**LAPORAN PRAKTIKUM**

**BIG DATA ANALYTICS**

**PERTEMUAN – 14**



**DISUSUN OLEH :**

**NAMA : Ridhwan Cahyadi**

**NIM : 215410028**

**PRODI : Informatika**

**PROGRAM STUDI SISTEM INFORMASI**

**PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA**

**YOGYAKARTA**

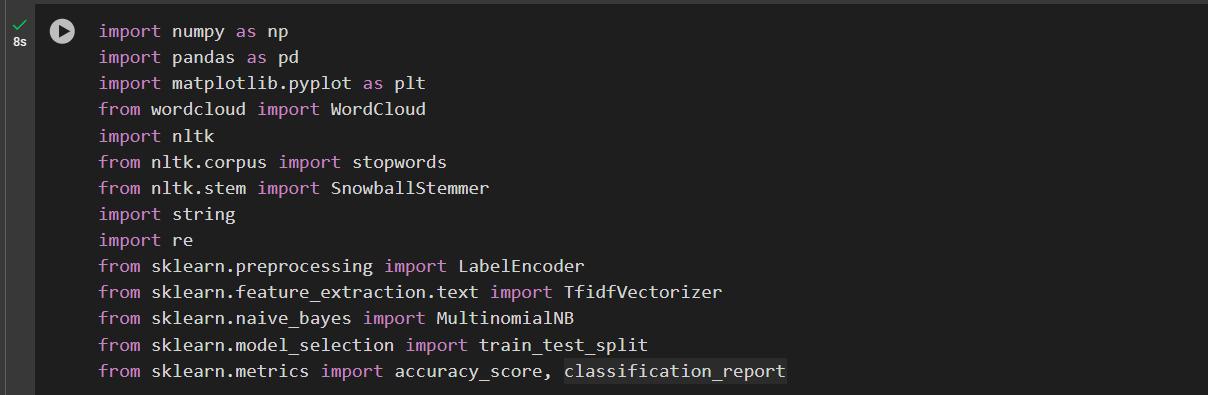
**2024**

**Modul 14 Studi Kasus**

Pada modul ini mahasiswa membuat proyek Big Data Analytic dengan ketentuan sebagai berikut:

1. Membuat analisis sentiment dengan data yang telah diberikan
2. Implementasikan dengan metode naïve bayes.
3. Kemudian berikan kesimpulan dari hasil pengolahan data yang anda lakukan

**Pembahasan:**

**Import library yang dibutuhkan**

**Pembahasan:**

Kode di atas mengimpor berbagai library dan modul yang diperlukan untuk analisis sentimen, sebuah tugas umum dalam pengolahan bahasa alami (NLP). Library yang diimpor termasuk NumPy dan Pandas untuk manipulasi data, Matplotlib untuk visualisasi data, WordCloud untuk menghasilkan awan kata, dan NLTK (Natural Language Toolkit) untuk praproses teks. Selain itu, kode juga mengimpor modul spesifik dari NLTK, seperti stopwords, SnowballStemmer, dan string, untuk melakukan tugas seperti menghapus kata-kata stop, stemming, dan menghapus tanda baca. Kode juga mengimpor library scikit-learn, termasuk LabelEncoder, TfidfVectorizer, dan MultinomialNB, yang akan digunakan untuk mengkodekan label, mengkonversi data teks menjadi fitur numerik, dan melatih kelasifikasi Naive Bayes, masing-masing.

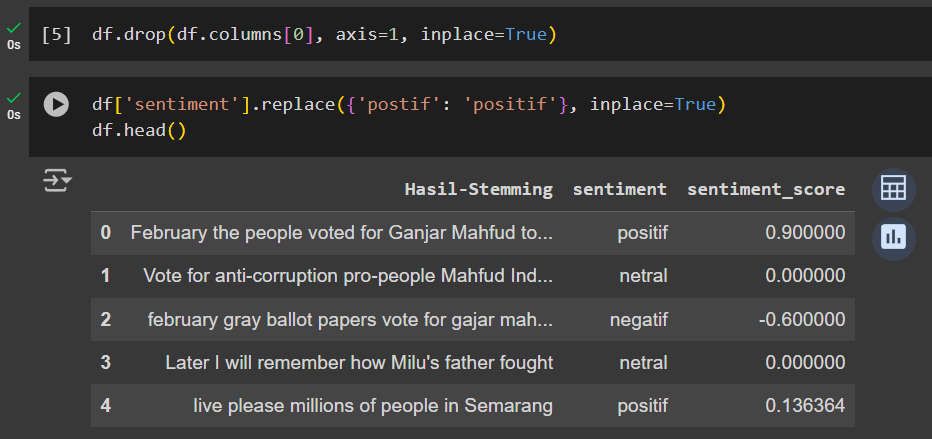
Dalam konteks analisis sentimen, library dan modul ini akan digunakan untuk menganalisis data teks untuk menentukan apakah mereka postif, negatif ataukah netral. Modul WordCloud dapat digunakan untuk menghasilkan representasi visual dari kata-kata yang paling umum digunakan dalam ulasan, sedangkan modul NLTK akan digunakan untuk praproses data teks dengan menghapus kata-kata stop, stemming, dan menghapus tanda baca. Library scikit-learn kemudian akan digunakan untuk melatih kelasifikasi untuk memprediksi sentimen ulasan baru yang belum terlihat.

