LAPORAN TUGAS KLASIFIKASI DAN KLASTERING
MATA KULIAH TEXT MINING & NATURAL LANGUAGE PROCESSING

Tim Penyusun:

1. <5231811022> < Lathif Ramadhan >
2. <5231811029> < Andini Angel M >
3. <5231811033> < Rama Panji N >
4. <5231811036> < Giffari Riyanda P>

PROGRAM STUDI SAINS DATA PROGRAM SARJANA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA

2025

KLASIFIKASI MENGGUNAKAN DECISION TREE

Link Proyek Notebook:

https://github.com/LatiefDataVisionary/text-mining-and-natural-language-processing-college-task/blob/main/notebooks/decision_tree_model.ipynb

1. Tulis/Screenshot Import Library yang digunakan

Jawab:

Berikut adalah library yang digunakan dalam proyek klasifikasi teks ini. Setiap library memiliki peran penting dalam tahapan analisis data, mulai dari pemrosesan data hingga pemodelan dan evaluasi.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_graphviz
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    ConfusionMatrixDisplay,
    roc_auc_score,
    roc_curve
)
import matplotlib.pyplot as plt
import seaborn as sns # Untuk visualisasi yang lebih menarik
import graphviz # Untuk visualisasi graphviz
import nltk # Untuk pra-pemrosesan teks lebih lanjut
from nltk.stem import WordNetLemmatizer # Untuk lemmatization
from nltk.corpus import stopwords as nltk_stopwords # Untuk custom stop words
```

```
# Download resource NLTK yang mungkin dibutuhkan
# Catch the LookupError directly when nltk.data.find fails
try:
    nltk.data.find('corpora/wordnet')
except LookupError:
    print("NLTK resource 'wordnet' not found. Downloading...")
    nltk.download('wordnet')
except Exception as e:
    print(f"An unexpected error occurred while checking/downloading 'wordnet':
    {e}")

try:
    nltk.data.find('corpora/omw-1.4')
except LookupError:
    print("NLTK resource 'omw-1.4' not found. Downloading...")
    nltk.download('omw-1.4') # WordNet multilingual resource
except Exception as e:
    print(f"An unexpected error occurred while checking/downloading 'omw-1.4':
    {e}")
```

```
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    print("NLTK resource 'stopwords' not found. Downloading...")
    nltk.download('stopwords')
except Exception as e:
    print(f"An unexpected error occurred while checking/downloading
    'stopwords': {e}")


# Pengaturan umum untuk plot agar lebih menarik
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_palette("viridis") # Atau palet lain seperti 'pastel', 'muted'
```

Berikut adalah daftar library Python yang di-import untuk menjalankan analisis sentimen menggunakan model Decision Tree ini. Setiap library memainkan peran spesifik dalam alur kerja machine learning, mulai dari pemuatan data, pra-pemrosesan, pemodelan, hingga evaluasi dan visualisasi.

- `pandas (pd)`: Digunakan untuk manipulasi dan analisis data tabular, terutama untuk memuat dan mengelola dataset.
- `numpy (np)`: Menyediakan dukungan untuk array dan matriks multidimensi besar, bersama dengan kumpulan fungsi matematika tingkat tinggi untuk beroperasi pada array ini.

- `sklearn.model_selection`:
    - `train_test_split`: Untuk membagi dataset menjadi set pelatihan dan pengujian.
    - `GridSearchCV`: Untuk melakukan pencarian hyperparameter terbaik secara sistematis menggunakan cross-validation.
    - `cross_val_score`: Untuk mengevaluasi performa model menggunakan cross-validation.
- `sklearn.feature_extraction.text`:
    - `TfidfVectorizer`: Untuk mengubah koleksi dokumen teks mentah menjadi matriks fitur TF-IDF.
- `sklearn.tree`:
    - `DecisionTreeClassifier`: Implementasi algoritma Decision Tree untuk klasifikasi.
    - `plot_tree`: Untuk memvisualisasikan pohon keputusan secara langsung menggunakan matplotlib.
    - `export_graphviz`: Untuk mengekspor pohon keputusan dalam format DOT, yang kemudian dapat dirender oleh Graphviz.
- `sklearn.metrics`:
    - `accuracy_score`: Untuk menghitung akurasi klasifikasi.
    - `confusion_matrix`: Untuk menghitung confusion matrix guna mengevaluasi akurasi klasifikasi.
    - `classification_report`: Untuk membangun laporan teks yang menunjukkan metrik klasifikasi utama (precision, recall, F1-score).
    - `ConfusionMatrixDisplay`: Untuk memvisualisasikan confusion matrix.
    - `roc_auc_score` dan `roc_curve`: Untuk mengevaluasi performa model klasifikasi biner menggunakan kurva ROC dan AUC.
- `matplotlib.pyplot (plt):` Library plotting 2D yang komprehensif untuk membuat visualisasi statis, animasi, dan interaktif.
- `seaborn (sns)`: Library visualisasi data Python berbasis matplotlib yang menyediakan antarmuka tingkat tinggi untuk menggambar grafik statistik yang menarik dan informatif.
- `graphviz`: Digunakan untuk merender output dari export_graphviz (file DOT) menjadi representasi grafis dari pohon keputusan.

- `nltk` (Natural Language Toolkit): Platform utama untuk membangun program Python untuk bekerja dengan data bahasa manusia.
    - `WordNetLemmatizer`: Untuk melakukan lemmatisasi (mengubah kata ke bentuk dasarnya/lemma).
    - `nltk_stopwords`: Menyediakan daftar stop words umum.

2. Tulis/screenshot import dataset yang digunakan

Jawab:

Dataset yang digunakan dalam analisis sentimen ini adalah ***ramadan_labeled_sentiment.csv*** (Dataset yang kami gunakan pada tugas-tugas sebelumnya). Dataset ini berisi tweet yang berkaitan dengan Ramadan beserta label sentimennya (positif atau negatif) dan skor sentimen lainnya.

