*Laporan Mata Kuliah Teks dan Web Mining*

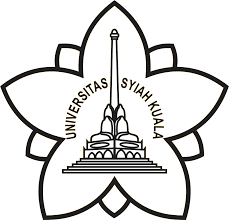
TUGAS 6: MEMBANGUN FITUR DAN MELAKUKAN KALSIFIKASI DENGAN METODE SVM DAN KNN

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

1. **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.

1. **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.

1. 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.

1. 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.

1. **Penjelasan Umum**
2. 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.

1. 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.

1. 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.

1. **Penjelasan Scripts**
2. **Remove-one-gram.py, remove-two-gram.py, dan remove-three-gram.py**



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



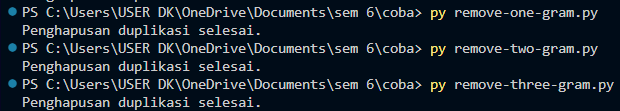
**Gambar D.a.2.** Scripts remove-two-gram.py



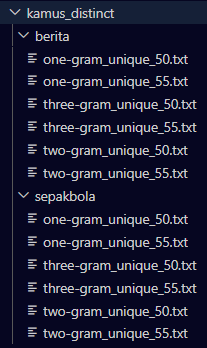
**Gambar D.a.3.** Scripts remove-three-gram.py

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.



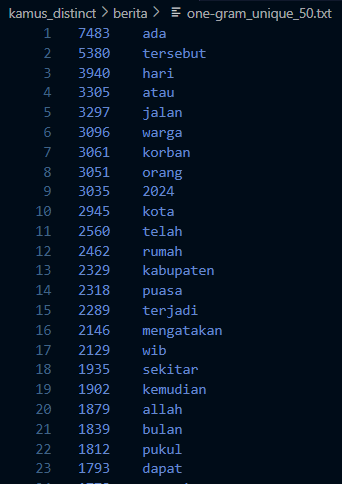
**Gambar D.a.4.** Output sukses membangun kamus one-gram, two-gram, dan three-gram



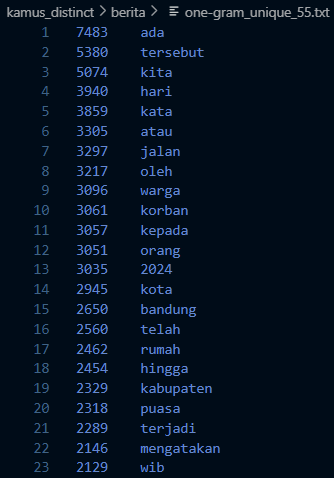
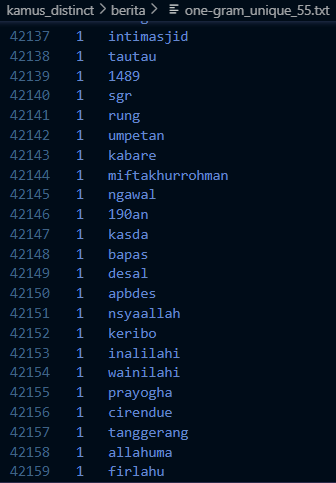
**Gambar D.a.5.** Directory yang dihasilkan

Output yang dihasilkan:

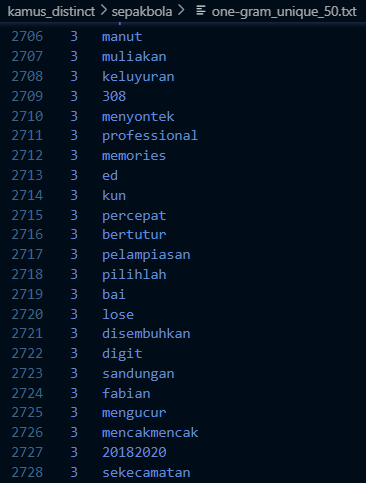
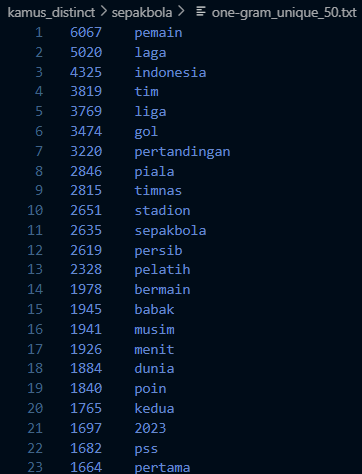
1. Remove-one-gram.py

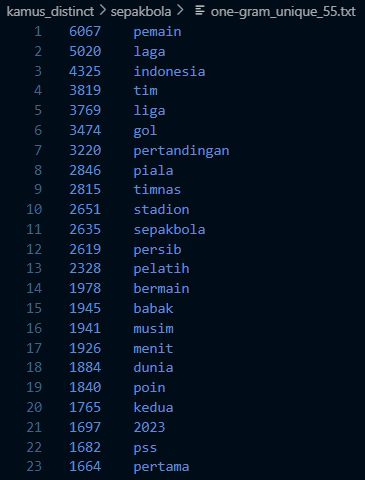
**Gambar D.a.6.** Hasil kamus one-gram\_unique\_50.txt kategori berita