**Lihat DataFrame**

**Pembahasan:**

Perintah di atas digunakan untuk membaca dan menampilkan data dari file CSV yang berlokasi di Google Drive. File CSV tersebut berisi data tentang sentimen dari beberapa kalimat, yang telah di-stemming dan diberi label sentimen serta skor sentimen. Perintah **pd.read\_csv(dataset)** digunakan untuk membaca file CSV dan menyimpannya dalam variabel **df**. Kemudian, perintah **df.head()** digunakan untuk menampilkan beberapa baris pertama dari data tersebut, yang dalam hal ini menampilkan 5 baris pertama.

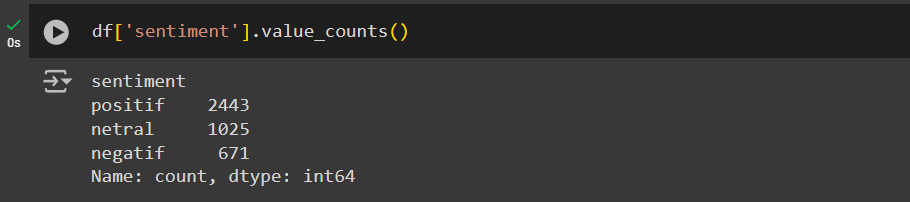
Dalam output yang dihasilkan, kita dapat melihat bahwa data tersebut memiliki empat kolom, yaitu **Unnamed: 0**, **Hasil-Stemming**, **sentiment**, dan **sentiment\_score**. Kolom **Hasil-Stemming** berisi kalimat yang telah di-stemming, kolom **sentiment** berisi label sentimen dari kalimat tersebut (positif, netral, atau negatif), dan kolom **sentiment\_score** berisi skor sentimen dari kalimat tersebut. Dengan menampilkan data ini, kita dapat memahami struktur dan isi dari data yang akan kita analisis.

**Hapus Kolom Pertama Dan Ubah Sentiment Postif Menjadi Positif**

**Pembahasan:**

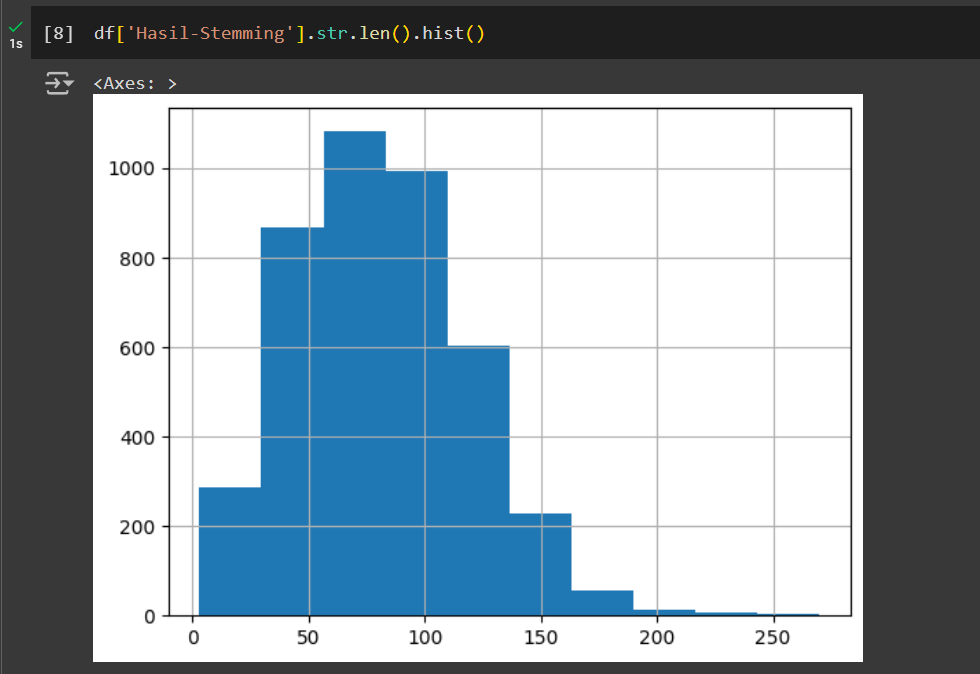
Perintah di atas digunakan untuk melakukan beberapa perubahan pada data yang telah dibaca sebelumnya.Pertama, perintah df.drop(df.columns[0], axis=1, inplace=True) digunakan untuk menghapus kolom pertama dari data, yang bernama Unnamed: 0. Kolom ini tidak diperlukan dalam analisis sentimen, sehingga dihapus untuk memudahkan proses analisis.

Kemudian, perintah df['sentiment'].replace({'postif': 'positif'}, inplace=True) digunakan untuk mengganti nilai postif menjadi positif pada kolom sentiment. Hal ini dilakukan karena terdapat kesalahan penulisan pada label sentimen, yaitu postif yang seharusnya ditulis positif. Setelah melakukan perubahan ini, perintah df.head() digunakan untuk menampilkan beberapa baris pertama dari data yang telah diubah. Dalam output yang dihasilkan, kita dapat melihat bahwa kolom Unnamed:0 telah dihapus dan label sentimen postif telah diganti menjadi positif.

