



KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE

Dosen Pengampu: Adityo Permana Wibowo, S.Kom., M.Cs

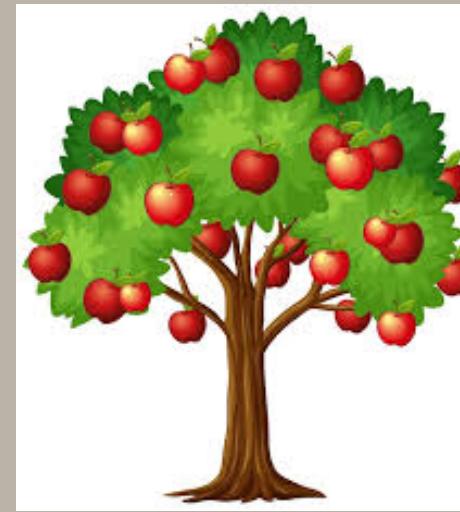
Mata Kuliah Text Mining & Natural Language Processing

Program Studi Sains Data_Fakultas Sains dan Teknologi_Universitas Teknologi Yogyakarta 2025

ANGGOTA KELOMPOK 6



Lathif Ramadan
5231811022



Andini Angel M
5231811029



Rama Panji N
5231811033



Giffari Riyanda
5231811036

LINK LAPORAN: https://github.com/LatiefDataVisionary/text-mining-and-natural-language-processing-college-task/blob/main/docs/TA_TMNLP_5231811022__Lathif%20Ramadan.pdf



O O O O

DAFTAR ISI

1a. KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE (Menggunakan 2 Kategori Label: Positive dan Negative)

1b. KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE (Menggunakan 3 Kategori Label: Positive dan Negative) 14

2a. PERBAIKAN KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE DENGAN TUNING PARAMETER (Menggunakan 2 Kategori Label: Positive dan Negative) 25

2b. PERBAIKAN KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE DENGAN TUNING PARAMETER (Menggunakan 3 Kategori Label: Positive, Negative, dan Neutral).

3a. PROSES KLASIFIKASI MENGGUNAKAN ALGORITMA (Menggunakan 2 Kategori Label: Positive dan Negative)

3b. PROSES KLASIFIKASI MENGGUNAKAN ALGORITMA (Menggunakan 3 Kategori Label: Positive, Negative, dan Neutral) 91

KESIMPULAN UTAMA

o o o o

1a. KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE (Menggunakan 2 Kategori Label: Positive dan Negative)

Dataset: *ramadan_labeled_sentiment.csv* (836 baris, 8 kolom).

- Preprocessing Awal: Teks pada tweet_clean langsung digunakan dengan TF-IDF (*max_features=1000*, menghasilkan 937 fitur efektif pada data training).
- Pembagian Data: 75% Training (627 sampel), 25% Testing (209 sampel), stratifikasi diterapkan.
- Tuning Hyperparameter (GridSearchCV):
 - Parameter terbaik: *criterion: 'entropy'*, *max_depth: 40*, *min_samples_leaf: 1*, *min_samples_split: 20*, *class_weight: None*.
 - Skor cross-validation (accuracy) pada data training: 0.7177.

○ ○ ○ ○

- **Evaluasi pada Data Testing:**

- Akurasi: 0.7033 (70.33%)
- ROC AUC Score: 0.7365

Laporan Klasifikasi Decision Tree:

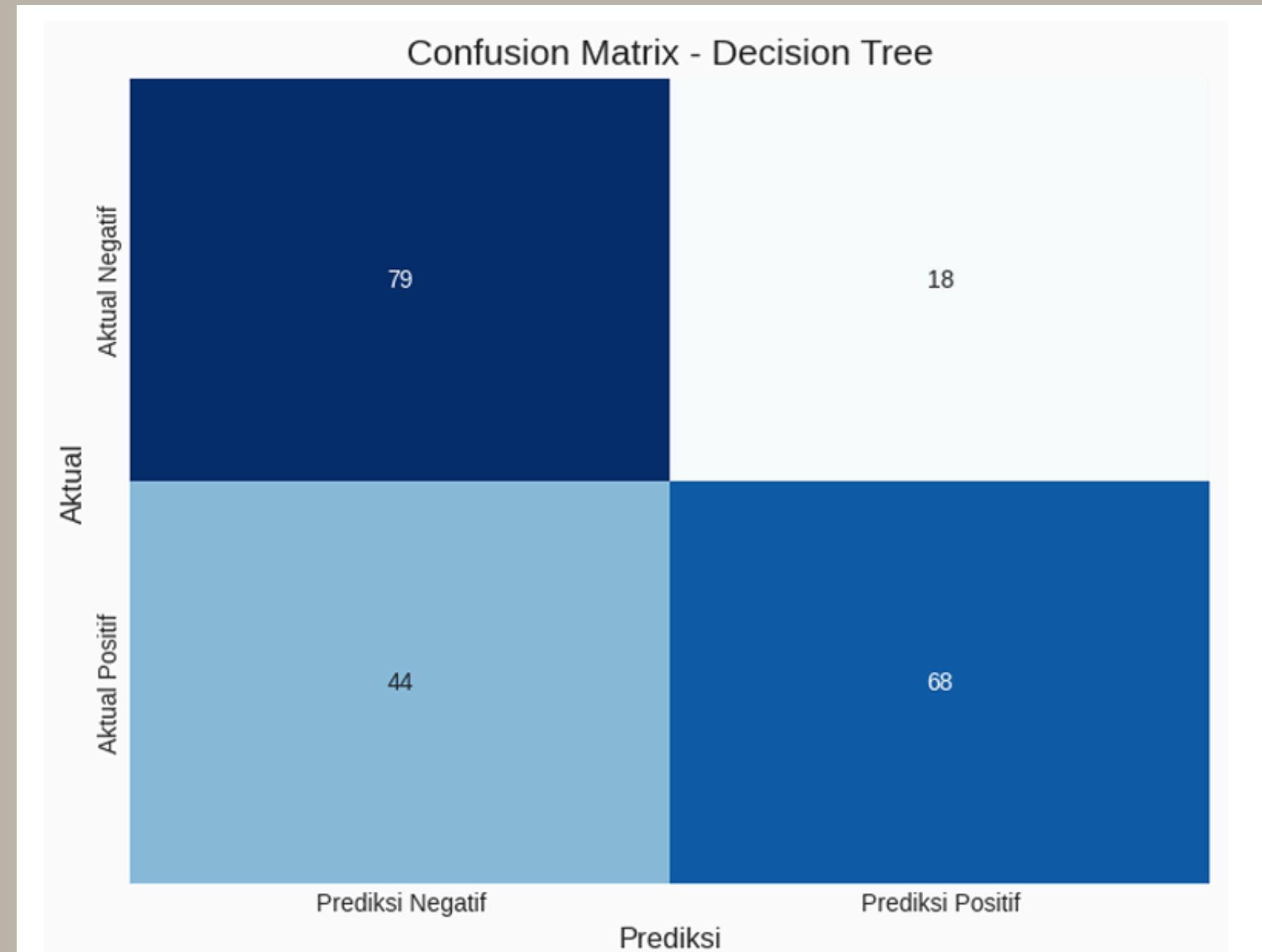
	precision	recall	f1-score	support
Negative (0)	0.64	0.81	0.72	97
Positive (1)	0.79	0.61	0.69	112
accuracy			0.70	209
macro avg	0.72	0.71	0.70	209
weighted avg	0.72	0.70	0.70	209

- Dari Laporan Klasifikasi:
 - Model lebih baik dalam recall untuk kelas 'Negative' (0.81), namun presisinya lebih rendah (0.64).
 - Model lebih baik dalam presisi untuk kelas 'Positive' (0.79), namun recall-nya lebih rendah (0.61).
 - F1-score rata-rata tertimbang (weighted avg) adalah 0.70.

o o o o

Visualisasi:

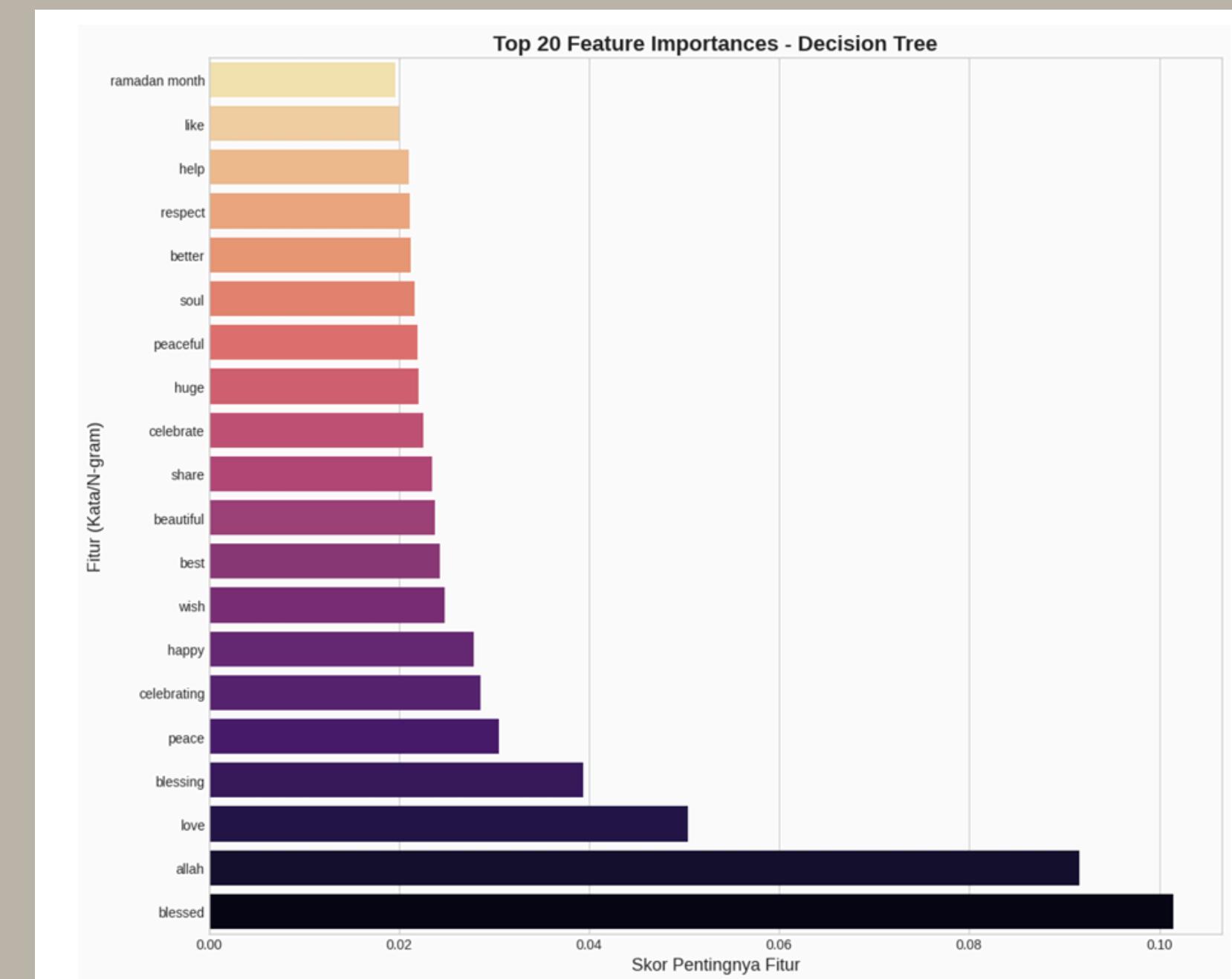
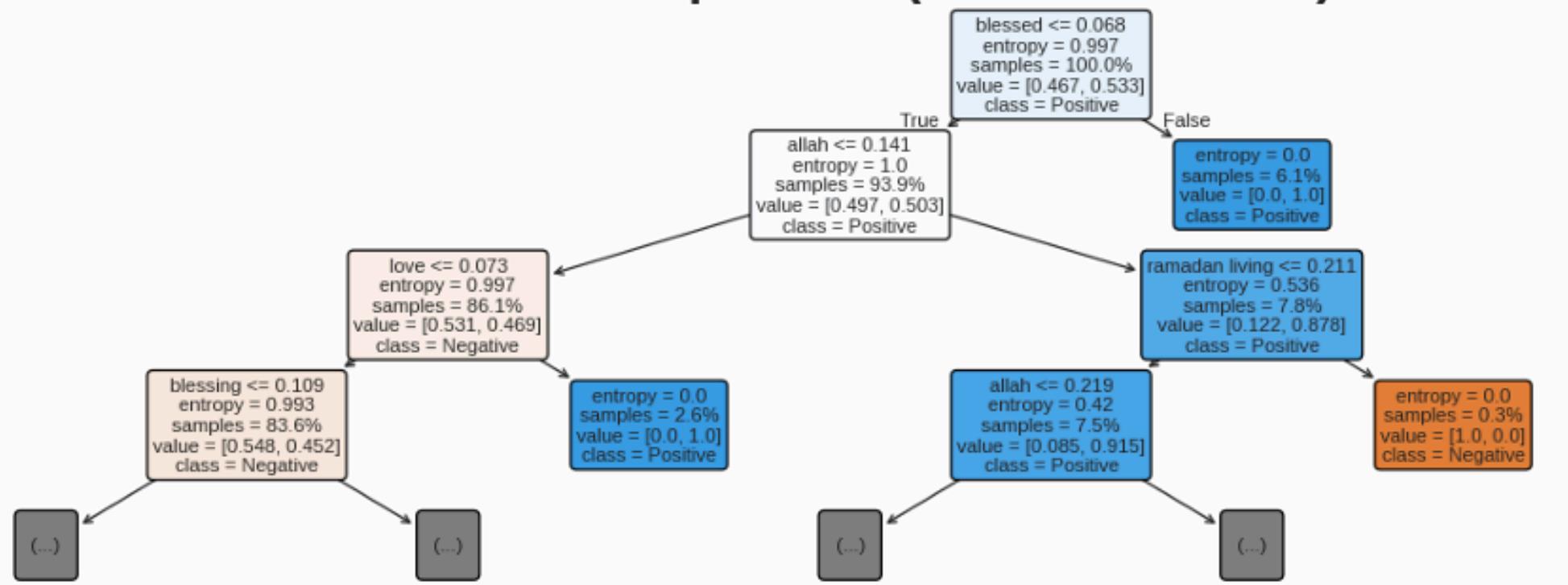
- Confusion Matrix menunjukkan adanya False Positives (18) dan False Negatives (44) yang cukup signifikan, mengindikasikan ruang untuk perbaikan.



Four solid black circles are arranged in a horizontal row, centered on the page.

Pohon Keputusan (3 level) dan Feature Importances (kata kunci seperti blessed, allah, love) memberikan wawasan awal tentang bagaimana model membuat keputusan.

Visualisasi Pohon Keputusan (3 Level Pertama)



o o o o

1b. KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE (Menggunakan 3 Kategori Label: Positive dan Negative)

- Dataset: data_3_kelas_real.csv (836 baris, 11 kolom).
- Preprocessing Awal: Teks pada text_processed_raw dengan TF-IDF (max_features=1500).
- Pembagian Data: 75% Training (627 sampel), 25% Testing (209 sampel), stratifikasi diterapkan.
- Tuning Hyperparameter (GridSearchCV):
 - Parameter terbaik: ccp_alpha: 0.0, class_weight: None, criterion: 'entropy', max_depth: 10, min_samples_leaf: 8, min_samples_split: 2.
 - Skor cross-validation (accuracy) pada data training: 0.8565.



Akurasi Model Decision Tree pada Data Testing: 0.8612
ROC AUC Score tidak dapat dihitung (mungkin hanya satu kelas yang diprediksi).

Laporan Klasifikasi Decision Tree:

	precision	recall	f1-score	support
Negative (0)	0.00	0.00	0.00	4
Positive (1)	0.88	0.97	0.92	173
Neutral (2)	0.67	0.38	0.48	32
accuracy			0.86	209
macro avg	0.52	0.45	0.47	209
weighted avg	0.83	0.86	0.84	209

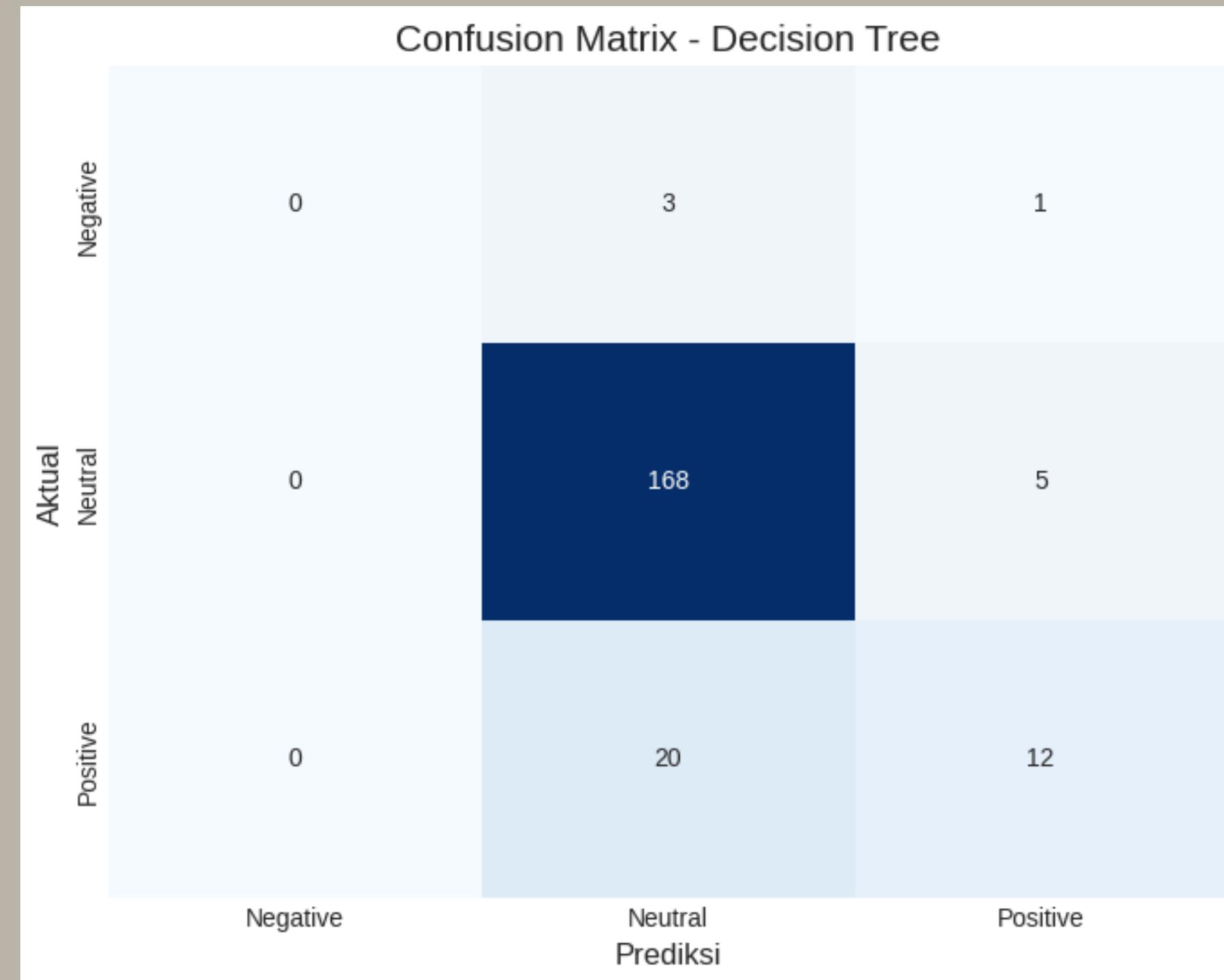
• Evaluasi pada Data Testing:

- Akurasi: 0.8612 (86.12%)
- ROC AUC Score: Tidak dapat dihitung (kemungkinan karena kelas minoritas 'Negative' tidak terprediksi dengan baik atau sama sekali tidak terprediksi sebagai benar).
- Dari Laporan Klasifikasi:
- Kelas 'Negative' (0) sangat buruk performanya (presisi, recall, f1-score 0.00), yang mengindikasikan model kesulitan mengidentifikasi kelas minoritas ini.
- Kelas 'Positive' (1) menunjukkan performa yang sangat baik (precision 0.88, recall 0.97, f1-score 0.92).
- Kelas 'Neutral' (2) memiliki performa sedang (precision 0.67, recall 0.38, f1-score 0.48).
- F1-score rata-rata tertimbang (weighted avg) adalah 0.84.

