

LAPORAN UAS KECERDASAN BUATAN

Laporan ini dibuat untuk memenuhi Ujian Akhir Semester mata kuliah Kecerdasan Buatan



DISUSUN OLEH:

32230094 - Raffael Bernard Wijaya

32230110 - Chrisna Octavianus

32230112 - Fricilia Angelica

32230149 - Marwan Widyanto

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS BUNDA MULIA

2025

DAFTAR ISI

DAFTAR ISI.....	1
LINK GOOGLE DRIVE DAN COLAB.....	2
PENDAHULUAN.....	3
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan.....	4
1.4 Batasan Masalah.....	4
DATASET.....	6
2.1 Sumber Data.....	6
2.2 Deskripsi Data.....	6
2.3 Contoh Data.....	6
Exploratory Data Analysis (EDA).....	7
3.1 Distribusi Kelas.....	7
3.2 Visualisasi Data (Pie chart dari distribusi label).....	7
3.3 Analisis Distribusi Label dan Keseimbangan Data.....	9
PREPROCESSING DATA.....	10
4.1 Pembersihan Data.....	10
4.2 TF-IDF.....	11
4.3 Pembagian Data (Train:Test = 70:30).....	13
MODELING.....	15
5.1 Pemilihan Algoritma.....	15
5.2 Pelatihan Model.....	15
5.3 Parameter yang Digunakan.....	15
EVALUASI MODEL.....	18
6.1 Akurasi Model.....	18
6.2 Confusion Matrix.....	19
6.3 Precision, Recall, F1-Score.....	20
6.4 Analisis Hasil Evaluasi.....	20
KESIMPULAN.....	22

LINK GOOGLE DRIVE DAN COLAB

 UAS

 Coding

PENDAHULUAN

1.1 Latar Belakang

Di tengah pesatnya perkembangan teknologi digital, media sosial telah menjelma menjadi wadah utama bagi masyarakat untuk berbagi opini, pengalaman, dan informasi secara spontan dan masif. Platform-platform seperti Twitter, Facebook, dan Instagram memungkinkan ratusan juta pengguna menyampaikan pandangan mereka tentang beragam isu, mulai dari review produk hingga dinamika sosial-politik. Interaksi digital ini menghasilkan limpahan data yang dikenal sebagai user-generated content, yang menyimpan potensi analisis mendalam.

Analisis sentimen muncul sebagai teknik krusial untuk menafsirkan emosi dan sikap pengguna dari data teks tersebut. Metode ini secara otomatis mengklasifikasikan opini ke dalam kategori positif, netral, atau negatif, memberikan kemampuan bagi pelaku bisnis, pemerintah, maupun individu untuk mengambil keputusan berbasis data. Nilai strategisnya terlihat dari beragam aplikasi, perusahaan memakainya untuk mengukur kepuasan pelanggan dan mitigasi reputasi, politisi memantau respon konstituen, sementara industri kreatif seperti perfilman menggunakannya untuk menguji penerimaan karya.

Kemajuan di bidang kecerdasan buatan dan pemrosesan bahasa alami (NLP) semakin mempertegas peran analisis sentimen. Tantangan seperti volume data yang membengkak dan kebutuhan analisis real-time menjadikan pengembangan model yang akurat dan efisien sebagai topik relevan, baik secara akademis maupun praktis. Studi ini bertujuan mengeksplorasi implementasinya, sekaligus menilai dampaknya dalam mendukung pengambilan keputusan yang lebih terinformasi.

1.2 Rumusan Masalah

1. Bagaimana melakukan eksplorasi data awal (*Exploratory Data Analysis/EDA*) terhadap dataset sentimen Twitter?
2. Bagaimana proses *preprocessing* dilakukan untuk membersihkan dan mempersiapkan data teks?
3. Bagaimana membagi dataset menjadi data latih dan data uji dengan proporsi 70%:30%?

4. Bagaimana membangun dan mengimplementasikan model analisis sentimen menggunakan algoritma Random Forest?
5. Apakah model yang dibangun mampu mencapai akurasi minimal sebesar 70% sesuai dengan kriteria yang ditentukan?