**Lihat Distribusi Nilai Kolom Sentiment**

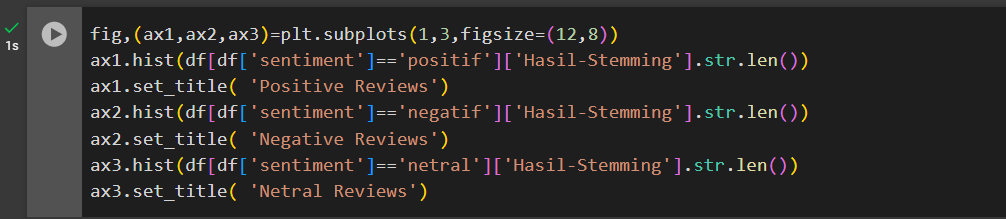
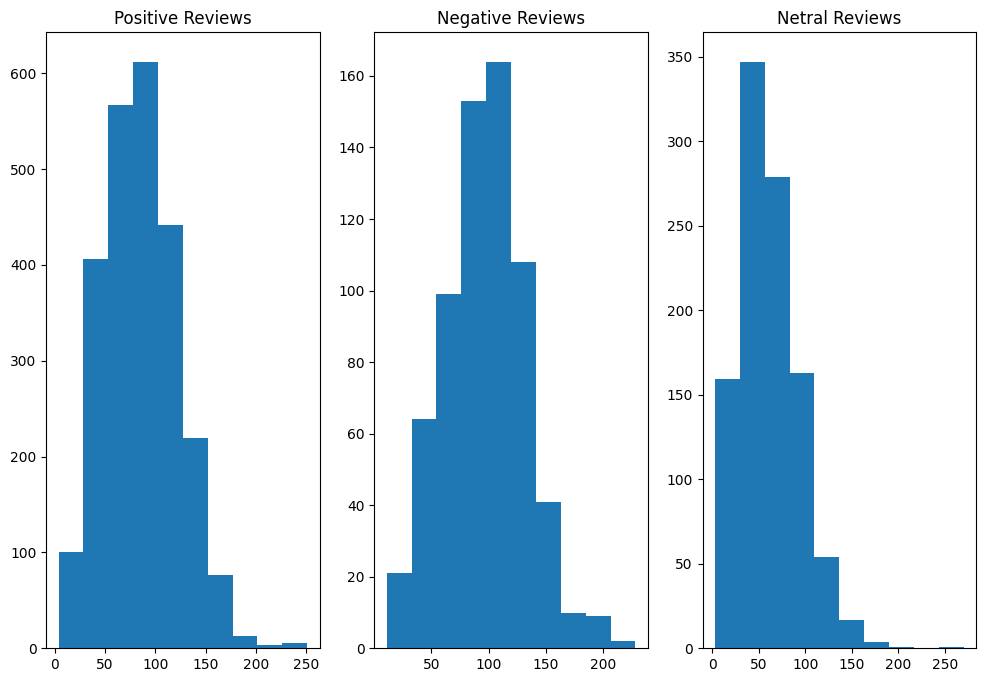
**Pembahasan:**

Perintah di atas digunakan untuk mengetahui distribusi nilai pada kolom **sentiment** dalam data. Perintah **df['sentiment'].value\_counts()** digunakan untuk menghitung jumlah kemunculan setiap nilai unik pada kolom **sentiment**. Hasilnya menunjukkan bahwa nilai **positif** muncul sebanyak 2443 kali, nilai **netral** muncul sebanyak 1025 kali, dan nilai **negatif** muncul sebanyak 671 kali.

**Pembahasan:**

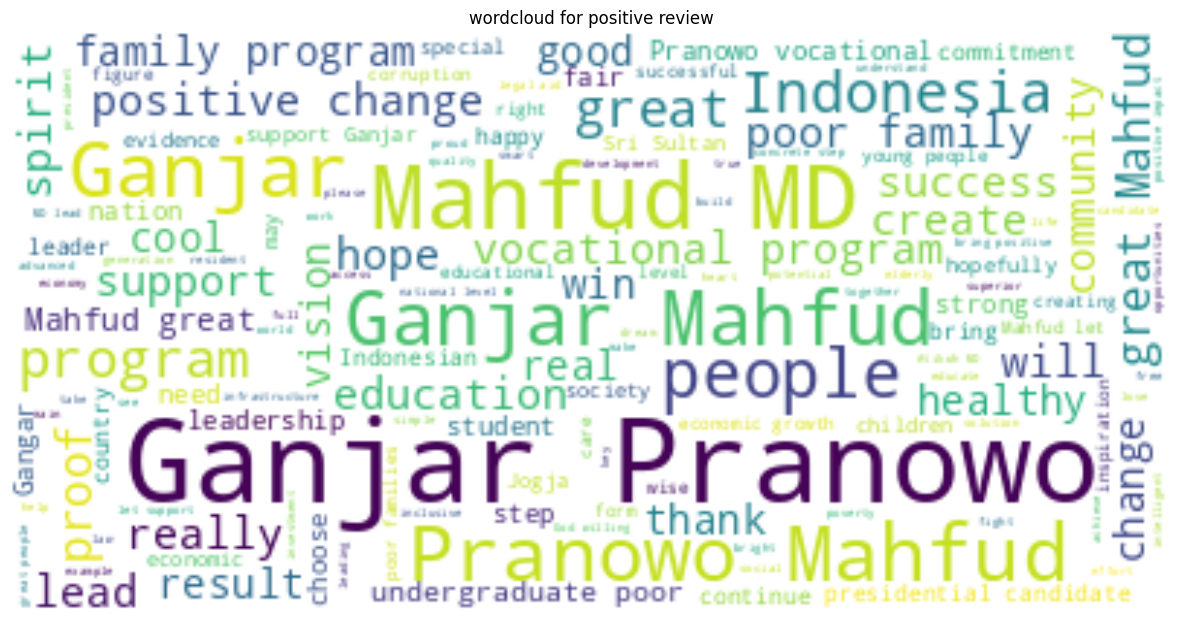
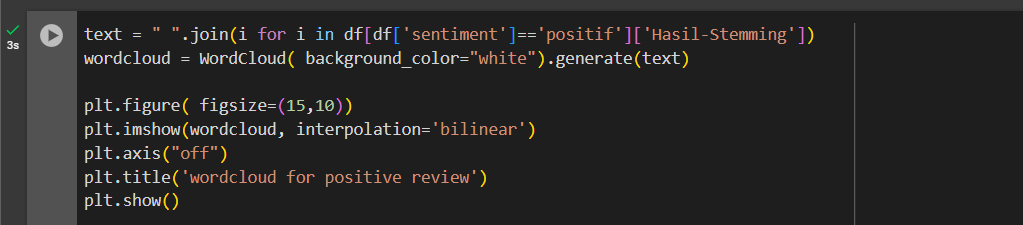
Perintah df['Hasil-Stemming'].str.len().hist() merupakan perintah untuk menampilkan histogram dari panjang karakter pada kolom 'Hasil-Stemming' dalam DataFrame df. Perintah ini bekerja dengan mengambil kolom 'Hasil-Stemming', kemudian mengambil panjang setiap string dalam kolom tersebut menggunakan .str.len(). Hasilnya kemudian diplot sebagai histogram menggunakan .hist().