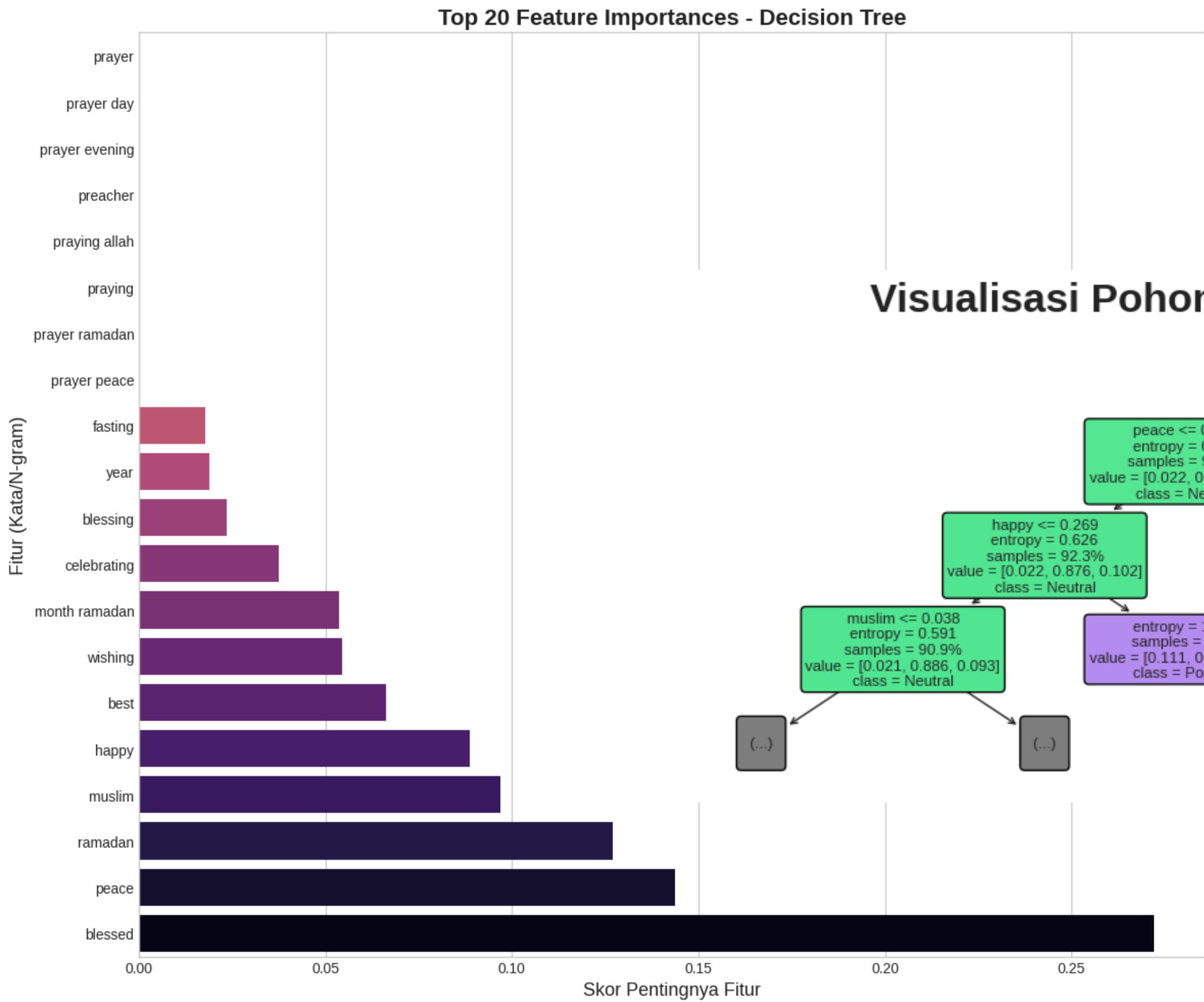
o o o o

Visualisasi:

- Confusion Matrix (3x3) secara jelas menunjukkan kesulitan model pada kelas 'Negative' dan beberapa misklasifikasi antara 'Positive' dan 'Neutral'.

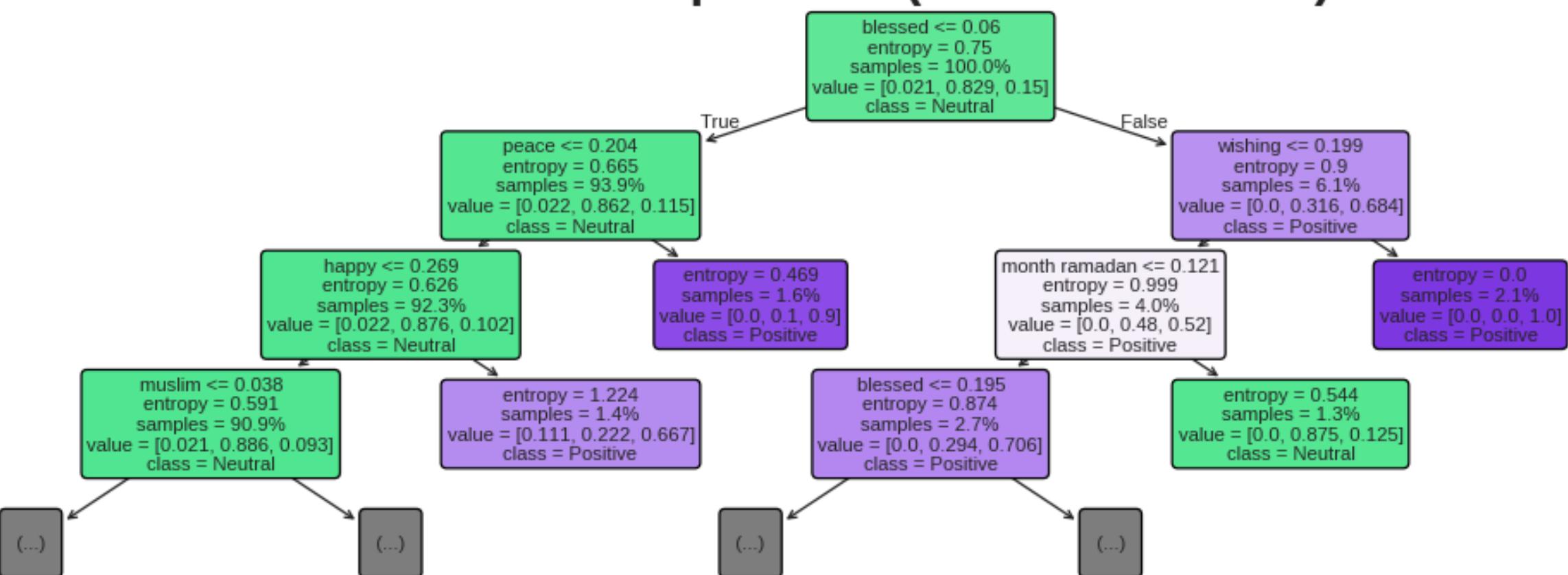


o o o o



- Pohon Keputusan (3 level) dan Feature Importances (kata kunci seperti blessed, peace, ramadan) juga ditampilkan.

Visualisasi Pohon Keputusan (3 Level Pertama)



○ ○ ○ ○

Observasi Umum Bagian 1:

- Akurasi awal untuk 3 kategori (0.8612) lebih tinggi daripada 2 kategori (0.7033) pada tahap ini. Namun, performa pada kelas minoritas 'Negative' di skenario 3 label sangat rendah.
- Penggunaan GridSearchCV membantu menemukan kombinasi parameter yang lebih baik daripada default.
- Ada indikasi jelas bahwa preprocessing teks yang lebih canggih, feature engineering/selection yang lebih baik, dan teknik penanganan ketidakseimbangan kelas (terutama untuk kasus 3 label) diperlukan untuk meningkatkan performa model lebih lanjut.
- Output kode dan visualisasi (confusion matrix, pohon keputusan, feature importances) sangat membantu dalam memahami perilaku model awal dan mengidentifikasi area perbaikan.

o o o o

2. PERBAIKAN KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE DENGAN TUNING PARAMETER

Bagian kedua eksperimen difokuskan pada upaya meningkatkan performa model Decision Tree melalui serangkaian teknik, mulai dari preprocessing teks yang lebih komprehensif, tuning parameter yang lebih ekstensif, hingga penanganan ketidakseimbangan data. Hasil perbaikan ini diamati untuk skenario 2 kategori label dan 3 kategori label.

○ ○ ○

**2a. PERBAIKAN KLASIFIKASI MENGGUNAKAN ALGORITMA
DECISION TREE DENGAN TUNING PARAMETER (Menggunakan 2
Kategori Label: Positive dan Negative).**

o o o o

1. Perbaikan Klasifikasi Decision Tree (2 Kategori Label - Positive, Negative):

- Preprocessing Teks Mendalam:
 - Kode: Implementasi preprocess_text_step1 yang mencakup:
 - Case Folding (huruf kecil semua).
 - Penghilangan URL, mention, dan hashtag.
 - Penghilangan karakter non-alfanumerik (kecuali spasi, membiarkan angka).
 - Tokenizing menggunakan word_tokenize NLTK.
 - Stopword Removal menggunakan daftar stopwords default NLTK.
 - Lemmatization menggunakan WordNetLemmatizer NLTK.
 - Output: Teks yang lebih bersih dan terstandardisasi (text_processed), serta pengecekan isnull().sum() untuk memastikan tidak ada missing values setelah preprocessing dan fillna('').

○ ○ ○ ○

Tuning Parameter Lanjutan (GridSearchCV):

	Sebelum	Sesudah
Max_depth	Nilai yang dipilih 30	Nilai yang dipilih 40
Hasil akurasi	0.70	0.70

	Sebelum	Sesudah
Min_samples_split	Nilai yang dipilih 20	Nilai yang dipilih 5
Hasil akurasi	0.70	0.72

	Sebelum	Sesudah
criterion	Gini	Entropy
Hasil akurasi	0.72	0.72

GridSearchCV selesai.

Parameter terbaik yang ditemukan untuk Decision Tree:
`{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'entropy',
'max_depth': 40, 'min_samples_leaf': 8, 'min_samples_split': 5}`

o o o o

- **Tuning Parameter Lanjutan (GridSearchCV):**

- Beberapa iterasi tuning dilakukan dengan memperluas rentang parameter:
- `max_depth`: Dari [None, 10, ..., 50] (awal: 40, akurasi CV 0.7177) menjadi [None, 5, ..., 120] (tetap terpilih 40, akurasi tes 0.70 -> 0.70).
- `min_samples_split`: Dari [2, 5, ..., 20] (awal: 20, akurasi CV 0.7177) menjadi [2, 5, ..., 30] (terpilih 5, akurasi tes 0.70 -> 0.72).
- `criterion`: Tetap 'entropy' setelah dibandingkan dengan 'gini' (akurasi tes 0.72 -> 0.72).
- Kombinasi terbaik awal setelah tuning parameter ini adalah: `criterion='entropy'`, `max_depth=40`, `min_samples_split=5` (dan parameter `leaf` & `class_weight` dari tuning pertama).

○ ○ ○ ○

Tuliskan perbandingan hasilnya pada tabel di bawah ini:

	Sebelum	Sesudah
validation	Kami memakai 2 yakni Split-Validation (Test size = 25%) dan Cross-Validation.	Kami memakai Split-Validation (Test size = 20%).
Hasil akurasi	0.72	0.72

Metode Validasi:

- Split Validation: `test_size` diubah dari 0.25 (awal) menjadi 0.2 untuk evaluasi akhir.
- Cross-Validation (CV): Digunakan dalam `GridSearchCV (cv=5)` untuk pemilihan hyperparameter yang robust pada data training.
- Kedua metode ditekankan pentingnya; CV untuk pengembangan model, Split Validation untuk estimasi kinerja akhir.

o o o o

- **Feature Selection (SelectKBest dengan Chi2):**

- Kode: Menggunakan SelectKBest(score_func=chi2, k=1500) pada matriks TF-IDF (yang memiliki 941 fitur).
- Output: UserWarning muncul karena k=1500 lebih besar dari n_features=941, sehingga semua 941 fitur dikembalikan (tidak ada pengurangan fitur).
- Efek pada Akurasi: Meskipun fitur tidak berkurang, model yang dilatih setelah tahap "seleksi" ini (dengan 941 fitur) menunjukkan peningkatan akurasi signifikan pada data tes: dari 0.72 menjadi 0.80. Ini mengindikasikan bahwa proses fit_transform dari SelectKBest, meskipun tidak mengurangi fitur, mungkin telah mengubah representasi fitur atau interaksinya dengan cara yang positif bagi model. (Penting untuk dicatat bahwa TF-IDF disini max_features nya sudah tidak lagi 1000 seperti awal, melainkan tidak dibatasi dan menghasilkan 941 fitur)



