiap artikel



Gambar D.b.3. Output yang dihasilkan

Gambar D.b.4. Tampilan isi dari file training_features.csv

Gambar D.b.5. Tampilan isi dari file testing_features.csv

Gambar D.b.6. Tampilan isi dari file combined_features.csv

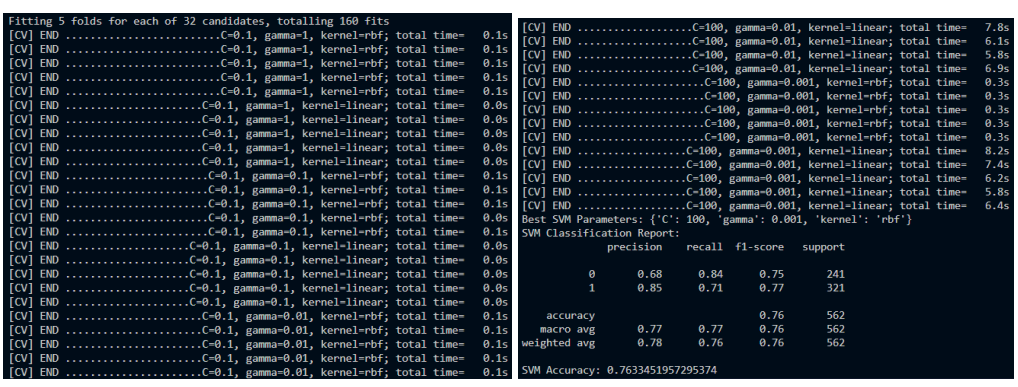
c. Classification.py

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler

# Fungsi untuk membaca data dari CSV dan memisahkan fitur dan label
def load_data(file_path):
    df = pd.read_csv(file_path, header=None)
    X = df.iloc[:, :-1].values
    y = df.iloc[:, -1].values
    return X, y

# Baca data dari CSV
X_train, y_train = load_data("training_features.csv")
X_test, y_test = load_data("testing_features.csv")

# Normalisasi data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Hyperparameter tuning untuk SVM
svm_params = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf', 'linear']
}
svm_grid = GridSearchCV(SVC(), svm_params, refit=True, verbose=2, cv=5)
svm_grid.fit(X_train, y_train)
print("Best SVM Parameters:", svm_grid.best_params_)
svm_best = svm_grid.best_estimator_
y_pred_svm = svm_best.predict(X_test)
print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm))
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))