```
path = 'https://raw.githubusercontent.com/LatiefDataVisionary/
text-mining-and-natural-language-processing-college-task/refs/heads/main/
datasets/ramadan_labeled_sentiment.csv'
df = pd.read_csv(path)

print(f"Dataset berhasil di-load dari: {path}")
print(f"Jumlah baris: {df.shape[0]}, Jumlah kolom: {df.shape[1]}")
```

```
Dataset berhasil di-load dari: https://raw.githubusercontent.com/LatiefDataVis
Jumlah baris: 836, Jumlah kolom: 8
```

3. Pemrosesan pembagian (splitdata) Data Training dan Data Testing yang digunakan

Dataset akan dibagi menjadi dua bagian: data training (untuk melatih model) dan data testing (untuk menguji performa model pada data yang belum pernah dilihat sebelumnya).

a. Tulis/screenshot codingnya

Jawab:

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.25,  # 25% data digunakan untuk testing
    random_state=42,  # Untuk reproduktifitas hasil
    stratify=y  # Mempertahankan proporsi kelas sentimen pada data training dan testing
)

print("Ukuran Data Setelah Pembagian:")
print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")
print(f"X_test shape: {X_test.shape}, y_test shape: {y_test.shape}")

print("\nDistribusi kelas pada data training (proporsi):")
print(y_train.value_counts(normalize=True))
print(y_train.value_counts())

print("\nDistribusi kelas pada data testing (proporsi):")
print(y_test.value_counts(normalize=True))
print(y_test.value_counts())
```

b. Tulis/screenshot hasilnya

Jawab:

```
Ukuran Data Setelah Pembagian:
X_train shape: (627, 937), y_train shape: (627,)
X_test shape: (209, 937), y_test shape: (209,)

Distribusi kelas pada data training (proporsi):
sentiment
1    0.532695
0    0.467305
Name: proportion, dtype: float64
sentiment
1    334
0    293
Name: count, dtype: int64

Distribusi kelas pada data testing (proporsi):
sentiment
1    0.535885
0    0.464115
Name: proportion, dtype: float64
sentiment
1    112
0     97
Name: count, dtype: int64
```

4. Pemodelan Decision Tree (perhitungan entropy sampai dengan information gain):

Model Decision Tree akan dilatih menggunakan data training. Scikit-learn secara internal menangani perhitungan entropy/information gain saat membangun pohon. Kita akan menggunakan GridSearchCV untuk menemukan hyperparameter terbaik untuk model Decision Tree.

a. Tuliskan/screenshotkan codingnya

Jawab:

Pelatihan Model dengan Hyperparameter Tuning

```
# Mendefinisikan grid parameter yang akan diuji, dengan rentang yang lebih luas
# dan penambahan class_weight untuk menangani potensi imbalance
param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30, 40, 50], # Perluas sedikit
    'min_samples_split': [2, 5, 10, 15, 20], # Perluas sedikit
    'min_samples_leaf': [1, 2, 4, 6, 8],    # Perluas sedikit
    'class_weight': [None, 'balanced'] # Untuk menangani imbalance kelas
    # 'ccp_alpha': [0.0, 0.001, 0.005, 0.01] # Cost-Complexity Pruning, bisa dicoba
}

dt_model = DecisionTreeClassifier(random_state=42)

grid_search_dt = GridSearchCV(
    estimator=dt_model,
    param_grid=param_grid_dt,
    cv=5, # 5-fold cross-validation
    scoring='accuracy', # Metrik evaluasi utama
    n_jobs=-1, # Gunakan semua core CPU
    verbose=1 # Tampilkan log
)

print("Memulai GridSearchCV untuk Decision Tree...")
grid_search_dt.fit(X_train, y_train)

best_dt_model = grid_search_dt.best_estimator_
print("\nGridSearchCV selesai.")

print("Parameter terbaik yang ditemukan untuk Decision Tree:")
print(grid_search_dt.best_params_)
```

```
print(f"\nSkor akurasi cross-validation terbaik untuk Decision Tree:
{grid_search_dt.best_score_:.4f}")

print("\nPenjelasan Terkait Pemilihan Split pada Decision Tree:")
print("Model Decision Tree yang dilatih menggunakan kriteria impurity untuk
menentukan split terbaik pada setiap node.")
print(f"Kriteria impurity yang dipilih oleh GridSearchCV untuk model terbaik
ini adalah: '{best_dt_model.criterion}'.")
if best_dt_model.criterion == 'entropy':
    print("   - Dengan kriteria 'entropy', model bertujuan untuk memaksimalkan
    Information Gain.")
    print("   - Information Gain mengukur pengurangan ketidakpastian setelah
    dataset di-split berdasarkan sebuah atribut.")
    print("   - Dihitung sebagai: IG(D, A) = Entropy(D) - Σ (|Dv| / |D|) *
    Entropy(Dv).")
elif best_dt_model.criterion == 'gini':
    print("   - Dengan kriteria 'gini', model bertujuan untuk meminimalkan Gini
    Impurity.")
    print("   - Gini Impurity mengukur probabilitas kesalahan klasifikasi jika
    sebuah elemen acak dari set labelnya ditebak secara acak sesuai distribusi
    label di set tersebut.")
    print("   - Dihitung sebagai: Gini(D) = 1 - Σ (pi)^2.")
print("Algoritma seperti ID3, C4.5 (menggunakan Information Gain atau Gain
Ratio), dan CART (menggunakan Gini Index) adalah implementasi dari konsep ini.")
print("Scikit-learn mengimplementasikan versi optimasi dari algoritma CART
untuk Decision Trees.")
```

b. Tuliskan/screenshotkan hasil

Jawab:

```
Memulai GridSearchCV untuk Decision Tree...
Fitting 5 folds for each of 600 candidates, totalling 3000 fits

GridSearchCV selesai.
Parameter terbaik yang ditemukan untuk Decision Tree:
{'class_weight': None, 'criterion': 'entropy', 'max_depth': 40, 'min_samples_leaf': 1, 'min_samples_split': 20}

Skor akurasi cross-validation terbaik untuk Decision Tree: 0.7177

Penjelasan Terkait Pemilihan Split pada Decision Tree:
Model Decision Tree yang dilatih menggunakan kriteria impurity untuk menentukan split terbaik pada setiap node.
Kriteria impurity yang dipilih oleh GridSearchCV untuk model terbaik ini adalah: 'entropy'.
   - Dengan kriteria 'entropy', model bertujuan untuk memaksimalkan Information Gain.
   - Information Gain mengukur pengurangan ketidakpastian setelah dataset di-split berdasarkan sebuah atribut.
   - Dihitung sebagai: IG(D, A) = Entropy(D) - Σ (|Dv| / |D|) * Entropy(Dv).
Algoritma seperti ID3, C4.5 (menggunakan Information Gain atau Gain Ratio), dan CART (menggunakan Gini Index) adalah implementasi dari konsep ini
Scikit-learn mengimplementasikan versi optimasi dari algoritma CART untuk Decision Trees.
```