Hasil grafik menunjukkan distribusi frekuensi dari panjang karakter pada kolom 'Hasil-Stemming'. Grafik menunjukkan bahwa sebagian besar teks pada kolom 'Hasil-Stemming' memiliki panjang karakter antara 50 hingga 100 karakter. Sebagian kecil teks memiliki panjang karakter lebih dari 100 karakter dan bahkan lebih sedikit lagi yang memiliki panjang karakter lebih dari 150 karakter. Distribusi ini menunjukkan bahwa sebagian besar teks pada kolom 'Hasil-Stemming' relatif pendek.

**Lihat Distribusi Nilai Masing Masing Sentiment**

**Pembahasan:**

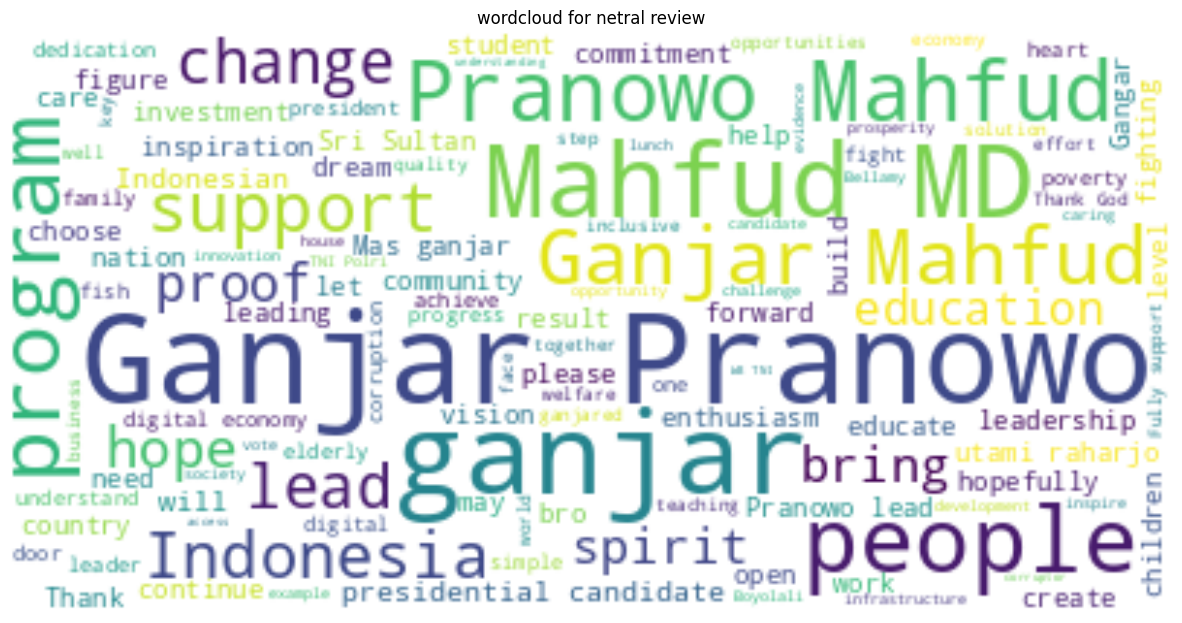
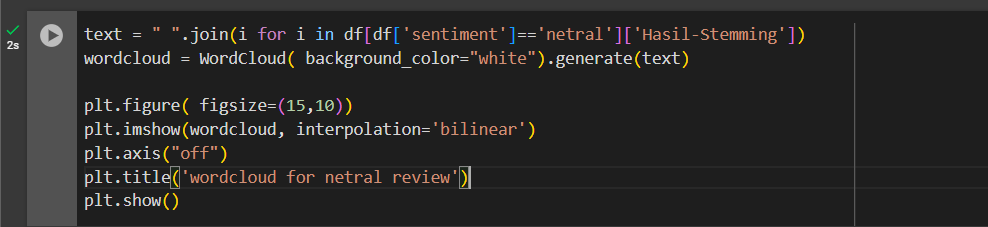
Perintah tersebut digunakan untuk membuat grafik histogram yang menunjukkan distribusi panjang teks pada setiap sentimen (positif, negatif, dan netral). Hasil dari grafik tersebut menunjukkan bahwa panjang teks pada sentimen positif cenderung lebih pendek, sedangkan panjang teks pada sentimen negatif dan netral cenderung lebih panjang. Hal ini dapat diinterpretasikan bahwa orang lebih cenderung untuk menulis komentar yang lebih singkat dan langsung ketika mereka memiliki sentimen positif, sementara mereka lebih cenderung untuk menulis komentar yang lebih panjang dan detail ketika mereka memiliki sentimen negatif atau netral.