Tuliskan perbandingan hasilnya pada tabel di bawah ini:

	Sebelum	Sesudah
Feature selection (jumlah kolom)	Dalam proyek ini, kami menggunakan TF-IDF Vectorization untuk mengubah data teks menjadi representasi numerik. Meskipun kami membatasi jumlah fitur maksimum yang dihasilkan oleh TF-IDF (menggunakan parameter <code>max_features</code>), kami tidak menerapkan metode seleksi fitur eksplisit seperti pemilihan fitur berdasarkan statistik (contoh: SelectKBest) atau seleksi fitur berbasis model sebelum melatih model Decision Tree. Pembatasan fitur dilakukan langsung selama proses vektorisasi TF-IDF. Analisis pentingnya fitur (feature importance) dilakukan setelah model dilatih untuk mengidentifikasi fitur yang paling berkontribusi pada prediksi model.	941 fitur
Hasil akurasi	0.72	0.8

○ ○ ○ ○

- **Efek pada Akurasi:**

Meskipun fitur tidak berkurang, model yang dilatih setelah tahap "seleksi" ini (dengan 941 fitur) menunjukkan peningkatan akurasi signifikan pada data tes: dari 0.72 menjadi 0.80. Ini mengindikasikan bahwa proses `fit_transform` dari `SelectKBest`, meskipun tidak mengurangi fitur, mungkin telah mengubah representasi fitur atau interaksinya dengan cara yang positif bagi model. (Penting untuk dicatat bahwa TF-IDF disini `max_features` nya sudah tidak lagi 1000 seperti awal, melainkan tidak dibatasi dan menghasilkan 941 fitur)

o o o o

- **Penanganan Imbalance Data (SMOTE):**

- Kode: Implementasi SMOTE pada data training setelah feature selection.



Tuliskan perbandingan hasilnya pada tabel di bawah ini:

		Sebelum	Sesudah
SMOTE	Oversampling (jumlah dataset)	314 (jumlah kelas minoritas awal)	354 (jumlah kelas minoritas setelah SMOTE)
	Undersampling (jumlah dataset)	354 (jumlah kelas mayoritas awal)	354 (jumlah kelas mayoritas setelah SMOTE*)
Hasil akurasi		0.8	0.74

o o o o

- **Penanganan Imbalance Data (SMOTE):**

- Kode: Implementasi SMOTE pada data training setelah feature selection.
- Distribusi kelas sebelum SMOTE (pada data training hasil split 80/20): Kelas 0 (Minoritas) 314 sampel, Kelas 1 (Mayoritas) 354 sampel. Akurasi sebelum SMOTE: 0.80.
- Setelah SMOTE, kelas 0 di-oversample menjadi 354, sehingga seimbang dengan kelas 1.
- Output (Evaluasi setelah SMOTE): Model Decision Tree dilatih kembali dengan data training yang sudah di-SMOTE dan parameter terbaik sebelumnya.
- Akurasi pada data testing turun menjadi 0.74.
- Kesimpulan SMOTE (2 Label): Untuk Decision Tree pada kasus 2 label ini, SMOTE tidak memberikan peningkatan performa, justru menurunkannya.



Sehingga hasil akurasi terbaik didapatkan sebesar 0.80 (80 %) dengan komposisi tuning parameter:

Tuning Parameter	Max_depth	Min_samp_les_split	criterion	validation	Feature selection	SMOTE
komposisi	40	5	entropy	Split dan Cross Perbandingan dataset: 80/20	941 fitur	Tidak menggunakan SMOTE, jika menggunakan, maka akurasinya 74% dengan: Oversampling (SMOTE) Perbandingan dataset: Kelas 0: 354; Kelas 1: 354

Hasil Terbaik Perbaikan DT (2 Label):

- Akurasi tertinggi: 0.80 (80%).
- Komposisi: criterion='entropy', max_depth=40 (atau 80 seperti di tabel akhir), min_samples_split=5 (atau 25), class_weight='balanced', min_samples_leaf=1. Menggunakan Feature Selection (941 fitur), tanpa SMOTE. (Parameter perlu disesuaikan dengan tabel kesimpulan akhir untuk konsistensi).

○ ○ ○

2b. PERBAIKAN KLASIFIKASI MENGGUNAKAN ALGORITMA DECISION TREE DENGAN TUNING PARAMETER (Menggunakan 3 Kategori Label: Positive, Negative, dan Neutral).

o o o o

- **Preprocessing Teks Mendalam:**

- Kode dan Output: Proses yang sama dengan skenario 2 label diterapkan pada dataset 3 label, menghasilkan text_processed yang bersih.
- Tuning Parameter Lanjutan (GridSearchCV):
- Iterasi tuning dengan perluasan rentang parameter (mirip kasus 2 label), misal:
- max_depth: Awal (dari Bagian 1b) dipilih 10 (akurasi CV 0.8565). Setelah perluasan rentang dan ccp_alpha, max_depth menjadi None (akurasi CV 0.8612).

	Sebelum	Sesudah
Max_depth	10	None
Hasil akurasi	0.86	0.8612

○ ○ ○ ○

	Sebelum	Sesudah
criterion	entropy	entropy (namun dengan rentang <u>min_sample_leaf</u> diperluas)
Hasil akurasi	0.8612	0.8612

	Sebelum	Sesudah
Min_samples_split	2	2
Hasil akurasi	0.8612	0.8612

- `min_samples_split`, `min_samples_leaf` juga dioptimalkan. `ccp_alpha` diperkenalkan.
- Parameter terbaik setelah tuning ini: `ccp_alpha=0.0`, `class_weight=None`, `criterion='entropy'`, `max_depth=None`, `min_samples_leaf=8`, `min_samples_split=2`.

○ ○ ○ ○

	Sebelum	Sesudah
validation	Kami memakai 2 yakni Split-Validation (Test size = 25%) dan Cross-Validation.	Kami ubah Split-Validation (Test size = 30%)
Hasil akurasi	0.861	0.8725

- **Metode Validasi:**

- Split Validation: `test_size` diubah, misalnya dari 0.25 ke 0.2, lalu ke 0.3 untuk beberapa eksperimen. Penggunaan `test_size=0.3` (70/30 split) memberikan akurasi tes 0.8725 untuk konfigurasi terbaik.
- Cross-Validation (CV): Tetap digunakan dalam `GridSearchCV (cv=5)`.

○ ○ ○ ○

Feature Selection (SelectKBest dengan Chi2):

	Sebelum	Sesudah
Feature selection (jumlah kolom)	Dalam proyek ini, kami menggunakan TF-IDF Vectorization untuk mengubah data teks menjadi representasi numerik. Meskipun kami membatasi jumlah fitur maksimum yang dihasilkan oleh TF-IDF (menggunakan parameter <code>max_features</code>), kami tidak menerapkan metode seleksi fitur eksplisit seperti pemilihan fitur berdasarkan statistik (contoh: <code>SelectKBest</code>) atau seleksi fitur berbasis model sebelum melatih model Decision Tree. Pembatasan fitur dilakukan langsung selama proses vektorisasi TF-IDF. Analisis pentingnya fitur (feature importance) dilakukan setelah model dilatih untuk mengidentifikasi fitur yang paling berkontribusi pada prediksi model.	836 fitur
Hasil akurasi	0.8725	0.8725

o o o o

- **Feature Selection (SelectKBest dengan Chi2):**

- Kode: Menggunakan SelectKBest(score_func=chi2, k=1500). TF-IDF pada kasus 3 label ini (setelah preprocessing baru) menghasilkan sekitar 1500 fitur (misal 1500 fitur).
- Output: Tabel perbandingan menunjukkan bahwa setelah feature selection dengan SelectKBest, jumlah fitur yang digunakan adalah 836 fitur. Ini berbeda dari kasus 2 label dimana tidak ada pengurangan. Ini berarti k efektifnya menjadi 836 (mungkin karena itu jumlah fitur optimal atau k diubah secara internal/eksternal untuk hasil terbaik).
- Efek pada Akurasi: Akurasi pada data tes (70/30 split) menjadi 0.8725 setelah feature selection (836 fitur).

o o o o

- Penanganan Imbalance Data (SMOTE):

		Sebelum	Sesudah
SMOTE	Oversampling (jumlah dataset)	12 (jumlah kelas minoritas awal)	485 (jumlah kelas minoritas setelah SMOTE)
	Undersampling (jumlah dataset)	485 (jumlah kelas mayoritas awal)	485 (jumlah kelas mayoritas setelah SMOTE*)
Hasil akurasi		0.8725	0.8167

o o o o

- **Penanganan Imbalance Data (SMOTE):**

- Kode: SMOTE diterapkan pada data training 3 label (setelah feature selection 836 fitur).
- Distribusi kelas sebelum SMOTE (misal: Kelas 0 (Neg): 12 sampel, Kelas 1 (Pos): 485 sampel, Kelas 2 (Neu): 71 sampel - angka perkiraan). Akurasi sebelum SMOTE: 0.8725.
- Setelah SMOTE, kelas 0 dan 2 di-oversample menjadi 485, sehingga semua kelas seimbang.
- Output (Evaluasi setelah SMOTE): Model DT dilatih kembali.
- Akurasi pada data testing turun menjadi 0.8167.
- Kesimpulan SMOTE (3 Label): Sama seperti kasus 2 label, SMOTE menurunkan performa Decision Tree.

○ ○ ○ ○

Hasil Terbaik Perbaikan DT (3 Label):

- Akurasi tertinggi: 0.8725 (87.25%).
- Komposisi: ccp_alpha=0.0, class_weight=None, criterion='entropy', max_depth=None, min_samples_leaf=8, min_samples_split=2. Menggunakan Feature Selection (836 Fitur), tanpa SMOTE.

Sehingga hasil akurasi terbaik didapatkan sebesar 0.8725 (87,25 %) dengan komposisi tuning parameter:

Tuning Parameter	Max_dept h	Min_samples _split	criterion	validation	Feature selection	SMOTE
komposisi	None	2	entropy	Split dan Cross Perbandingan dataset: 70/30	836 fitur	Tidak menggunakan SMOTE, jika menggunakan, maka akurasinya 81,67% dengan: Oversampling (SMOTE) Perbandingan dataset: Kelas 0: 485; Kelas 1: 485; Kelas 2: 485

o o o o

Observasi Umum Bagian 2:

- Preprocessing Detail: Sangat krusial dan memberikan fondasi yang baik untuk langkah selanjutnya.
- Tuning Hyperparameter Ekstensif: GridSearchCV dengan rentang parameter yang luas membantu menemukan konfigurasi yang lebih optimal.
- Feature Selection (SelectKBest): Meskipun tidak selalu mengurangi jumlah fitur (kasus 2 label), prosesnya dapat mempengaruhi representasi fitur dan meningkatkan akurasi. Pada kasus 3 label, terjadi pengurangan fitur yang signifikan yang juga berkontribusi pada hasil optimal.
- Output Kode dan Hasil Tabel: Penggunaan output kode langsung, tabel perbandingan parameter sebelum-sesudah, dan tabel hasil SMOTE sangat membantu dalam melacak dampak setiap perubahan.