1.3 Tujuan

1. Melakukan eksplorasi data awal (*Exploratory Data Analysis*/EDA) untuk memahami karakteristik dataset sentimen Twitter.
2. Melakukan *preprocessing* data teks agar data lebih bersih, konsisten, dan siap digunakan untuk pemodelan.
3. Membagi dataset menjadi data latih dan data uji dengan rasio 70:30 untuk memastikan evaluasi model yang adil dan representatif.
4. Menetapkan label target sentimen dalam bentuk numerik, yaitu Positif = 1, Netral = 0, dan Negatif = -1, guna mempermudah proses pelatihan model.
5. Membangun model analisis sentimen menggunakan algoritma *Random Forest* untuk mengklasifikasikan opini atau sentimen pengguna Twitter.
6. Mengukur performa model dengan menggunakan matriks evaluasi seperti akurasi, precision, recall, dan F1-score, serta memastikan bahwa model mencapai akurasi minimal 70%.

1.4 Batasan Masalah

1. Dataset yang digunakan berasal dari tautan yang telah ditentukan oleh dosen, yaitu dataset sentimen Twitter yang berisi dua kolom: teks (tweet) dan label sentimen.
2. Label sentimen dibatasi hanya pada tiga kategori, yaitu Positif (1), Netral (0), dan Negatif (-1).
3. Proses analisis sentimen hanya menggunakan algoritma *Random Forest* sebagai metode klasifikasi.
4. Data dibagi dengan rasio 70% untuk pelatihan (*training*) dan 30% untuk pengujian (*testing*).

5. Evaluasi performa model dilakukan berdasarkan matriks seperti akurasi, precision, recall, dan F1-score.

DATASET

2.1 Sumber Data

<https://shorturl.at/oUsrX>

2.2 Deskripsi Data

- Jumlah Baris : 40000 baris
- Jumlah Kolom : 2 kolom
- Kolom Teks : kolom ini berisi ulasan (review) mengenai film “Thunderbirds”. Teks ini merupakan data utama yang digunakan untuk analisis sentimen atau klasifikasi berdasarkan opini terhadap film tersebut.
- Kolom Label : kolom ini merupakan label atau kategori sentimen dari teks ulasan. Kolom ini memiliki dua nilai yaitu 0 dan 1, yang dimana 0 berarti ulasan yang bersifat negatif, dan 1 berarti ulasan tersebut bersifat positif

2.3 Contoh Data

text	label
I grew up (b. 196	0
When I put this r	0
Why do people v	0
Even though I ha	0
Im a die hard Da	1
A terrible movie	0
Finally watched t	1

Exploratory Data Analysis (EDA)

3.1 Distribusi Kelas

```
=====

3. Distribusi Label Sentimen:
label
0      20019
1      19981
Name: count, dtype: int64

=====
```