# Hyperparameter tuning untuk KNN
knn_params = {
    'n_neighbors': range(1, 31),
    'weights': ['uniform', 'distance']
}
knn_grid = GridSearchCV(KNeighborsClassifier(), knn_params, refit=True, verbose=2, cv=5)
knn_grid.fit(X_train, y_train)
print("Best KNN Parameters:", knn_grid.best_params_)
knn_best = knn_grid.best_estimator_
y_pred_knn = knn_best.predict(X_test)
print("KNN Classification Report:")
print(classification_report(y_test, y_pred_knn))
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
```

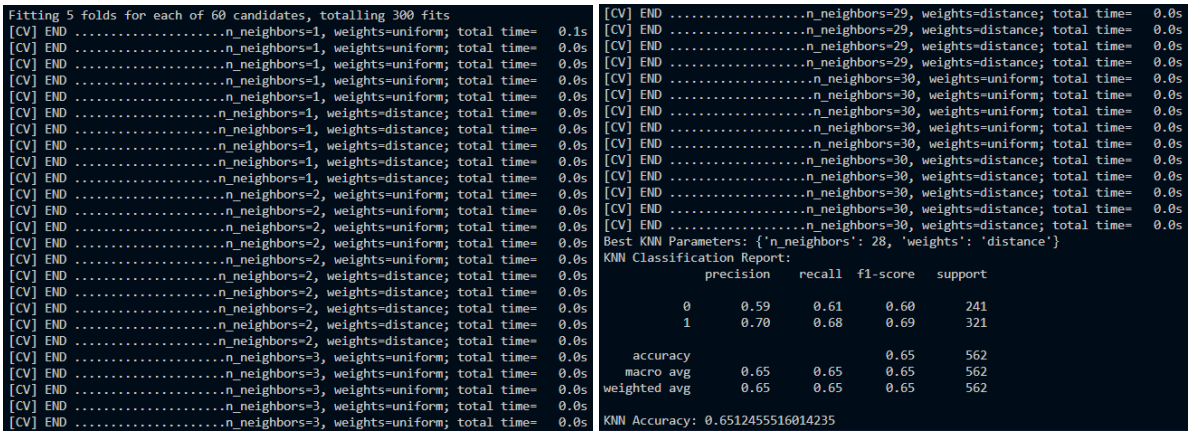
Gambar D.c.1. Scripts create_feature.py

Kode dalam `classification.py` bertujuan untuk melatih dan mengevaluasi model klasifikasi menggunakan dua algoritma, yaitu Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN).

1. Fungsi `load_data` digunakan untuk membaca data dari file CSV. Data dibaca menggunakan `pandas` dan dipisahkan menjadi fitur (X) dan label (y). Fungsi ini membaca data dari file `training_features.csv` dan `testing_features.csv` dan mengembalikan dua array: `X_train`, `y_train` untuk data latih dan `X_test`, `y_test` untuk data uji.
2. Untuk memastikan bahwa semua fitur berada pada skala yang sama, data fitur dinormalisasi menggunakan `StandardScaler` dari `sklearn.preprocessing`. `StandardScaler` menyesuaikan skala data latih (`X_train`) guna memastikan setiap fitur memiliki rata-rata 0 dan standar deviasi 1, yang penting bagi algoritma seperti SVM dan KNN yang sensitif terhadap skala data. Kemudian diterapkan transformasi yang sama ke data uji (`X_test`).
3. Model SVM (`SVC`) dilatih dan dioptimalkan menggunakan `GridSearchCV` untuk menemukan kombinasi parameter terbaik. Parameter yang dicoba meliputi 'C' (regulasi), 'gamma' (koefisien kernel), dan 'kernel' (jenis kernel). Grid search dilakukan dengan 5-fold cross-validation (`cv=5`). Model terbaik dipilih berdasarkan hasil tuning, dan prediksi dilakukan pada data uji menggunakan model terbaik (`svm_best`). Evaluasi performa model dilakukan menggunakan `classification_report` dan `accuracy_score`.
4. Model KNN (`KNeighborsClassifier`) juga dioptimalkan menggunakan `GridSearchCV`. Parameter yang dicoba meliputi 'n_neighbors' (jumlah tetangga terdekat) dan 'weights' (bobot voting). Sama seperti pada SVM, grid search dilakukan dengan 5-fold cross-validation. Model terbaik dipilih berdasarkan hasil tuning, dan prediksi dilakukan pada data uji menggunakan model terbaik (`knn_best`). Evaluasi performa model dilakukan dengan `classification_report` dan `accuracy_score`.



Gambar D.c.2 Tampilan nilai akurasi model SVM



Gambar D.c.3 Tampilan nilai akurasi model KNN

E. Hasil

Dari output yang dihasilkan pada gambar , terlihat bahwa terdapat dua tahap utama dalam pengklasifikasian menggunakan metode Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN). Tahap pertama adalah proses hyperparameter tuning di mana model SVM dan KNN dioptimalkan. Untuk SVM, parameter yang diuji meliputi 'C' (penalty parameter), 'gamma' (koefisien kernel), dan 'kernel' (jenis kernel). Sedangkan untuk KNN, parameter-parameter yang diuji meliputi 'n_neighbors' (jumlah tetangga terdekat) dan 'weights' (bobot voting). Proses ini dilakukan dengan menggunakan Grid Search Cross Validation dengan 5-fold cross-validation, yang berarti data dibagi menjadi 5 subset dan model diuji menggunakan 4 subset sebagai data pelatihan dan 1 subset sebagai data validasi.

Setelah proses hyperparameter tuning selesai, hasil terbaik dari masing-masing model SVM dan KNN ditampilkan. Untuk SVM, parameter terbaik yang ditemukan adalah C=100, gamma=0.001, dan kernel='rbf'. Sedangkan untuk KNN, parameter terbaik yang ditemukan adalah n_neighbors=28 dan weights='distance'. Kedua model kemudian digunakan untuk melakukan prediksi pada data uji, dan hasil prediksi tersebut dievaluasi menggunakan metrik-metrik klasifikasi seperti precision, recall, dan f1-score untuk masing-masing kelas (berita dan sepakbola). Selain itu, juga dilaporkan nilai akurasi dari masing-masing model.

Dari hasil evaluasi yang diberikan, terlihat bahwa kinerja model SVM dan KNN cukup baik, meskipun SVM memiliki akurasi yang sedikit lebih tinggi dibandingkan KNN. Model SVM mencapai akurasi sebesar 76.33%, sedangkan model KNN mencapai akurasi sebesar 65.12%. Precision, recall, dan f1-score untuk kedua kelas pada model SVM juga lebih seimbang daripada pada model KNN. Dengan demikian, hasil ini menunjukkan bahwa model SVM lebih unggul dalam memprediksi kelas-kelas yang ada dalam dataset tersebut dibandingkan dengan model KNN.

F. Kesimpulan

1. Proses ekstraksi fitur dari teks dilakukan menggunakan skrip Create-Feature.py. Fitur-fitur diekstraksi dari judul dan konten artikel dengan memperhitungkan kamus-kamus yang telah dibentuk sebelumnya. Fitur-fitur ini kemudian digunakan sebagai masukan untuk algoritma klasifikasi. Tahapan-tahapan dalam skrip ini meliputi membersihkan teks, menghitung jumlah kata, dan menghitung fitur-fitur berdasarkan kamus-kamus yang telah dibangun sebelumnya.
2. Model klasifikasi menggunakan dua algoritma, yaitu Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN). Proses ini dilakukan dalam skrip Classification.py. Tahapan-tahapan dalam skrip ini meliputi pembacaan data, normalisasi data, hyperparameter tuning menggunakan GridSearchCV, pelatihan model dengan data latih, prediksi pada data uji, dan evaluasi performa model menggunakan metrik-metrik klasifikasi seperti precision, recall, f1-score, dan akurasi.
3. Untuk memperoleh model terbaik, kedua algoritma klasifikasi, SVM dan KNN, dioptimalkan menggunakan teknik Grid Search Cross Validation dengan 5-fold cross-validation. Proses ini membantu menemukan kombinasi parameter terbaik untuk masing-masing algoritma, seperti 'C', 'gamma', dan 'kernel' untuk SVM, serta 'n_neighbors' dan 'weights' untuk KNN. Hasil dari proses optimasi ini memungkinkan untuk memilih model dengan performa terbaik.
4. Dari hasil evaluasi yang diberikan, terlihat bahwa model SVM (0.7633451957295374 atau 76.33%) memiliki performa yang lebih baik dibandingkan dengan model KNN (0.6512455516014235 atau 65.12%) dalam memprediksi kelas-kelas dalam dataset tersebut. . Untuk SVM, parameter terbaik yang ditemukan adalah C=100, gamma=0.001, dan kernel='rbf'. Sedangkan untuk KNN, parameter terbaik yang ditemukan adalah n_neighbors=28 dan weights='distance'.