**Lihat Representasi Kata Untuk Se**n**timent Positif**

**Pembahasan:**

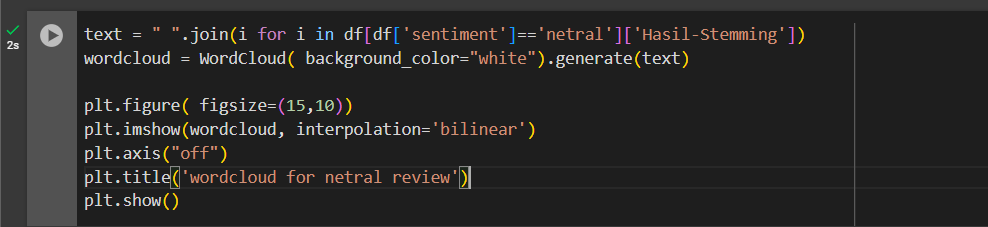
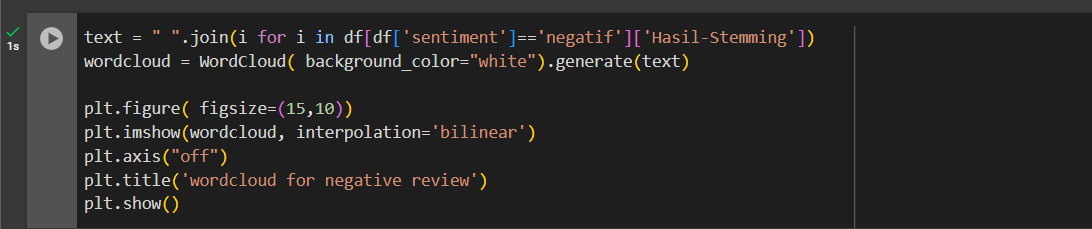
Perintah pada kode diatas adalah untuk membuat wordcloud dari teks yang berisi kata-kata yang telah di-stemming (disederhanakan) dari kolom 'Hasil-Stemming' pada dataframe 'df'. Kata-kata yang di-stemming ini hanya diambil dari baris yang memiliki sentiment 'positif'. Wordcloud yang dihasilkan akan menampilkan kata-kata yang paling sering muncul dalam teks, dan ukuran kata akan menunjukkan frekuensi kemunculannya.

Hasil grafik diatas adalah wordcloud yang menampilkan kata-kata yang paling sering muncul dalam teks hasil stemming dari kolom 'Hasil-Stemming' pada dataframe 'df' dengan sentiment 'positif'. Kata-kata yang paling sering muncul adalah 'Ganjar', 'Mahfud', 'Pranowo', 'program', 'people', 'support', 'good', 'great', 'Indonesia', 'family', dan 'success'. Ini menunjukkan bahwa teks tersebut secara umum memiliki sentiment positif dan memuji Ganjar, Mahfud, Pranowo, serta program, people, dan success yang mereka lakukan.