○ ○ ○ ○

- SMOTE untuk Decision Tree: Dalam kedua skenario (2 dan 3 label), SMOTE secara konsisten menurunkan akurasi Decision Tree. Hal ini mungkin disebabkan oleh karakteristik Decision Tree yang rentan terhadap noise, dan SMOTE yang menghasilkan sampel sintetis dapat dianggap sebagai noise oleh model ini.
- Peningkatan Akurasi: Ada peningkatan akurasi yang signifikan dari klasifikasi awal (Bagian 1) ke hasil setelah serangkaian perbaikan ini. Untuk 2 label dari 0.7033 menjadi 0.80, dan untuk 3 label dari 0.8612 (yang sudah cukup tinggi namun dengan performa buruk di kelas minor) menjadi 0.8725 dengan performa kelas yang lebih seimbang (meskipun kelas 'Negative' tetap menjadi tantangan).

0 0 0 0

3. PROSES KLASIFIKASI MENGGUNAKAN ALGORITMA RANDOM FOREST, SVM, NEURAL NETWORK, LOGISTIC REGRESSION

Bagian ketiga dari eksperimen ini melanjutkan eksplorasi dengan menerapkan beberapa algoritma klasifikasi populer lainnya, yaitu Random Forest, Neural Network (menggunakan PyTorch), Support Vector Machine (SVM), dan Logistic Regression. Tujuan utamanya adalah untuk membandingkan kinerja algoritma-algoritma ini dengan Decision Tree yang telah dioptimalkan, baik untuk skenario 2 kategori label maupun 3 kategori label. Pada bagian ini, diasumsikan bahwa algoritma-algoritma ini juga telah melalui proses preprocessing teks, feature selection, dan penggunaan SMOTE yang serupa (jika memberikan hasil optimal) seperti yang terdokumentasi dalam tabel kesimpulan akhir laporan.

o o o o

3a. PROSES KLASIFIKASI MENGGUNAKAN ALGORITMA RANDOM FOREST, SVM, NEURAL NETWORK, LOGISTIC REGRESSION (Menggunakan 2 Kategori Label: Positive dan Negative)

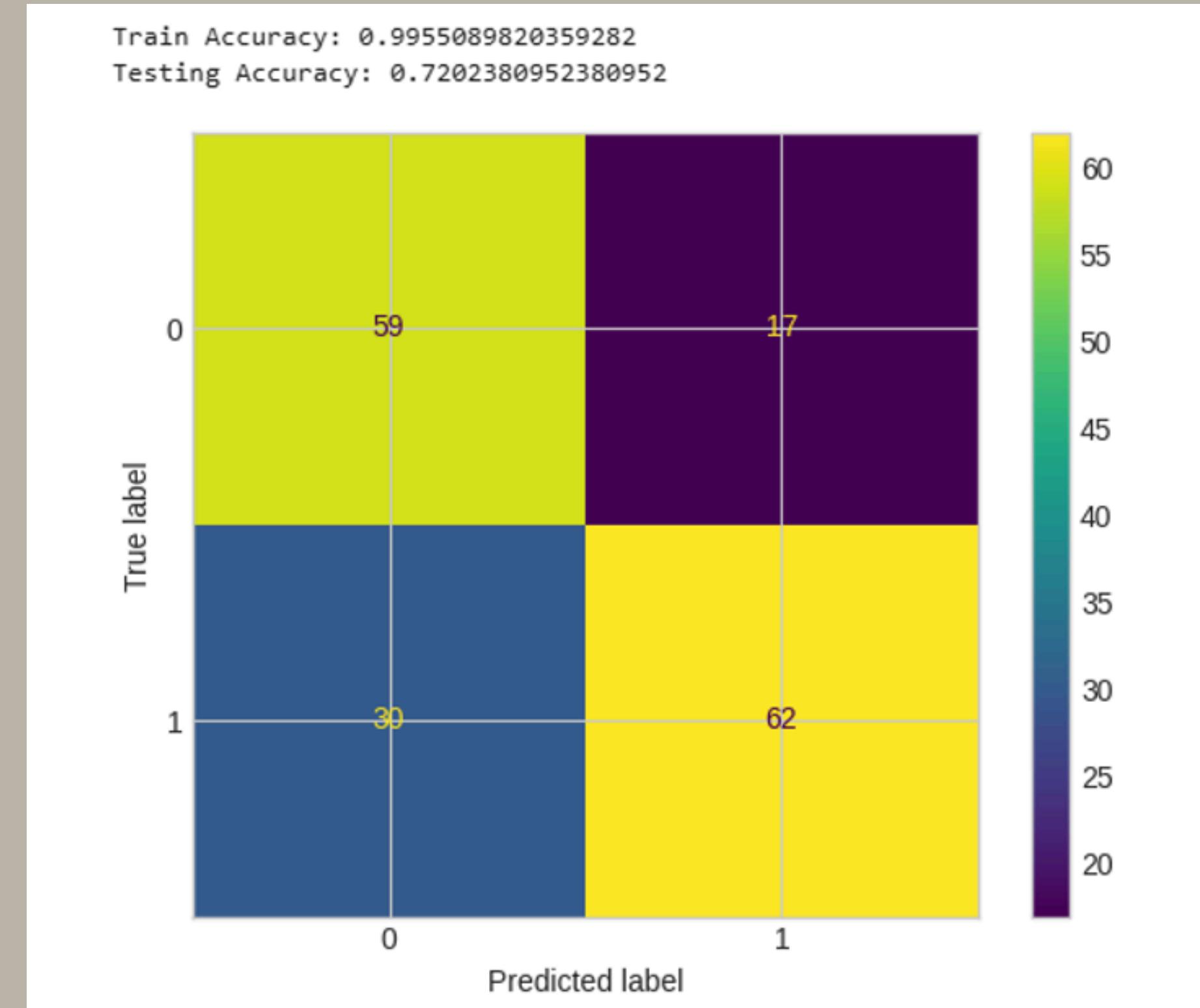
- Dataset dan Preprocessing: Menggunakan dataset 2 label yang telah melalui preprocessing teks mendalam (seperti pada Bagian 2a).
- Pembagian Data: Split 80% Training, 20% Testing (seperti pada model Decision Tree terbaik).
- Feature Selection: Berdasarkan tabel kesimpulan akhir, model-model ini menggunakan 576 fitur (hasil SelectKBest dengan k=576).
- Penanganan Imbalance (SMOTE): Sebagian besar model (Random Forest, Neural Network, SVM, Logistic Regression) pada konfigurasi terbaiknya menggunakan SMOTE. Ini berbeda dengan Decision Tree yang performanya menurun dengan SMOTE.