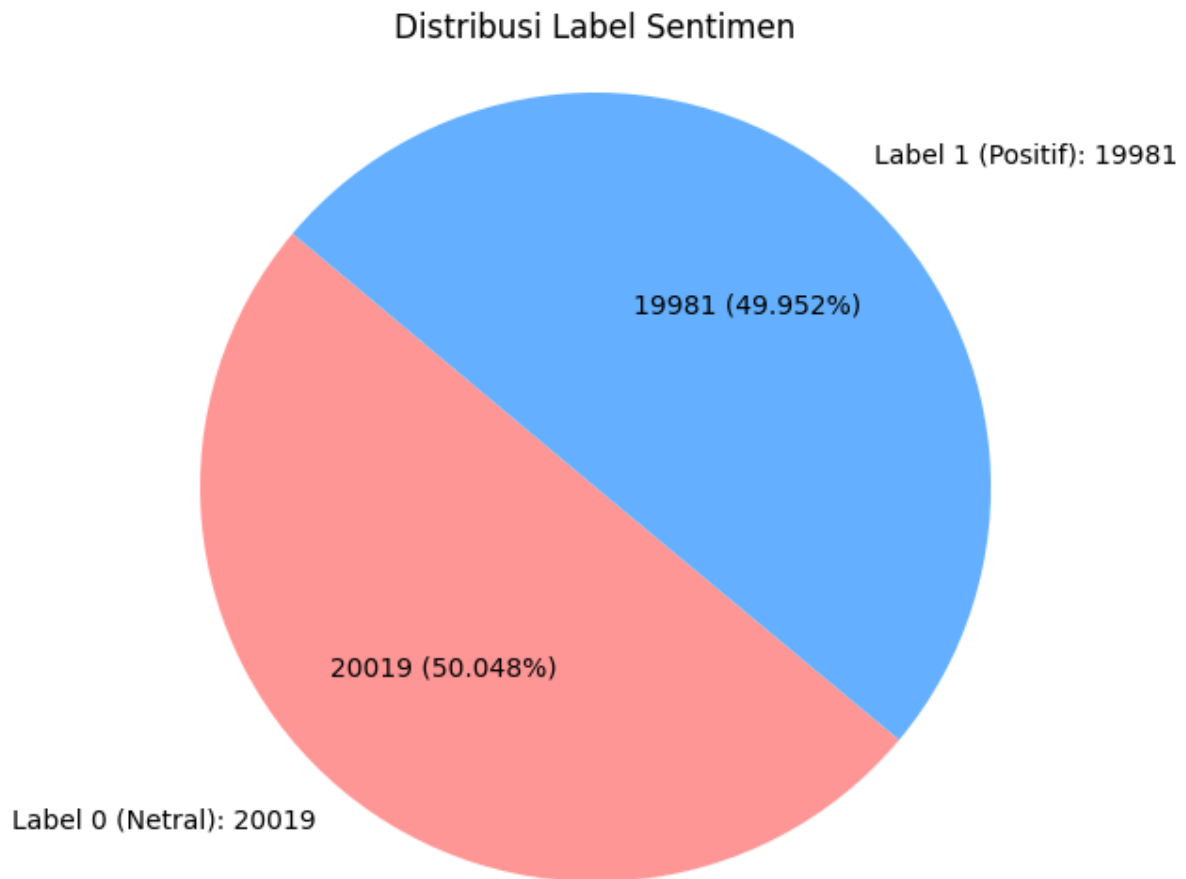
Berdasarkan gambar diatas, diketahui bahwa dataset memiliki dua kelas label yang didistribusikan sebagai berikut :

- Label 0 (Netral) : 20.019 data
- Label 1 (Positif) : 19.981 data
- Label -1 (Negatif) : 0 data

3.2 Visualisasi Data (Pie chart dari distribusi label)

```
1 # 5. Visualisasi Distribusi Label dengan Pie Chart
2 print("5. Visualisasi Distribusi Label Sentimen:")
3 label_counts = df['label'].value_counts()
4 # Fungsi untuk menampilkan jumlah dan persentase
5 def func(pct, allvalues):
6     absolute = int(np.round(pct / 100. * np.sum(allvalues)))
7     return f'{absolute} ({pct:.3f}%)'
8 # Membuat label yang lebih deskriptif
9 labels = [f'Label 0 (Netral): {label_counts[0]}' if i == 0 else f'Label 1 (Positif): {label_counts[1]}' for i in label_counts.index]
```

```
1 # Membuat Pie Chart
2 plt.figure(figsize=(8, 6))
3 plt.pie(label_counts, labels=labels, autopct=lambda pct: func(pct, label_counts), startangle=140, colors=['#ff9999', '#66b3ff', '#99ff99'])
4 plt.title('Distribusi Label Sentimen')
5 plt.axis('equal') # Equal aspect ratio ensures that pie chart is circular.
6 plt.show()
```

Gambar di atas menunjukkan visualisasi distribusi label sentimen dalam bentuk pie chart. Terlihat bahwa data terbagi hampir seimbang antara dua kelas, yaitu label '0' (Netral) sebanyak 20.019 data (50,048%) dan label '1' (Positif) sebanyak 19.981 data (49,952%). Tidak terdapat data dengan label '-1' (Negatif).

3.3 Analisis Distribusi Label dan Keseimbangan Data

```
1 # 3. Distribusi Label
2 print("3. Distribusi Label Sentimen:")
3 print(df['label'].value_counts())
4 print("\n" + "="*60 + "\n")
```

Menghitung jumlah data untuk tiap label:

0: 20019

1: 19981

Berdasarkan hasil tersebut, jumlah data tiap kelas hampir seimbang. Selisih antara kelas positif dan netral sangat kecil (0,096%), sehingga tidak ada masalah imbalance yang signifikan dalam dataset ini. Distribusi ini memungkinkan model untuk belajar secara adil dari kedua kelas tanpa perlu teknik penyeimbangan data tambahan seperti oversampling atau undersampling.