**Gambar D.a.7.** Hasil kamus one-gram\_unique\_55.txt kategori berita

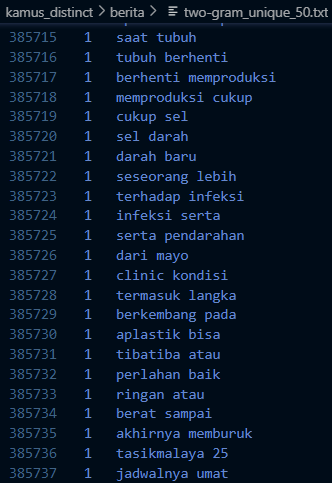


**Gambar D.a.8.** Hasil kamus one-gram\_unique\_50.txt kategori sepakbola



**Gambar D.a.9.** Hasil kamus one-gram\_unique\_55.txt kategori sepakbola

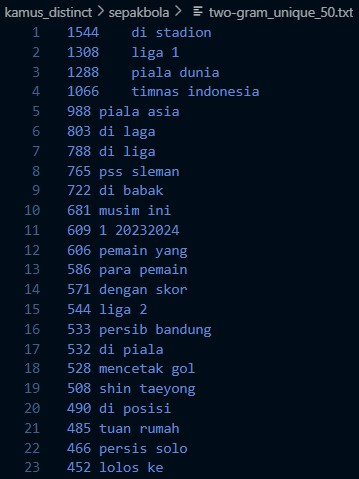
1. Remove-two-gram.py



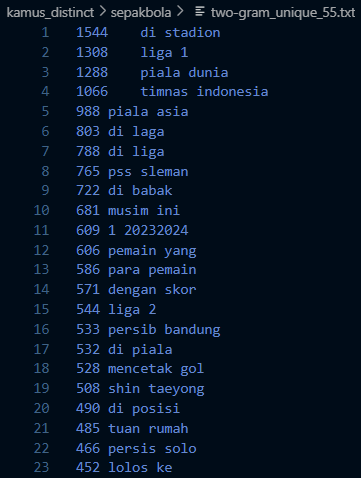
**Gambar D.a.10.** Hasil kamus two-gram\_unique\_50.txt kategori berita



**Gambar D.a.11.** Hasil kamus two-gram\_unique\_55.txt kategori berita

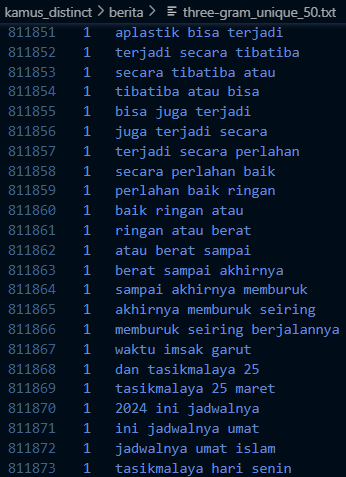
 

**Gambar D.a.12.** Hasil kamus two-gram\_unique\_50.txt kategori sepakbola

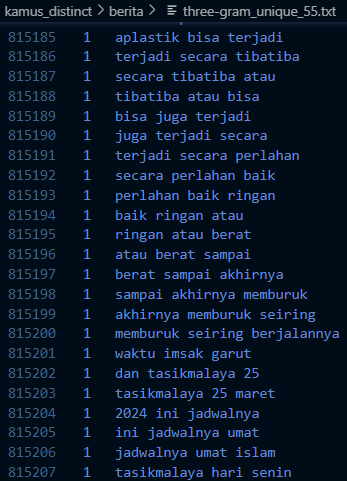
 

**Gambar D.a.13.** Hasil kamus two-gram\_unique\_55.txt kategori sepakbola

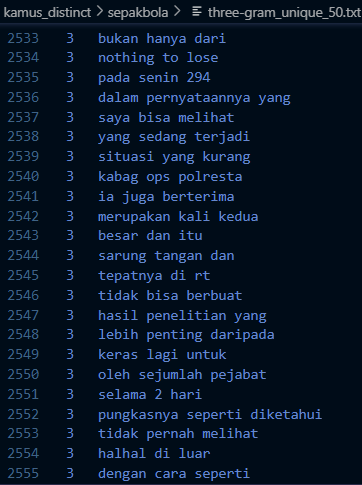
1. Remove-three-gram.py

**Gambar D.a.14.** Hasil kamus three-gram\_unique\_50.txt kategori berita



**Gambar D.a.15.** Hasil kamus three-gram\_unique\_55.txt kategori berita

**Gambar D.a.16.** Hasil kamus three-gram\_unique\_50.txt kategori sepakbola



**Gambar D.a.17.** Hasil kamus three-gram\_unique\_55.txt kategori sepakbola

1. **Create\_feature.py**

****

****

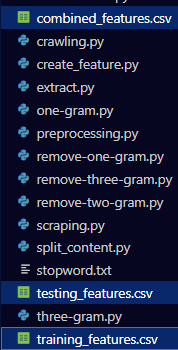
**Gambar D.b.1.** Scripts create\_feature.py

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.