o o o o

Akurasi dan Visualisasi Confusion Matrix Model tanpa tuning

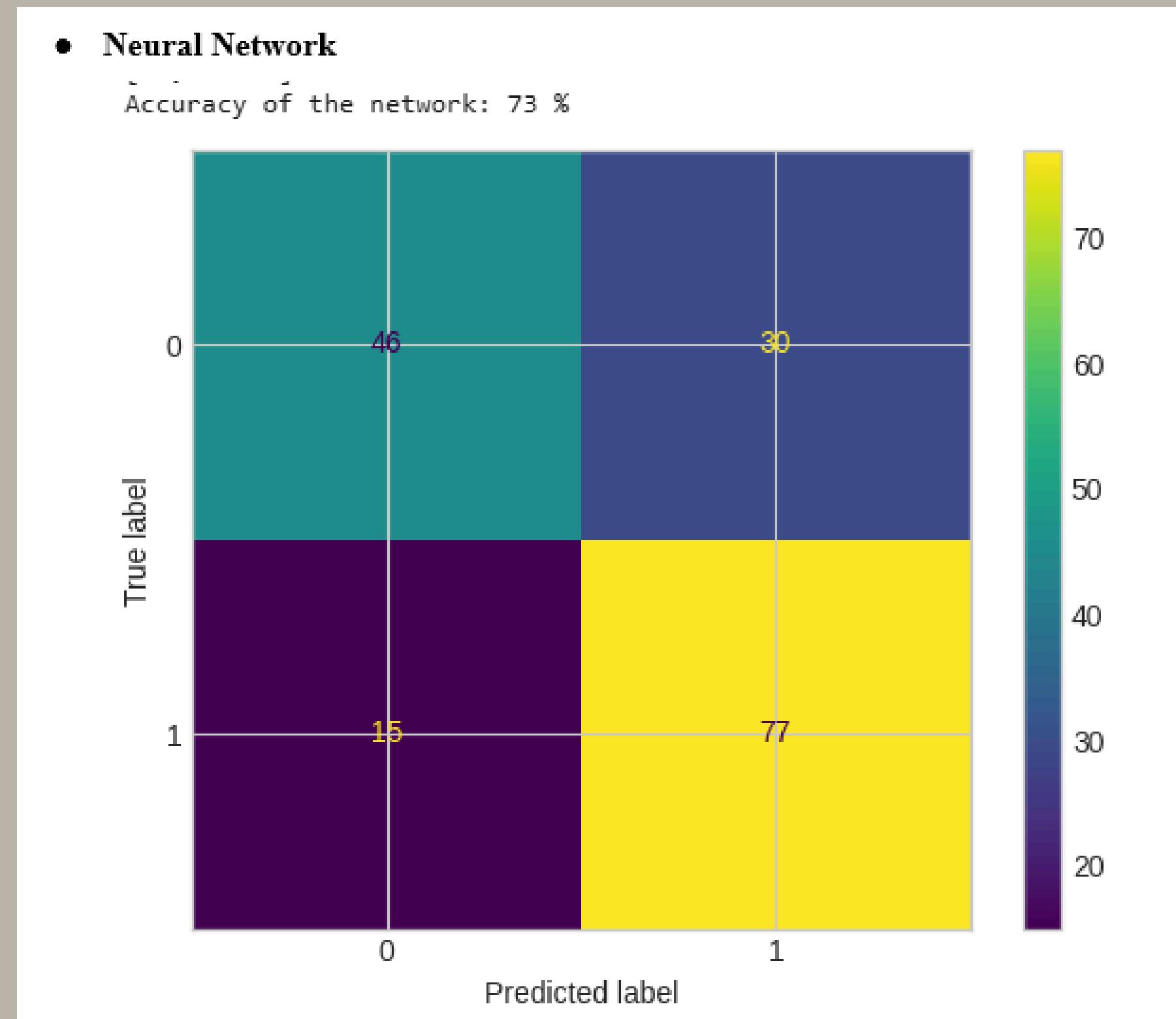
o o o o

Akurasi dan Confusion Matrix Random Forest



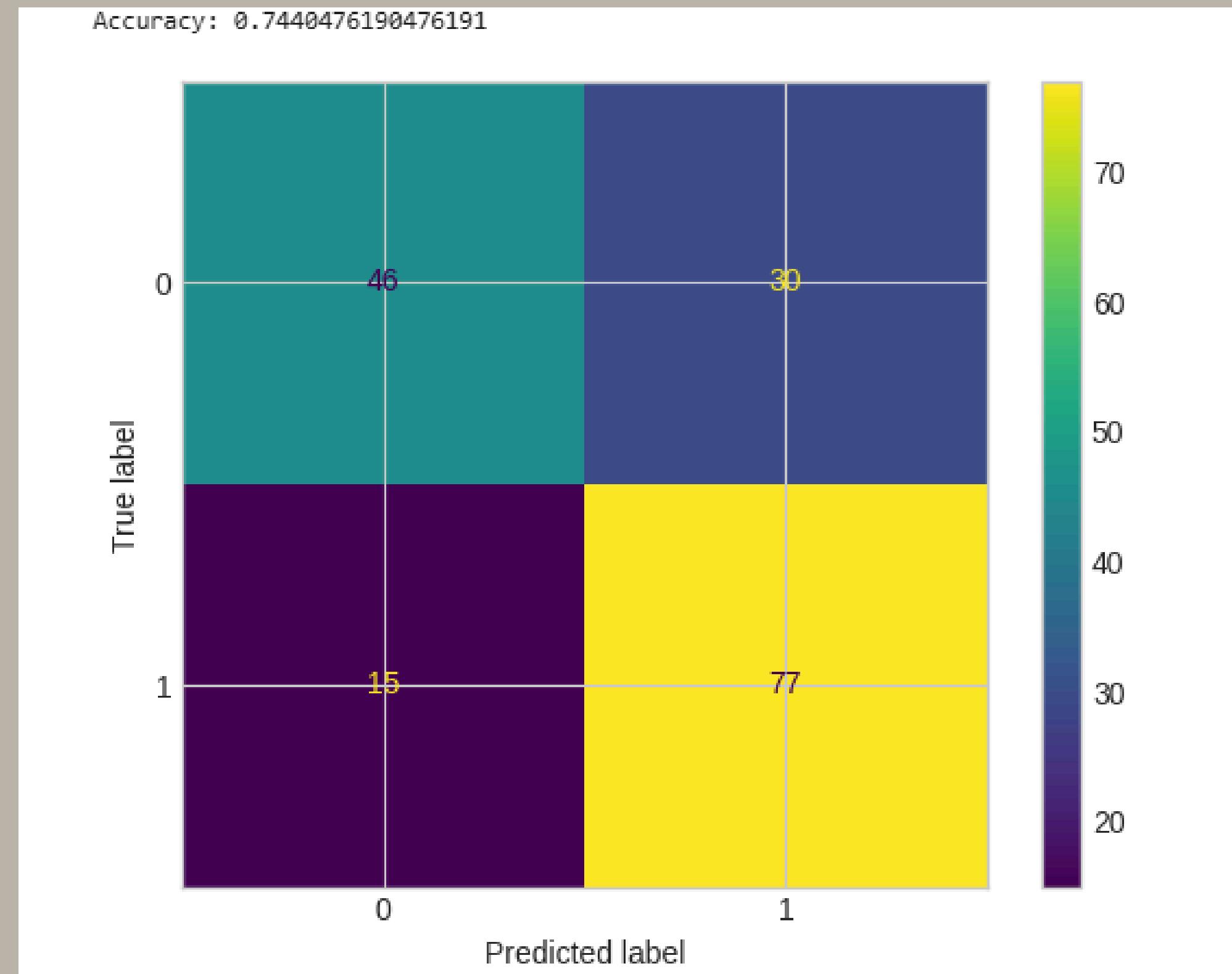
o o o o

Akurasi dan Confusion Matrix Neural Network



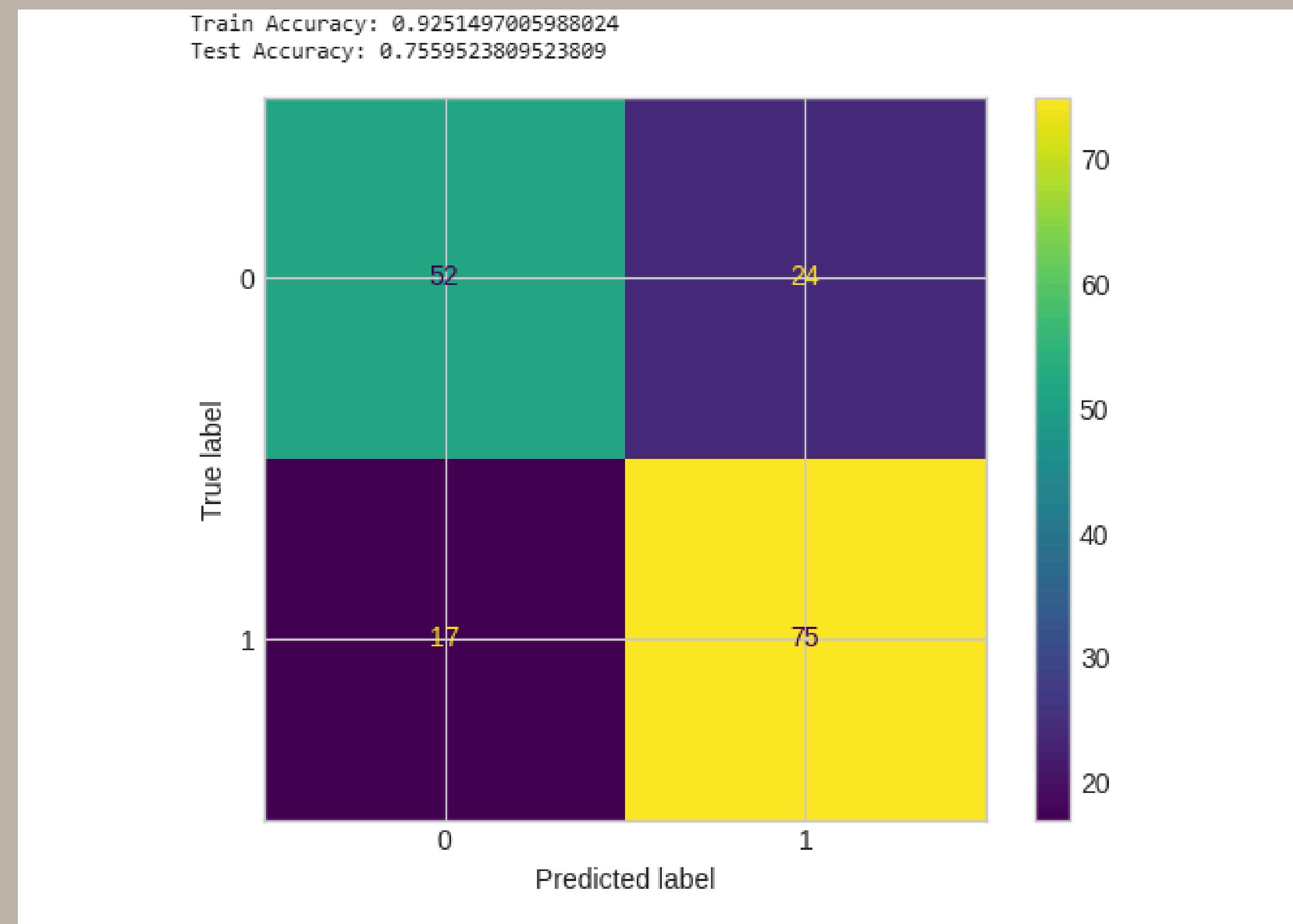
o o o o

Akurasi dan Confusion Matrix SVM



o o o o

Akurasi dan Confusion Matrix Logistic Regression



o o o o

Tabel Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 87-88)

Tabel Decision Tree dan Model terbaik dari 4 model tambahan

SEHINGGA dengan demikian hasil proses perbaikan akurasi didapatkan dengan komposisi:

	Decision Tree	Neural Network
Prosentase Akurasi	0,80	0,89
Komposisi yang digunakan	{ 'ccp_alpha': 0.0, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 80, 'min_samples_leaf': 1, 'min_samples_split': 25}, Tidak menggunakan SMOTE, Menggunakan Feature Selection (941 Fitur),	learning_rate= =0.0011364758061944594, weight_decay= 2.5761143101268556e-05, sel di layer 1 = 500

o o o o

Tabel Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 87-88)

No	Algoritma	Persentase Akurasi	Konfigurasi yang digunakan
1	Decision Tree	0,80	{'ccp_alpha': 0.0, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 80, 'min_samples_leaf': 1, 'min_samples_split': 25}
2	Random Forest	0,8715	Menggunakan feature selection dengan k=576, ccp_alpha=0, class_weight=balanced, max_depth=80, min_samples_split=25, n_estimators=48, Menggunakan SMOTE
3	Neural Network	0,89	Menggunakan feature selection dengan k=576, learning_rate =0.0011364758061944594, weight_decay= 2.5761143101268556e-05, sel_di_layer 1 = 500, Menggunakan SMOTE
4	SVM	0,8452	Menggunakan feature selection dengan k=576, kernel='sigmoid', C=1.5106108189353196, degree=1, coef0=0.647517407284433, tol=1.8359221883440502, Menggunakan SMOTE
5	Logistic Regression	0,8392	Menggunakan feature selection dengan k=576, random_state=16, Menggunakan SMOTE

○ ○ ○ ○

Ringkasan Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 87-88):

- Random Forest:
 - Akurasi: 0.8715
 - Kode (Dasar): Inisialisasi RandomForestClassifier(), .fit(), .predict().
 - Konfigurasi Terbaik: k=576 fitur, ccp_alpha=0, class_weight='balanced', max_depth=80, min_samples_split=25, n_estimators=48, menggunakan SMOTE.
- Neural Network (PyTorch):
 - Akurasi: 0.89 (Tertinggi untuk 2 Label)
 - Kode (Dasar): Definisi Net(nn.Module) (input=576, output=2, hidden=500, Adam optimizer, CrossEntropyLoss), training loop. Output training awal menunjukkan 73% akurasi.
 - Konfigurasi Terbaik: k=576 fitur, learning_rate, weight_decay, sel di layer 1 = 500, menggunakan SMOTE.