Fungsi dari cek data imbalance:

1. Mendeteksi masalah ketidakseimbangan (imbalance) dalam label dataset
2. Jika data tidak seimbang:
 - Model bisa bias terhadap kelas mayoritas.
 - Akurasi terlihat tinggi tapi performa sebenarnya rendah untuk kelas minoritas.
3. Memberi pertimbangan apakah perlu:
 - Oversampling (menambah data minoritas),
 - Undersampling (mengurangi data mayoritas),
 - Menggunakan algoritma khusus yang tahan terhadap imbalance.

PREPROCESSING DATA

4.1 Pembersihan Data

Pembersihan data adalah proses untuk menyiapkan teks mentah agar lebih bersih dan konsisten sebelum diproses oleh algoritma machine learning. Teks yang kotor bisa mempengaruhi akurasi model.

```
1 # 5. Preprocessing text
2 stop_words = set(stopwords.words('english')) - {
3     "not", "no", "nor", "won't", "can't", "don't", "didn't", "isn't", "wasn't", "couldn't", "shouldn't", "wouldn't"}
4 lemmatizer = WordNetLemmatizer()
5 stemmer = PorterStemmer()
6
7 def preprocess_for_sentiment(text):
8     # 1. Jadikan semua uruf kecil
9     text = text.lower()
10
11     # 2. Hilangkan tag HTML
12     text = BeautifulSoup(text, "html.parser").get_text()
13
14     # 3. Convert emoji ke teks
15     text = emoji.demojize(text, delimiters=(" ", " ")).replace("_", " ")
16
17     # 4. Hapus tanda baca (kecuali tanda negasi, karena pada stopwords tanda negasi diperlukan)
18     text = re.sub(r"[^\w\s']+", '', text)
19
20     # 5. Tokenisasi (memecah teks menjadi bagian-bagian yang lebih kecil)
21     words = nltk.word_tokenize(text)
22
23     # 6. Hapus kata-kata stopwords tetapi pertahankan negasi yang sudah ditentukan di atas
24     words = [word for word in words if word not in stop_words]
25
26     # 7. Lemmatization (mengubah kata ke bentuk dasar yang lebih bermakna dengan mempertimbangkan konteks dan makna kata tersebut)
27     words = [lemmatizer.lemmatize(word) for word in words]
28
29     # 8. Gabung kembali teks
30     return " ".join(words)
```

Teks dibersihkan menggunakan teknik:

1. Jadikan semua uruf kecil (*Lowercase*)
2. Menghilangkan tag HTML
3. Konversi emoji ke teks
4. Hapus tanda baca (kecuali tanda negasi, karena pada stopwords tanda negasi diperlukan)
5. Tokenisasi
6. Menghapus *stopword* tetapi mempertahankan kata-kata yang sudah ditentukan
7. Lemmatization (mengubah kata ke bentuk dasar yang lebih bermakna dengan mempertimbangkan konteks dan makna kata tersebut)
8. Gabung kembali teks

Proses ini memastikan data teks siap untuk diproses oleh model machine learning.