5. Tampilkan hasil akurasi dan tabel confusion matrix nya

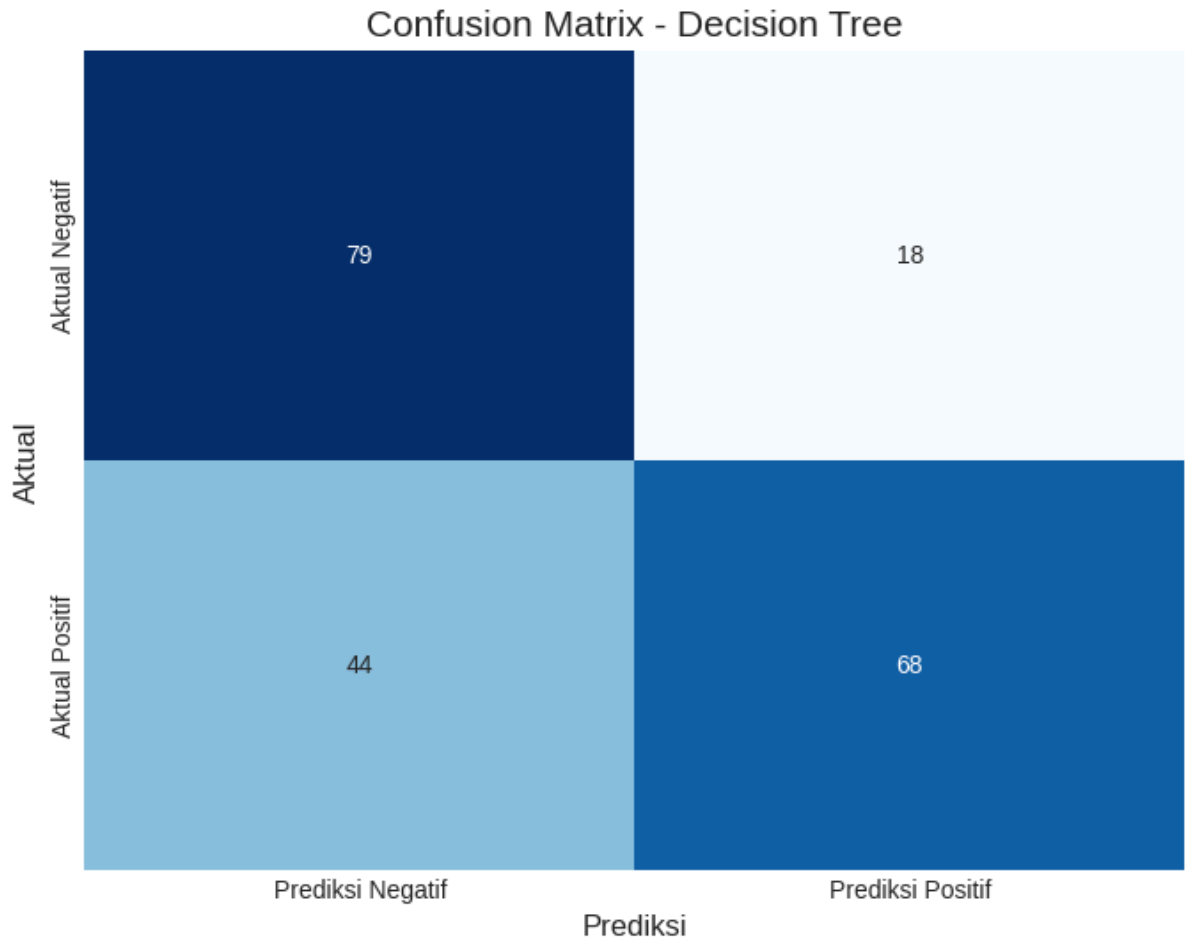
Setelah model dilatih, performanya akan dievaluasi menggunakan data testing.