**Lihat Representasi Kata Untuk Se**n**timent Netral**

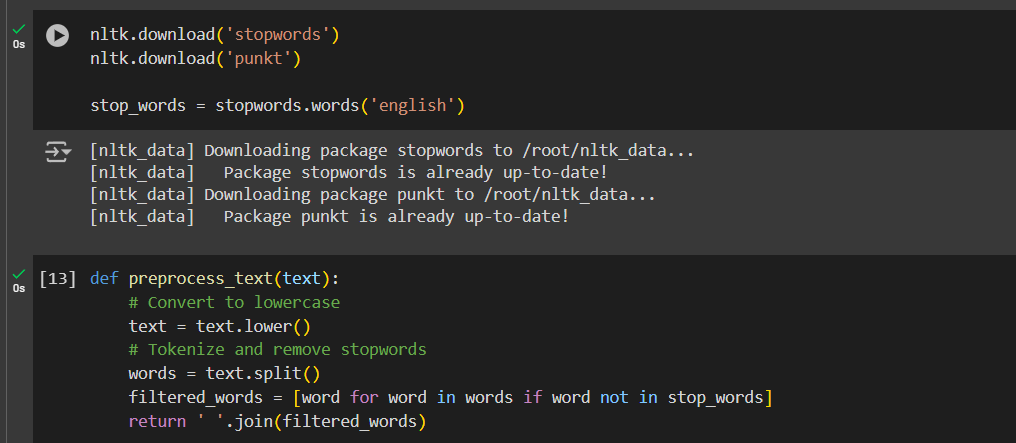
**Pembahasan:**

Seperti halnya untuk sentiment positif, kita lihat representasi kata yang paling muncul untuk sentiment netral. Terlihat hasil yang diperoleh menunjukkan kata-kata seperti "Pranowo", "Ganjar", "Mahfud", dan "Indonesia" muncul dengan frekuensi yang tinggi. Hal ini menunjukkan bahwa kata-kata tersebut merupakan kata-kata yang sering digunakan dalam teks yang berkaitan dengan sentimen netral.

**Lihat Representasi Kata Untuk Se**n**timent Negatif** 

**Pembahasan:**

Terakhir, kita lihat representasi sentiment negatif. Hasil dari visualisasi menunjukkan kata "poor" dan "family" merupakan kata yang paling sering muncul dalam teks sentimen negatif. Kata-kata lain yang muncul seperti "program", "student", "university", dan "indonesia" juga menunjukkan bahwa sentimen negatif yang diungkapkan terkait dengan pendidikan dan kondisi sosial di Indonesia.

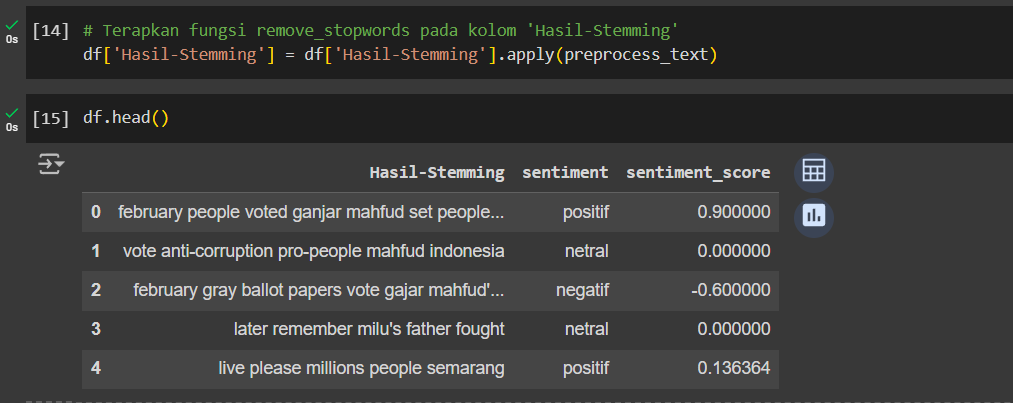
**Preprocessing Data**

**Pembahasan:**

Selanjutnya kita lakukan preprocessing data dengan dua hal yaitu mengubah teks menjadi huruf kecil dan menghilangkan stopwords. Kita dapat gunakan library NLTK yaitu ‘stopwords’ dan ‘punkt’. 'Stopwords' adalah kumpulan kata-kata yang tidak memiliki makna dalam suatu kalimat, seperti "the", "and", "a", dll. Sedangkan 'punkt' adalah paket untuk tokenisasi, yaitu memecah kalimat menjadi kata-kata individu.

Fungsi **preprocess\_text()** menerima input berupa teks dan mengembalikan teks yang telah diproses. Langkah pertama dalam fungsi ini adalah mengubah teks menjadi huruf kecil menggunakan metode lower(). Hal ini dilakukan untuk menghindari perbedaan dalam penggunaan huruf besar dan kecil dalam analisis teks.

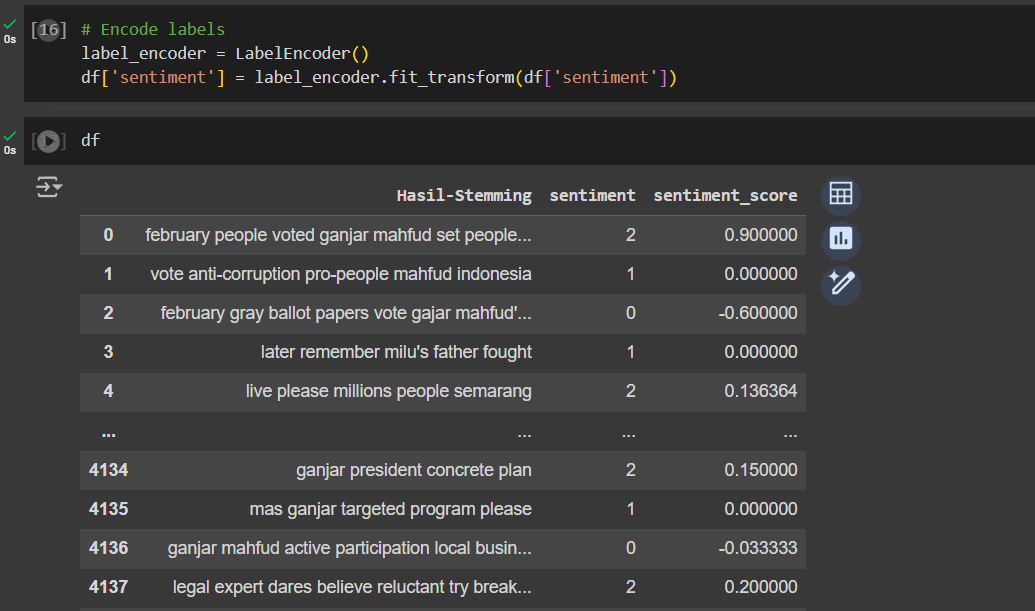
Langkah kedua dalam fungsi ini adalah menghilangkan stopwords dari teks. Stopwords adalah kata-kata yang tidak memiliki makna dalam suatu kalimat, seperti "the", "and", "a", dll. Untuk menghilangkan stopwords, fungsi ini menggunakan list stop\_words yang telah diunduh dari NLTK. Fungsi ini memecah teks menjadi kata-kata individu menggunakan metode split(), kemudian memfilter kata-kata tersebut menggunakan list comprehension untuk menghilangkan stopwords. Hasilnya adalah teks yang telah diproses dan siap untuk dianalisis lebih lanjut.