****

**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

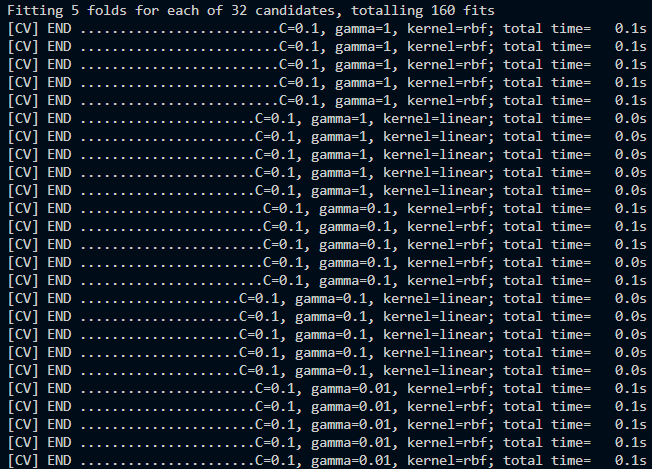
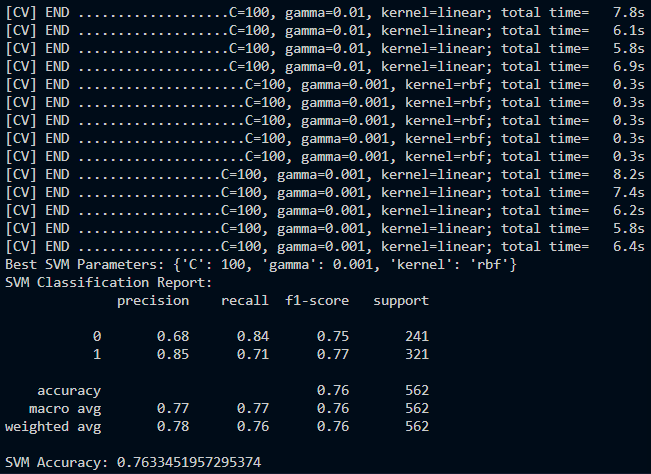
1. **Classification.py**

****

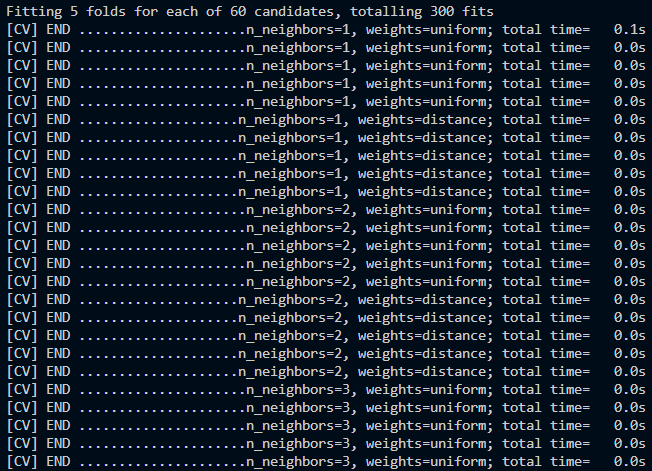
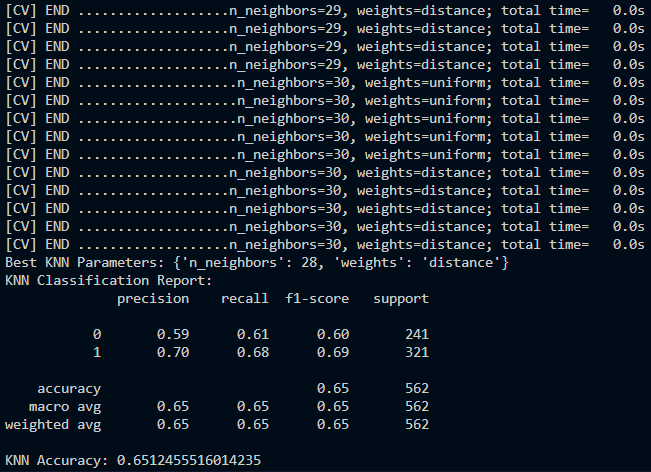
**Gambar D.c.1.** Scripts create\_feature.py

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

1. **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.

1. **Kesimpulan**
2. 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.
3. 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.
4. 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.
5. 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'.