Jawab:

```
[ ]  # Prediksi pada data testing
     y_pred_dt = best_dt_model.predict(X_test)
     y_pred_proba_dt = best_dt_model.predict_proba(X_test)[:, 1] # Probabilitas untuk kelas positif

     # Akurasi
     accuracy_dt = accuracy_score(y_test, y_pred_dt)
     print(f"Akurasi Model Decision Tree pada Data Testing: {accuracy_dt:.4f}")

     # ROC AUC Score
     try:
         roc_auc_dt = roc_auc_score(y_test, y_pred_proba_dt)
         print(f"ROC AUC Score Decision Tree: {roc_auc_dt:.4f}")
     except ValueError:
         print("ROC AUC Score tidak dapat dihitung (mungkin hanya satu kelas yang diprediksi).")


     # Laporan Klasifikasi
     print("\nLaporan Klasifikasi Decision Tree:")
     print(classification_report(y_test, y_pred_dt, target_names=['Negative (0)', 'Positive (1)']))
```

```
Akurasi Model Decision Tree pada Data Testing: 0.7033
ROC AUC Score Decision Tree: 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
```

```
# Confusion Matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Prediksi Negatif', 'Prediksi Positif'],
            yticklabels=['Aktual Negatif', 'Aktual Positif'])
plt.title('Confusion Matrix - Decision Tree', fontsize=15)
plt.ylabel('Aktual', fontsize=12)
plt.xlabel('Prediksi', fontsize=12)
plt.show()

print("\nInterpretasi Confusion Matrix:")
print(f"True Negatives (TN): {cm_dt[0,0]} - Tweet negatif yang diprediksi benar sebagai negatif.")
print(f"False Positives (FP): {cm_dt[0,1]} - Tweet negatif yang salah diprediksi sebagai positif (Type I Error).")
print(f"False Negatives (FN): {cm_dt[1,0]} - Tweet positif yang salah diprediksi sebagai negatif (Type II Error).")
print(f"True Positives (TP): {cm_dt[1,1]} - Tweet positif yang diprediksi benar sebagai positif.")
```

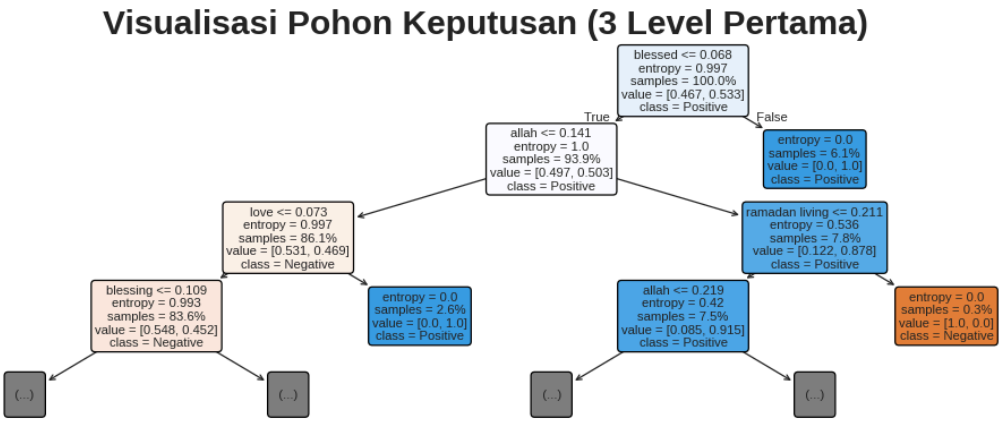



```
Interpretasi Confusion Matrix:
True Negatives (TN): 79 - Tweet negatif yang diprediksi benar sebagai negatif.
False Positives (FP): 18 - Tweet negatif yang salah diprediksi sebagai positif (Type I Error).
False Negatives (FN): 44 - Tweet positif yang salah diprediksi sebagai negatif (Type II Error).
True Positives (TP): 68 - Tweet positif yang diprediksi benar sebagai positif.
```