4.2 TF-IDF

```
1 # Pisahkan data latih berdasarkan label
2 X_train_positive = X_train[y_train == 1]
3 X_train_netral = X_train[y_train == 0]
4
5 # Inisialisasi TF-IDF Vectorizer
6 tfidf = TfidfVectorizer(max_features=None)
7
8 # Fit dan transform data latih, lalu transform data uji
9 X_train_tfidf = tfidf.fit_transform(X_train) #digunakan sekali saja di data training → untuk membuat dan belajar dari kamus fitur.
10 X_test_tfidf = tfidf.transform(X_test) #digunakan pada data testing → untuk menggunakan kamus yang sudah dibuat agar dimensi fitur tetap konsisten.
11
12 # Tampilkan bentuk hasil vektorisasi
13 print(f"Bentuk X_train setelah TF-IDF: {X_train_tfidf.shape}")
14 print(f"Bentuk X_test setelah TF-IDF: {X_test_tfidf.shape}")
15
16 # TF-IDF untuk data positif
17 tfidf_pos = TfidfVectorizer()
18 X_pos_tfidf = tfidf_pos.fit_transform(X_train_positive)
19 words_pos = tfidf_pos.get_feature_names_out()
20 mean_pos = np.asarray(X_pos_tfidf.mean(axis=0)).ravel()
21 top_pos_idx = mean_pos.argsort()[::-1][:20]
22 top_words_pos = words_pos[top_pos_idx]
23 top_scores_pos = mean_pos[top_pos_idx]
24 tfidf_dict_pos = dict(zip(words_pos, mean_pos))
25
26 # TF-IDF untuk data netral
27 tfidf_net = TfidfVectorizer()
```

```

1  # ===== VISUALISASI POSITIF =====
2  plt.figure(figsize=(14, 5))
3
4  # Diagram batang TF-IDF Positif
5  plt.subplot(1, 2, 1)
6  plt.barh(top_words_pos[::-1], top_scores_pos[::-1], color='green')
7  plt.title("Top 20 TF-IDF Kata - Positif")
8  plt.xlabel("Skor TF-IDF")
9
10 # WordCloud Positif
11 wordcloud_pos = WordCloud(width=600, height=400, background_color='white',
12                             colormap='Greens').generate_from_frequencies(tfidf_dict_pos)
13
14 plt.subplot(1, 2, 2)
15 plt.imshow(wordcloud_pos, interpolation='bilinear')
16 plt.axis('off')
17 plt.title("WordCloud - Komentar Positif", fontsize=16)
18
19 plt.tight_layout()
20 plt.show()
21
22 # ===== VISUALISASI NETRAL =====
23 plt.figure(figsize=(14, 5))
24
25 # Diagram batang TF-IDF Netral
26 plt.subplot(1, 2, 1)
27 plt.barh(top_words_net[::-1], top_scores_net[::-1], color='blue')
28 plt.title("Top 20 TF-IDF Kata - Netral")
29 plt.xlabel("Skor TF-IDF")
30
31 # WordCloud Netral
32 wordcloud_net = WordCloud(width=600, height=400, background_color='white',
33                             colormap='Blues').generate_from_frequencies(tfidf_dict_net)
34
35 plt.subplot(1, 2, 2)
36 plt.imshow(wordcloud_net, interpolation='bilinear')
37 plt.axis('off')
38 plt.title("WordCloud - Komentar Netral", fontsize=16)
39
40 plt.tight_layout()
41 plt.show()

```

Dalam proyek ini, teks diubah menjadi bentuk numerik dengan menggunakan metode TF-IDF (Term Frequency - Inverse Document Frequency), yang merupakan salah satu bagian teknik text vectorization yang menghitung bobot setiap kata dalam dataset berdasarkan dua aspek utama yaitu :

- Kata yang sering muncul di dalam dataset
- Kata yang jarang muncul di dalam dataset

Proses ini juga melakukan tokenisasi otomatis, yakni dengan memecah teks menjadi kata per kata sebelum menghitung bobotnya. Hasil dari proses ini merupakan representasi vektor numerik dari setiap teks, yang kemudian digunakan sebagai input pelatihan model machine learning. Metode TF-IDF dipilih karena tidak hanya

Tujuan dari pembagian ini adalah agar model machine learning dapat dilatih pada sebagian besar data, namun tetap memiliki bagian data yang tidak terlihat sebelumnya untuk mengevaluasi performa model secara adil.

```
1 # 6. Pembagian Data Train and Test
2 # Kolom 'text' berisi teks komentar dan kolom 'label' berisi sentimen (positif/neutral)
3 X = df['text']          # Fitur input (komentar)
4 y = df['label']         # Label target (sentimen)
5
6 # Bagi data 70% train dan 30% test
7 X_train, X_test, y_train, y_test = train_test_split(
8     X, y, test_size=0.3, random_state=42, stratify=y)
9 # random_state=42 membuat pembagian data tetap konsisten setiap kali dijalankan.
10 # stratify=y memastikan distribusi kelas tetap seimbang antara train dan test.
11
12 # Menampilkan jumlah data hasil pembagian
13 print(f"Jumlah data latih: {len(X_train)}")
14 print(f"Jumlah data uji: {len(X_test)}")
15
16 # Terapkan preprocessing
17 X_train = X_train.apply(preprocess_for_sentiment)
18 X_test = X_test.apply(preprocess_for_sentiment)
```

Berdasarkan gambar diatas, jumlah data train yakni sebanyak 28.000 data dan jumlah data test yakni sebanyak 12.000.