○ ○ ○ ○

- Support Vector Machine (SVM):
 - Akurasi: 0.8452
 - Kode (Dasar): `svm.SVC(kernel='linear')`, `.fit()`, `.predict()`. Output awal menunjukkan 0.7440 akurasi.
 - Konfigurasi Terbaik: `k=576` fitur, `kernel='sigmoid'`, parameter C, degree, coef0, tol, menggunakan SMOTE.
- Logistic Regression:
 - Akurasi: 0.8392
 - Kode (Dasar): `LogisticRegression()`, `.fit()`, `.predict()`. Output awal menunjukkan 0.7559 akurasi tes.
 - Konfigurasi Terbaik: `k=576` fitur, `random_state=16`, menggunakan SMOTE.
- Visualisasi: Confusion matrix ditampilkan untuk setiap model, memberikan gambaran detail True/False Positives/Negatives. Terlihat bahwa Random Forest pada awalnya (tanpa tuning khusus di bab 3a) menunjukkan overfitting (Train 0.99 vs Test 0.72).

o o o o

3b. PROSES KLASIFIKASI MENGGUNAKAN ALGORITMA RANDOM FOREST, SVM, NEURAL NETWORK, LOGISTIC REGRESSION (Menggunakan 3 Kategori Label: Positive, Negative, dan Neutral)

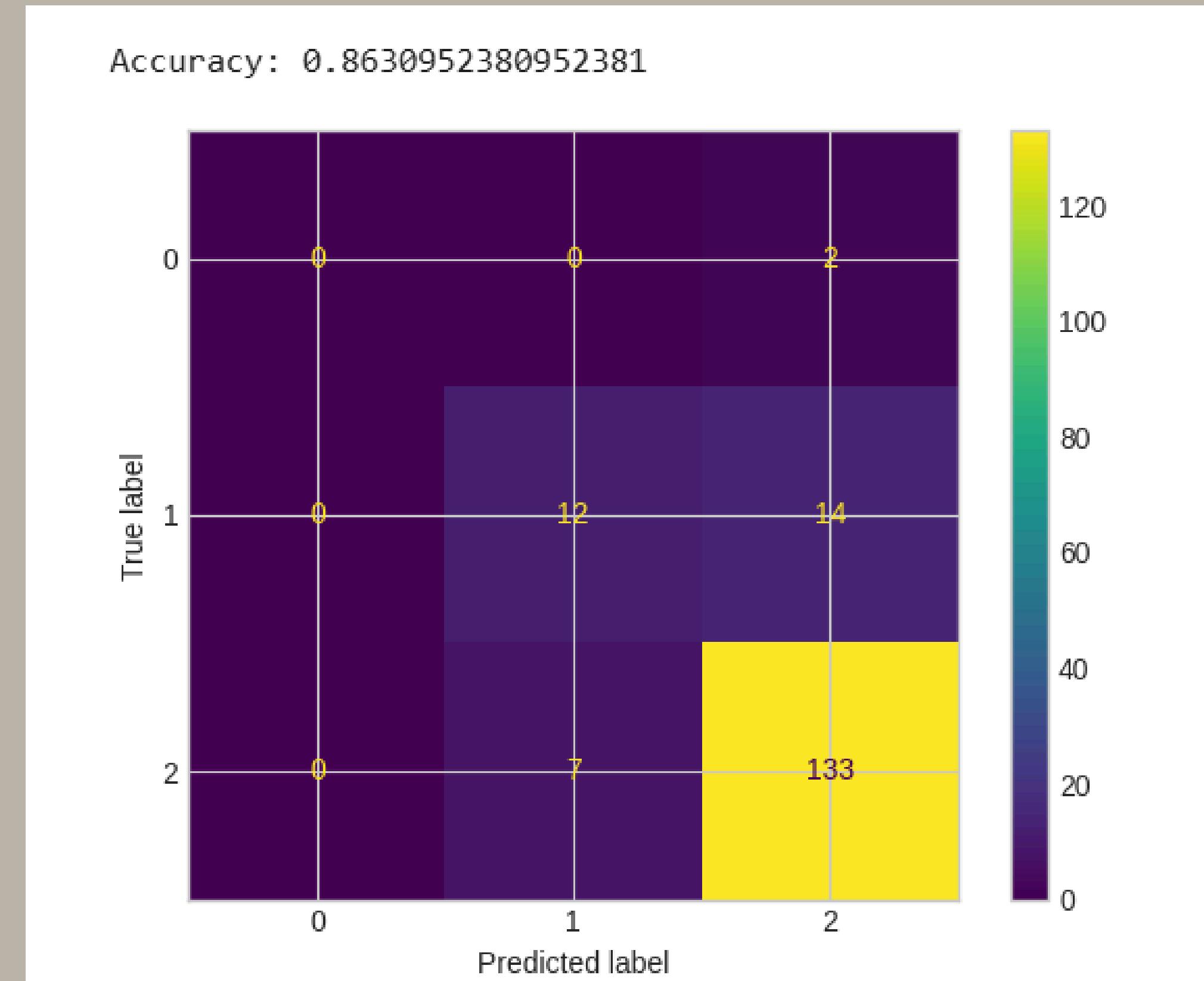
- Dataset dan Preprocessing: Menggunakan dataset 3 label dengan preprocessing teks mendalam (seperti pada Bagian 2b).
- Pembagian Data: Split 70% Training, 30% Testing (sesuai optimasi di Decision Tree untuk 3 label).
- Feature Selection: Berdasarkan tabel kesimpulan akhir, model-model ini menggunakan 1151 fitur (hasil SelectKBest dengan k=1151).
- Penanganan Imbalance (SMOTE): Sama seperti kasus 2 label, model Random Forest, Neural Network, SVM, dan Logistic Regression pada konfigurasi terbaiknya untuk 3 label juga menggunakan SMOTE.