6. Tampilkan pohon keputusannya

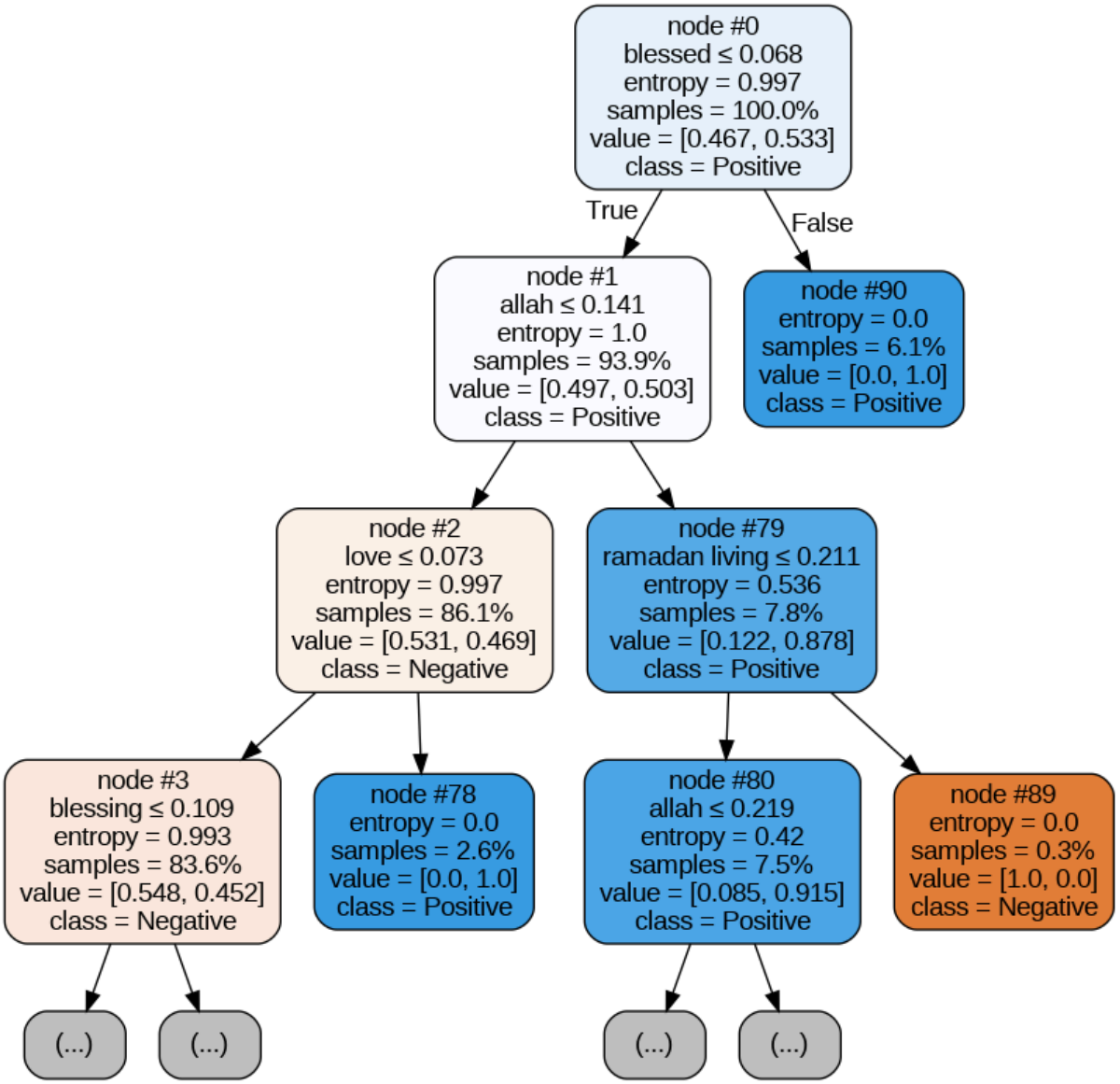
Jawab:

```
plt.figure(figsize=(15, 5), dpi=90) # Tingkatkan ukuran dan DPI untuk detail
plot_tree(
    best_dt_model,
    filled=True,
    feature_names=tfidf.get_feature_names_out(),
    class_names=['Negative', 'Positive'],
    max_depth=3, # Tampilkan 3 level pertama agar tidak terlalu ramai
    fontsize=9,
    proportion=True,
    rounded=True,
    precision=3, # Tingkatkan presisi
    impurity=True,
    label='all'
)
plt.title('Visualisasi Pohon Keputusan (3 Level Pertama)', fontsize=24, fontweight='bold')
plt.savefig("decision_tree_plot_tree.png", dpi=300, bbox_inches='tight') # Simpan gambar
plt.show()
print("Visualisasi dengan plot_tree telah ditampilkan dan disimpan sebagai 'decision_tree_plot_tree.png'")
```