MODELING

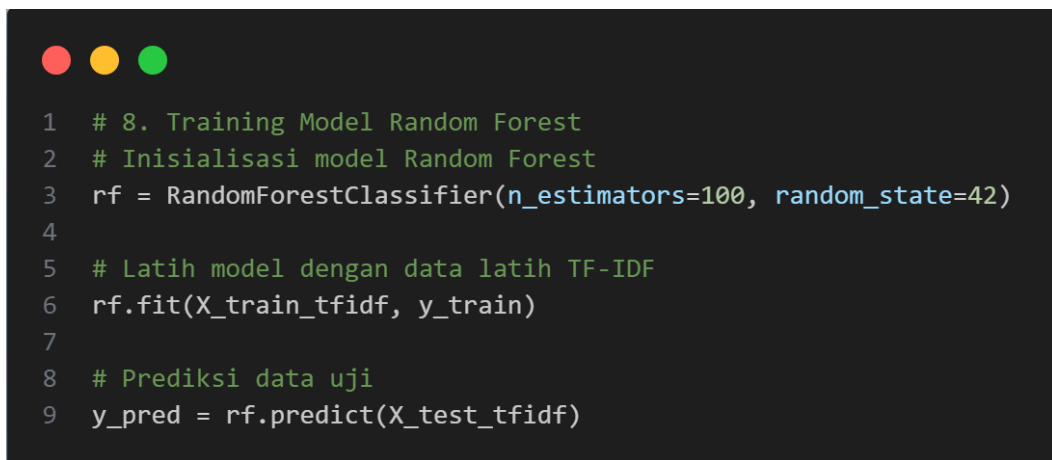
5.1 Pemilihan Algoritma

Dalam proyek ini, algoritma yang digunakan untuk melakukan klasifikasi sentimen adalah algoritma Random Forest. Algoritma ini dipilih karena :

- Mampu menangani data berdimensi tinggi seperti hasil dari TF-IDF.
- Memiliki performa yang stabil untuk tugas klasifikasi teks.
- Cukup cepat dan efisien untuk dataset yang berukuran 40.000 baris.

5.2 Pelatihan Model

Setelah data teks diubah menjadi vektor numerik dengan menggunakan TF-IDF, langkah berikutnya adalah melatih model Random Forest. Proses ini dikenal sebagai fitting, dimana ketika model mencoba mempelajari hubungan antara teks ulasan dan label sentimennya (positif atau netral) dari data train.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in Python and is numbered from 1 to 9. It shows the initialization of a Random Forest classifier and its training on TF-IDF data.

```
1 # 8. Training Model Random Forest
2 # Inisialisasi model Random Forest
3 rf = RandomForestClassifier(n_estimators=100, random_state=42)
4
5 # Latih model dengan data latih TF-IDF
6 rf.fit(X_train_tfidf, y_train)
7
8 # Prediksi data uji
9 y_pred = rf.predict(X_test_tfidf)
```

Dengan menjalankan kode tersebut, model dapat mempelajari pola-pola yang ada di dalam teks. Hasil yang didapatkan dari proses ini merupakan model yang siap digunakan untuk memprediksi sentimen dari teks baru yang belum pernah dilihat sebelumnya, yaitu data test.

5.3 Parameter yang Digunakan

- `train_test_split()`



```
1 train_test_split(  
2     X, y, test_size=0.3, random_state=42, stratify=y)
```

- X, y : Data input dan data target
- test_size=0.3 : merupakan 30% data yang akan digunakan sebagai data test, sisanya akan digunakan sebagai data train.
- random_state=42 : merupakan angka acak agar hasil pembagian data konsisten setiap kali dijalankan.
- stratify=y : pembagian data dengan menjaga proporsi label tetap seimbang di saat di train dan di test.

- RandomForestClassifier()



```
1 RandomForestClassifier(n_estimators=100, random_state=42)
```

- n_estimators=100: Jumlah pohon keputusan dalam model. Semakin banyak, semakin stabil.
- random_state=42: agar hasil pelatihan model konsisten tiap kali dijalankan.