**Lihat Data Kembali Hasil Pengolahan**

**Pembahasan:**

Perintah di atas untuk menerapkan fungsi preprocess\_text pada kolom 'Hasil-Stemming' dalam dataframe df. Fungsi preprocess\_text ini telah dijelaskan sebelumnya, yaitu untuk mengubah teks menjadi huruf kecil dan menghilangkan stopwords. Dengan menerapkan fungsi ini pada kolom 'Hasil-Stemming', maka teks-teks dalam kolom tersebut akan diproses untuk menghilangkan stopwords dan diubah menjadi huruf kecil.

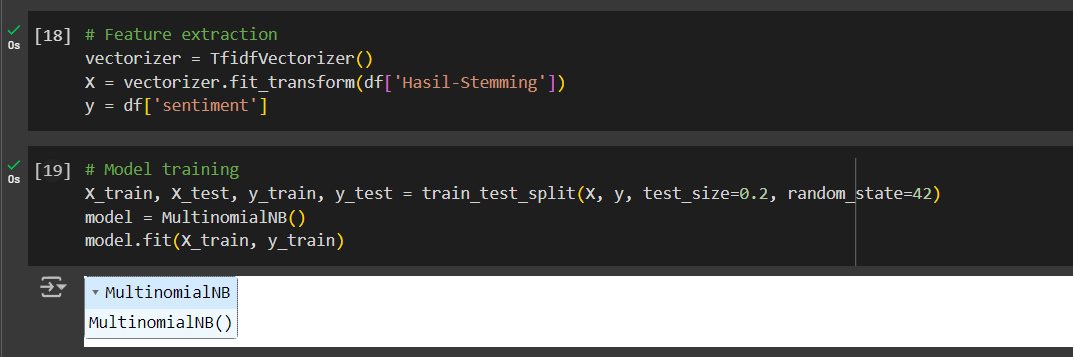
Setelah menerapkan fungsi preprocess\_text, maka dataframe df akan diupdate dengan kolom 'Hasil-Stemming' yang telah diproses. Kemudian, perintah df.head() digunakan untuk menampilkan beberapa baris pertama dari dataframe df untuk memastikan bahwa proses pengolahan teks telah berhasil.

**Encode Labels**

**Pembahasan:**

Perintah di atas adalah untuk mengkodekan label pada kolom 'sentiment' dalam dataframe df. Label encoder adalah sebuah metode yang digunakan untuk mengubah label kategorik menjadi nilai numerik. Dalam kasus ini, label encoder digunakan untuk mengubah label 'positif', 'netral', dan 'negatif' pada kolom 'entiment' menjadi nilai numerik.

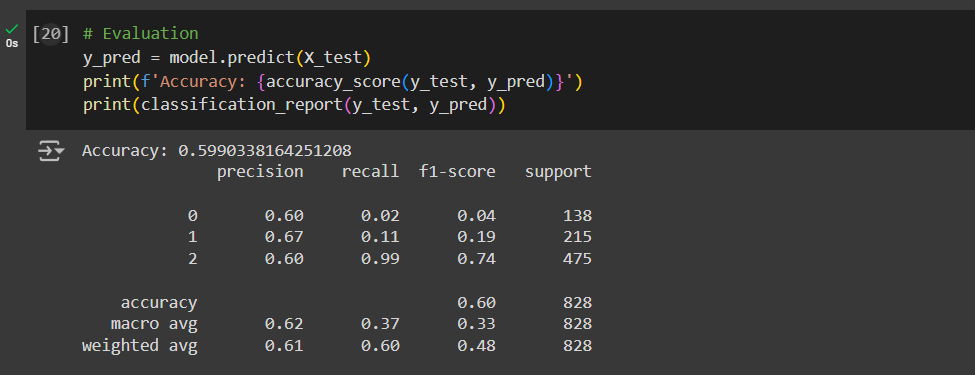
Dengan menggunakan perintah label\_encoder.fit\_transform(df['sentiment']), maka label encoder akan mengidentifikasi label unik pada kolom 'entiment' dan mengubahnya menjadi nilai numerik. Nilai numerik ini kemudian akan disimpan kembali pada kolom 'entiment' dalam dataframe df. Hasilnya adalah kolom 'entiment' yang telah dikodekan menjadi nilai numerik, yaitu 0 untuk 'negatif', 1 untuk 'netral', dan 2 untuk 'positif'.