o o o o

Akurasi dan Visualisasi Confusion Matrix Model tanpa tuning

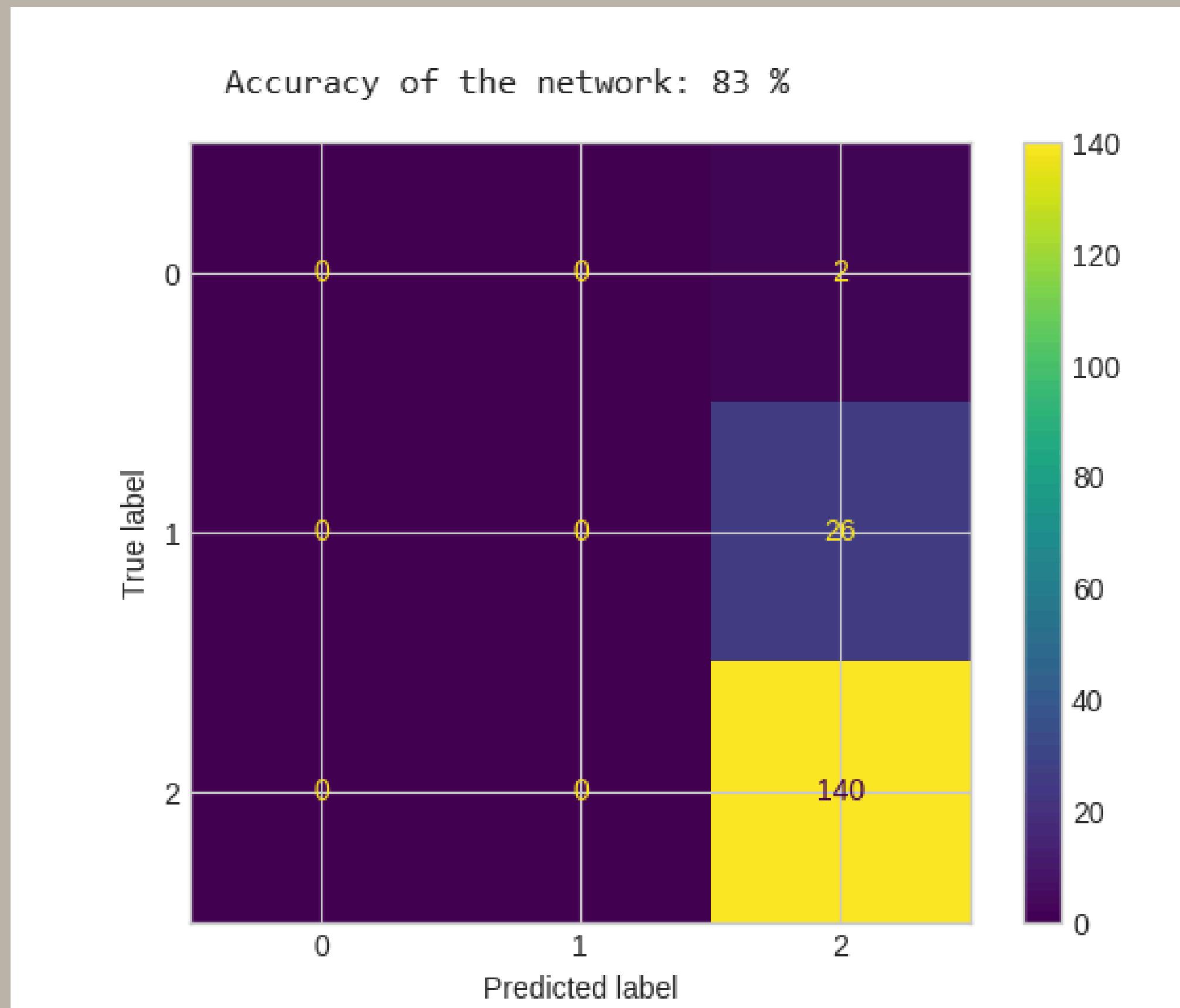
o o o o

Akurasi dan Confusion Matrix Random Forest



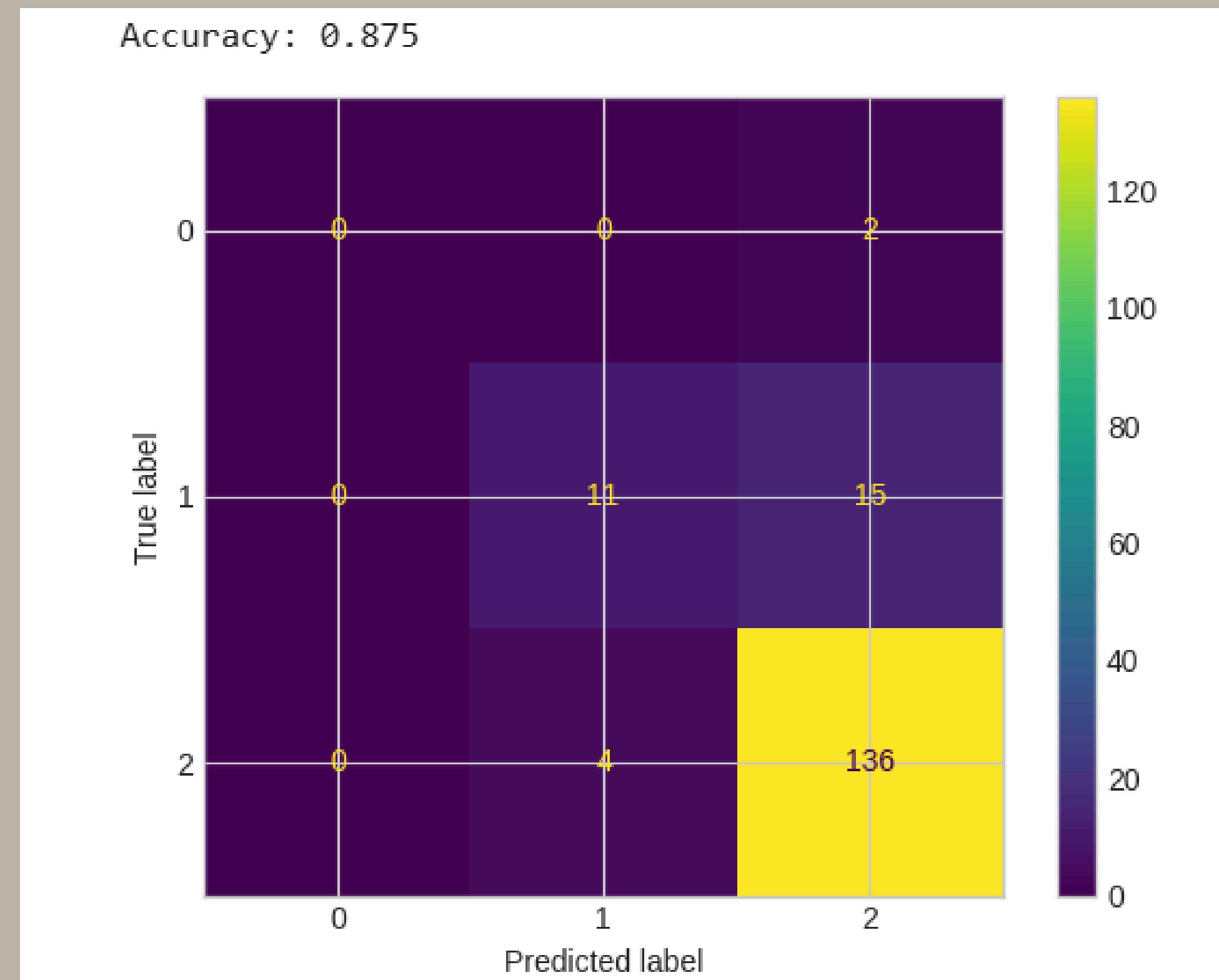
o o o o

Akurasi dan Confusion Matrix Neural Network



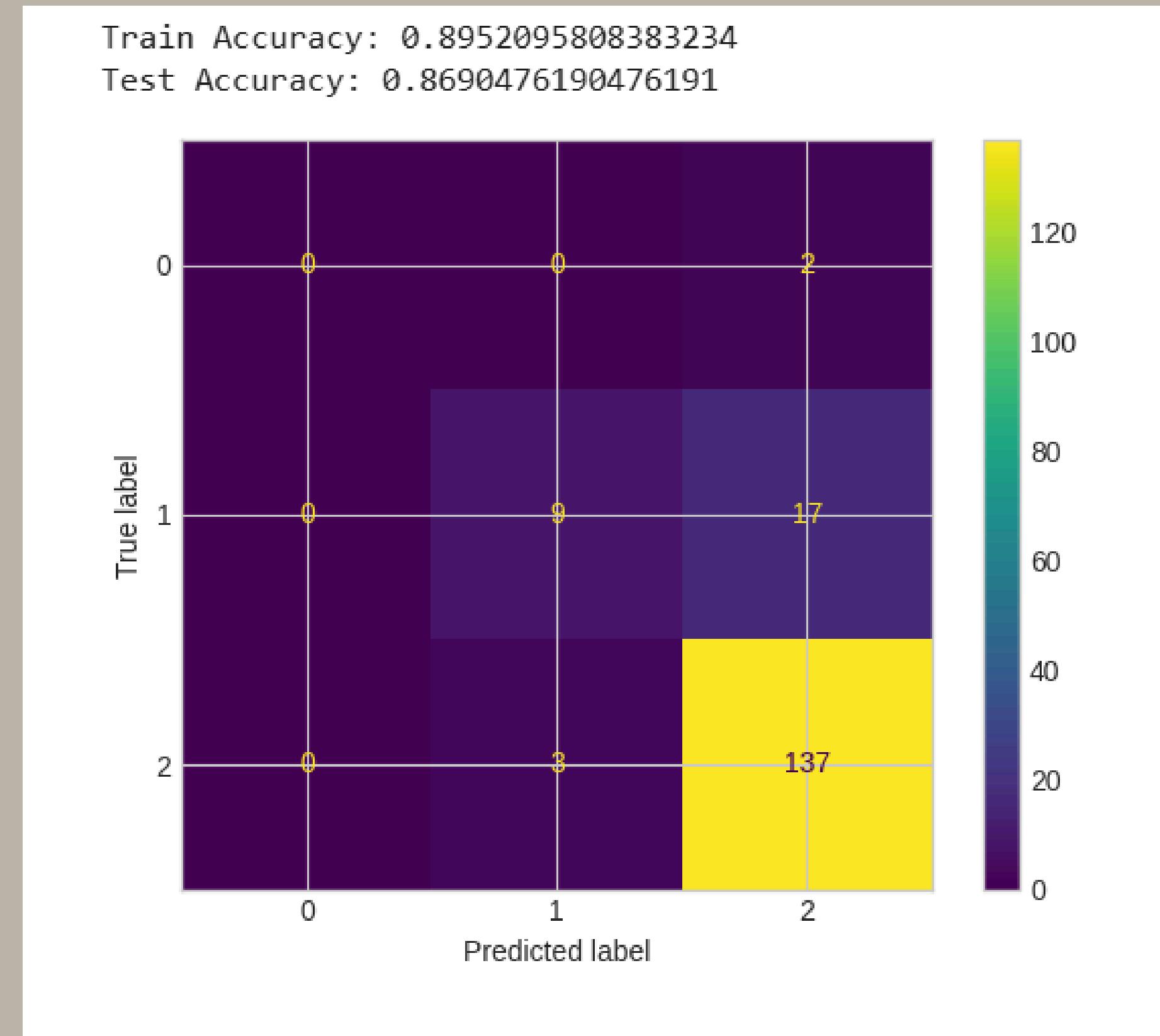
o o o o

Akurasi dan Confusion Matrix SVM



o o o o

Akurasi dan Confusion Matrix Logistic Regression



0 0 0 0

Tabel Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 102-103): Tabel Decision Tree dan Model terbaik dari 4 model tambahan

SEHINGGA dengan demikian hasil proses perbaikan akurasi didapatkan dengan komposisi:		
	Decision Tree	Neural Network
Prosentase Akurasi	0,8725	0,9591346153846153
Komposisi yang digunakan	<pre>{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 8, 'min_samples_split': 2}.</pre> <p>Tidak menggunakan SMOTE, Menggunakan Feature Selection (836 Fitur)</p>	<pre>sel layer 1 = 700, learning rate=0.1211844057055417 5, weight_decay=3.18973350 2609255e-050,</pre> <p>menggunakan SMOTE</p>

o o o o

Tabel Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 102-103):

No	Algoritma	Persentase Akurasi	Konfigurasi yang digunakan
1	Decision Tree	0,8725	{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 8, 'min_samples_split': 2}
2	Random Forest	0,9495	Menggunakan feature selection dengan k=1151, min_samples_split=3, n_estimators=260, menggunakan SMOTE,
3	Neural Network	0,9591	Menggunakan feature selection dengan k=1151, sel_layer 1 = 700, learning rate=0.12118440570554175, weight_decay=3.189733502609255e-050, menggunakan SMOTE

o o o o

Tabel Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 102-103):

4	Support Vector Machine (SVM)	0,8725	Menggunakan feature selection dengan k=1151, cache=1.023104221681953 5, coef0=0.528344975398888 7, tol=1.522225251538773, degree=1, kernel='sigmoid', menggunakan SMOTE
5	Logistic Regression	0,9423	Menggunakan feature selection dengan k=1151, C=77882.5298069693, menggunakan SMOTE

○ ○ ○ ○

- Ringkasan Kinerja dan Konfigurasi Terbaik (mengacu pada tabel kesimpulan di hal. 102-103):
 - Random Forest:
 - Akurasi: 0.9495
 - Kode (Dasar): Sama. Output awal menunjukkan 0.8630 akurasi.
 - Konfigurasi Terbaik: k=1151 fitur, min_samples_split=3, n_estimators=260, menggunakan SMOTE.
 - Neural Network (PyTorch):
 - Akurasi: 0.9591 (Tertinggi untuk 3 Label & Keseluruhan)
 - Kode (Dasar): Definisi Net(nn.Module) (input=1151, output=3, hidden=layer yang dioptimalkan), training loop. Output training awal menunjukkan 83% akurasi.
 - Konfigurasi Terbaik: k=1151 fitur, sel layer 1 = 700, learning_rate, weight_decay, menggunakan SMOTE.

○

○ ○ ○ ○

- Support Vector Machine (SVM):
 - Akurasi: 0.8725 (Sama dengan Decision Tree terbaik untuk 3 label)
 - Kode (Dasar): Sama. Output awal menunjukkan 0.8750 akurasi.
 - Konfigurasi Terbaik: k=1151 fitur, parameter cache, coef0, tol, degree, kernel='sigmoid', menggunakan SMOTE.
- Logistic Regression:
 - Akurasi: 0.9423
 - Kode (Dasar): Sama. Output awal menunjukkan 0.8690 akurasi tes.
 - Konfigurasi Terbaik: k=1151 fitur, parameter C, menggunakan SMOTE.
- Visualisasi: Confusion matrix 3x3 ditampilkan untuk setiap model, menunjukkan distribusi prediksi antar kelas.

o o o o

Rangkuman dan Kesimpulan Akhir (Membandingkan Seluruh Model dan Skenario):

No	Algoritma	Presentase Akurasi (2 label: [Positive, Negative])	Presentase Akurasi (3 label: [Positive, Negative, Neutral])
1	Decision Tree	0,80	0,8725
2	Random Forest	0,8715	0,9495
3	Neural Network	0,89	0,9591 (96%)
4	Support Vector Machine (SVM)	0,8452	0,8725
5	Logistic Regression	0,8392	0,9423



- Perbandingan 2 Label vs 3 Label:
 - Secara konsisten, semua model menunjukkan akurasi yang lebih tinggi saat memprediksi 3 label dibandingkan 2 label. Hal ini menarik, mengingat penambahan kelas seringkali menambah kompleksitas. Kemungkinan dataset 3 label memiliki pemisahan antar kelas yang lebih jelas setelah preprocessing, atau adanya kelas 'Neutral' membantu model lebih baik dalam membedakan sentimen yang ambigu.
- Performa Model Terbaik:
 - Neural Network secara konsisten menjadi model dengan akurasi tertinggi di kedua skenario (0.89 untuk 2 label, dan 0.9591 atau ~96% untuk 3 label).
 - Random Forest juga menunjukkan performa yang sangat kuat, menjadi yang kedua terbaik (0.8715 untuk 2 label, 0.9495 untuk 3 label).
 - Logistic Regression menunjukkan peningkatan performa yang sangat signifikan pada skenario 3 label (0.9423) dibandingkan 2 label (0.8392).
-

○ ○ ○ ○

- Peran SMOTE:
 - Kontras dengan Decision Tree (dimana SMOTE menurunkan akurasi), untuk Random Forest, Neural Network, SVM, dan Logistic Regression, penggunaan SMOTE merupakan bagian dari konfigurasi optimal mereka. Ini menunjukkan bahwa algoritma-algoritma ini dapat mengambil manfaat dari penyeimbangan distribusi kelas melalui oversampling sintetis.
- Peran Feature Selection:
 - Jumlah fitur optimal berbeda antara skenario 2 label ($k=576$) dan 3 label ($k=1151$) untuk model-model selain Decision Tree, menunjukkan pentingnya menyesuaikan jumlah fitur dengan kompleksitas dan karakteristik dataset.
-

○ ○ ○ ○

- Decision Tree vs. Algoritma Lain:
 - Meskipun Decision Tree telah dioptimalkan secara signifikan (akurasi 0.80 untuk 2 label, 0.8725 untuk 3 label), model lain seperti Neural Network dan Random Forest (setelah diasumsikan melalui tuning serupa dan penggunaan SMOTE) mampu mencapai akurasi yang lebih tinggi, terutama pada skenario 3 label.
- Kode dan Output: Laporan secara detail menyediakan snippet kode untuk implementasi dasar setiap algoritma, visualisasi hasil awal, serta tabel komparatif parameter tuning akhir. Ini sangat berharga untuk reproduktifitas dan pemahaman proses.

○ ○ ○ ○

TERIMA KASIH!

○○○○