- TfidfVectorizer()



```
1 tfidf = TfidfVectorizer(max_features=None)
```

- max_features=None: tidak membatasi jumlah kata unik yang digunakan.

- plt.pie()



```
1 plt.pie(label_counts, labels=labels, autopct=lambda pct:  
2 func(pct, label_counts), startangle=140,  
3 colors=['#ff9999', '#66b3ff', '#99ff99'])
```

- label_counts: merupakan data yang ingin divisualisasi.
- labels: merupakan label deskriptif tiap bagian pie.
- autopct: menampilkan persentase dan jumlah data per bagian.

- sns.heatmap()



```
1 sns.heatmap(cm, annot=True, fmt='d', cmap='Greens')
```

- cm: confusion matrix sebagai hasil evaluasi model.
- annot=True: menampilkan nilai di dalam kotak.
- fmt='d': Format angka desimal/integer.

EVALUASI MODEL

6.1 Akurasi Model



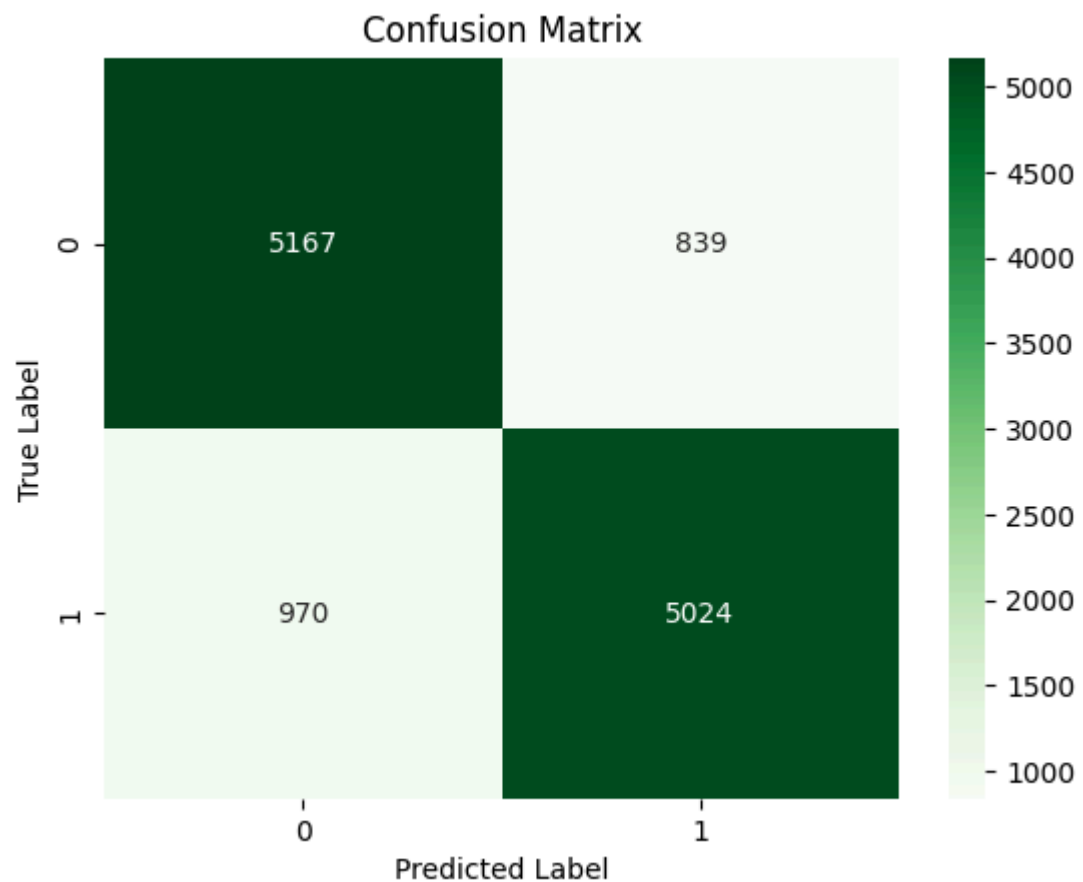
```
1 # Akurasi
2 acc = accuracy_score(y_test, y_pred)
3 print(f"Akurasi: {acc:.4f}")
```

===== Evaluasi Model =====

Akurasi: 0.8492

Berdasarkan hasil pengujian, model mencapai akurasi sebesar 0.8492 atau sekitar 84,92%, yang berarti sekitar 84 dari setiap 100 prediksi yang dihasilkan model adalah benar. Ini menunjukkan bahwa model memiliki kinerja yang cukup baik dalam membedakan antara dua kelas yang diuji, bahkan melebihi target minimal akurasi sebesar 70%.