**Ekstraksi Fitur dan Training Model**

**Pembahasan:**

Perintah di atas adalah untuk melakukan ekstraksi fitur dan pelatihan model pada dataset. Ekstraksi fitur dilakukan menggunakan TfidfVectorizer, yang mengubah teks dalam kolom 'Hasil-Stemming' menjadi vektor numerik yang dapat diproses oleh model. Vektor ini kemudian disimpan dalam variabel X. Label sentiment yang telah dikodekan sebelumnya disimpan dalam variabel y.

Pelatihan model dilakukan menggunakan MultinomialNB, yang merupakan sebuah algoritma klasifikasi Naive Bayes. Sebelum pelatihan, dataset dibagi menjadi dua bagian, yaitu dataset pelatihan (X\_train dan y\_train) dan dataset pengujian (X\_test dan y\_test). Dataset pelatihan digunakan untuk melatih model, sedangkan dataset pengujian digunakan untuk mengevaluasi kinerja model. Setelah pelatihan, model siap digunakan untuk melakukan klasifikasi sentiment pada teks baru.

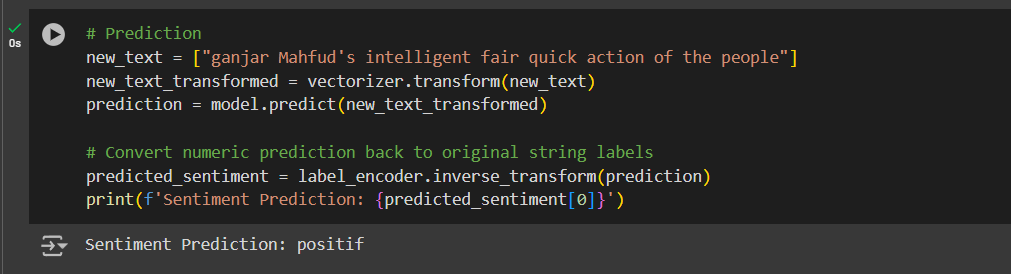
**Evaluasi**

**Pembahasan:**

Perintah di atas adalah untuk mengevaluasi kinerja model yang telah dilatih sebelumnya. Evaluasi dilakukan dengan menggunakan dataset pengujian (X\_test) untuk memprediksi label sentiment, dan hasil prediksi disimpan dalam variabel y\_pred. Kemudian, akurasi model dihitung menggunakan accuracy\_score dan dicetak ke layar. Akurasi ini menunjukkan proporsi data yang diprediksi dengan benar oleh model.

Hasil output yang diperoleh menunjukkan bahwa akurasi model adalah 0.599 atau 59.9%. Artinya, model ini dapat memprediksi label sentiment dengan benar pada 59.9% dari total data pengujian. Selain itu, laporan klasifikasi juga menunjukkan bahwa model ini memiliki presisi, recall, dan f1-score yang berbeda-beda untuk setiap kelas sentiment.

Berdasarkan akurasi tersebut, dapat dikatakan bahwa model ini tidak terlalu bagus. Akurasi 59.9% masih relatif rendah, artinya model ini masih membuat banyak kesalahan dalam memprediksi label sentiment. Idealnya, akurasi model harus di atas 80% atau 90% untuk dapat dikatakan sebagai model yang baik. Oleh karena itu, masih perlu dilakukan pengembangan dan penyesuaian model untuk meningkatkan akurasinya.

**Prediksi**

**Pembahasan:**

Perintah di atas adalah untuk membuat prediksi sentiment pada teks baru menggunakan model yang telah dilatih sebelumnya. Teks baru tersebut adalah "ganjar Mahfud's intelligent fair quick action of the people". Pertama, teks ini diubah menjadi bentuk vektor menggunakan vectorizer.transform() agar dapat diproses oleh model.

Kemudian, model membuat prediksi sentiment pada teks tersebut dan hasil prediksi disimpan dalam variabel prediction. Hasil prediksi ini masih dalam bentuk numerik, sehingga perlu diubah kembali menjadi label sentiment asli menggunakan label\_encoder.inverse\_transform(). Hasil akhirnya adalah prediksi sentiment "positif", yang berarti model memprediksi bahwa teks tersebut memiliki sentiment yang positif.

**KESIMPULAN**

Berdasarkan analisis yang telah dilakukan, kita dapat menyimpulkan bahwa model yang kita buat memiliki akurasi sekitar 59.9%. Akurasi ini masih relatif rendah dan perlu ditingkatkan agar model dapat memprediksi sentiment dengan lebih akurat. Meskipun demikian, model ini masih dapat memprediksi sentiment dengan cukup baik, seperti terlihat pada contoh teks "ganjar Mahfud's intelligent fair quick action of the people" yang diprediksi memiliki sentiment positif.

Kesimpulan lainnya adalah bahwa model ini dapat digunakan untuk menganalisis sentiment pada teks-teks yang berhubungan dengan Ganjar Pranowo dan Mahfud MD, dua tokoh yang terkait dengan pendidikan dan kemiskinan di Indonesia. Dengan demikian, model ini dapat membantu dalam menganalisis opini publik dan sentiment masyarakat terhadap isu-isu sosial dan politik di Indonesia. Namun, perlu diingat bahwa model ini masih perlu ditingkatkan dan disempurnakan agar dapat memberikan hasil yang lebih akurat dan reliable.