6.2 Confusion Matrix



Confusion matrix diatas menunjukkan bahwa model berhasil memprediksi 5167 komentar netral (True Netral) dan 5024 komentar positif (True Positive) dengan benar. Namun, masih terdapat 839 komentar netral yang salah diklasifikasikan sebagai positif (False Positive), serta 970 komentar positif yang diklasifikasikan sebagai netral (False Netral). Nilai kesalahan ini relatif seimbang, yang menunjukkan bahwa model tidak bias terhadap salah satu kelas.

6.3 Precision, Recall, F1-Score

Classification Report:				
	precision	recall	f1-score	support
0	0.84	0.86	0.85	6006
1	0.86	0.84	0.85	5994
accuracy			0.85	12000
macro avg	0.85	0.85	0.85	12000
weighted avg	0.85	0.85	0.85	12000

Berdasarkan gambar di atas, hasil classification report menunjukkan nilai precision, recall, dan F1-score yang seimbang antara kedua kelas, yaitu masing-masing sekitar 0.84–0.86. Hal ini menunjukkan bahwa model memiliki kemampuan prediksi yang konsisten dan stabil untuk kedua kelas, baik dalam mengenali kelas 0 maupun kelas 1. Jika kelas 0 dan 1 merepresentasikan komentar netral dan positif, maka model dapat mengenali keduanya dengan performa yang seimbang.

Keterangan :

- Precision : menunjukkan seberapa akurat model dalam memprediksi suatu kelas.
- Recall : menunjukkan seberapa baik model mendeteksi semua data yang termasuk dalam suatu kelas.
- F1-score : menunjukkan rata-rata harmonik dari precision dan recall yang berguna untuk melihat keseimbangan antara keduanya.
- Support : merupakan jumlah data aktual untuk setiap kelas yang berada di dalam dataset.

6.4 Analisis Hasil Evaluasi

Secara keseluruhan, model berhasil mencapai akurasi di atas target (84.92%), yang menunjukkan kinerja yang cukup baik. Tidak ditemukan adanya kelas yang sangat sulit diprediksi, karena precision dan recall untuk kedua kelas relatif seimbang.

Hal ini didukung juga oleh distribusi label yang seimbang, sehingga model tidak bias terhadap salah satu kelas.

KESIMPULAN

Dalam proyek ini, telah dilakukan analisis sentimen terhadap dataset ulasan film menggunakan algoritma Random Forest. Proses dimulai dari eksplorasi data, pembersihan teks, transformasi data menggunakan metode TF-IDF, hingga pelatihan dan evaluasi model. Dataset yang digunakan cukup seimbang antara dua kelas, yaitu netral dan positif, sehingga memungkinkan model untuk belajar secara adil tanpa bias kelas.

Model Random Forest dipilih karena kemampuannya dalam menangani data berdimensi tinggi dan menunjukkan performa klasifikasi yang stabil. Hasil evaluasi menunjukkan bahwa model mampu mencapai akurasi sebesar 84,92%, melebihi target akurasi minimum yang ditetapkan yaitu 70%. Nilai precision, recall, dan F1-score yang relatif seimbang menunjukkan bahwa model dapat mengenali kedua kelas dengan baik dan tidak menunjukkan bias terhadap salah satu kelas.

Secara keseluruhan, proyek ini membuktikan bahwa algoritma Random Forest cukup efektif dalam melakukan klasifikasi sentimen terhadap data teks, khususnya dalam konteks ulasan film. Pendekatan yang digunakan dalam preprocessing, pembagian data, dan evaluasi juga memberikan hasil yang andal serta dapat dijadikan dasar untuk pengembangan lebih lanjut dalam bidang analisis sentimen.