Visualisasi dengan graphviz (Untuk Potensi Export):

```
dot_data = export_graphviz(
    best_dt_model,
    out_file=None,
    feature_names=tfidf.get_feature_names_out(),
    class_names=['Negative', 'Positive'],
    filled=True,
    rounded=True,
    max_depth=3, # Batasi kedalaman untuk Graphviz juga
    special_characters=True,
    proportion=True,
    impurity=True,
    node_ids=True, # Tambahkan ID node
    label='all'
)

graph = graphviz.Source(dot_data, format="png")
try:
    graph.render("decision_tree_graphviz") # Simpan sebagai 'decision_tree_graphviz.png'
    print("\nVisualisasi pohon keputusan dengan Graphviz telah dirender dan disimpan sebagai 'decision_tree_graphviz.png'")
    from IPython.display import Image
    display(Image(filename='decision_tree_graphviz.png'))
except graphviz.backend.execute.CalledProcessError as e:
    print(f"Error saat merender Graphviz: {e}")
    print("Pastikan Graphviz terinstal dan ada di PATH sistem Anda.")
    print("Anda masih bisa mengonversi file 'decision_tree_graphviz.dot' (jika ada) secara manual menggunakan Graphviz.")
```


node #0
blessed ≤ 0.068
entropy = 0.997
samples = 100.0%
value = [0.467, 0.533]
class = Positive
True
False
node #1
allah ≤ 0.141
entropy = 1.0
samples = 93.9%
value = [0.497, 0.503]
class = Positive
node #90
entropy = 0.0
samples = 6.1%
value = [0.0, 1.0]
class = Positive
node #2
love ≤ 0.073
entropy = 0.997
samples = 86.1%
value = [0.531, 0.469]
class = Negative
node #79
ramadan living ≤ 0.211
entropy = 0.536
samples = 7.8%
value = [0.122, 0.878]
class = Positive
node #3
blessing ≤ 0.109
entropy = 0.993
samples = 83.6%
value = [0.548, 0.452]
class = Negative
node #78
entropy = 0.0
samples = 2.6%
value = [0.0, 1.0]
class = Positive
node #80
allah ≤ 0.219
entropy = 0.42
samples = 7.5%
value = [0.085, 0.915]
class = Positive
node #89
entropy = 0.0
samples = 0.3%
value = [1.0, 0.0]
class = Negative
(...)
(...)
(...)
(...)

KLASTERING MENGGUNAKAN K-MEANS

1. **Tulis/Screenshot Import Library yang digunakan**

   Jawab:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.decomposition import PCA
import re, seaborn as sns
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import ListedColormap
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

2. **Tulis/screenshot import dataset yang digunakan**

   Jawab:

```
dm = pd.read_csv("https://raw.githubusercontent.com/LatiefDataVisionary/text-mining-and-natural-language-processing-college-task/refs/heads/main/datasets/ramadan_labeled_sentiment.csv", usecols=["tweet_clean", "sentiment"])
dm.columns = ["tweet_clean", "sentiment"]
dm.head(10)
```

3. **Perhitungan jumlah klaster**

   a. **Tulis/screenshot codingnya**

      Jawab:

```
[ ]  K = range(2, 8)
     fits = []
     score = []

     for k in K:
         # train model gunain k didalam loop
         model = KMeans(n_clusters = k, random_state = 0, n_init='auto').fit(X_train)

         # append model ke fits
         fits.append(model)

         # Append score inertia ke score
         score.append(model.inertia_)
         # score_sil.append(silhouette_score(X_train, model.labels_, metric='euclidean'))
```

```
# Ngeplot k vs score
sns.set(style='whitegrid')
sns.lineplot(x=K, y=score, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Score')
plt.title('Score vs Number of Clusters')
plt.show()
```

b. Tulis/screenshot hasilnya

Jawab:

4. **Pembuatan grafik dalam bentuk scatter plot berdasarkan centroid**

   a. **Tuliskan/screenshotkan codingnya**

   Jawab:

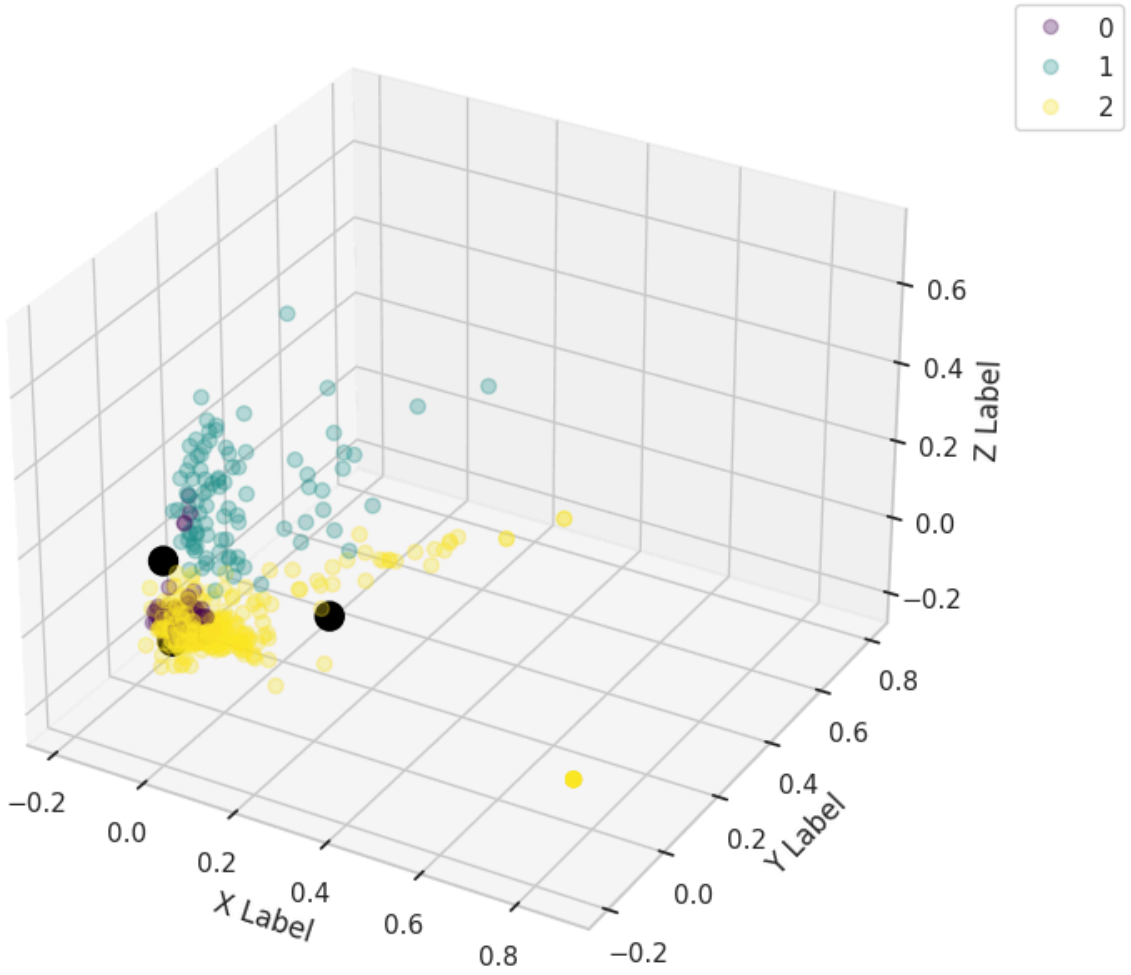
```
# Memilih kolom pca
pca = PCA(n_components=3)

# Membuat df menggunakan koordinat cluster center
cluster = fits[1].cluster_centers_
df_clus = pca.fit_transform(cluster)

# Mendapatkan nama kolom
col_pca = pca.get_feature_names_out(input_features=None)

# Menjadikan pca menjadi datatframe
pca_cl = pd.DataFrame(df_clus, columns=col_pca)
pca_cl
```

```python
# generate data
x = df_pca['pca0']
y = df_pca['pca1']
z = df_pca['pca2']

xc = pca_cl['pca0']
yc = pca_cl['pca1']
zc = pca_cl['pca2']

# axes instance
fig = plt.figure(figsize=(6,6))
ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

# dapetin colormap dari seaborn
cmap = ListedColormap(sns.color_palette("husl", 256).as_hex())

# plot
ax.scatter(xc, yc, zc,c="black",s=150,label="Centers",alpha=1)
sc = ax.scatter(x, y, z, s=40, c=fits[1].labels_, marker='o', cmap='viridis', alpha=0.3)
# ax.scatter(, c='red', s=150, marker='X', label='Cluster Centers')

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

# legend
plt.legend(*sc.legend_elements(), bbox_to_anchor=(1.05, 1), loc=2)
```

b. Tuliskan/screenshotkan hasilnya

Jawab:

	pca0	pca1	pca2
0	-0.184392	0.078457	1.464114e-16
1	0.195790	0.065508	1.464114e-16
2	-0.011398	-0.143965	1.464114e-16



5. **Pembuatan grafik dalam bentuk wordcloud berdasarkan centroid**

   a. **Tulis/screenshot codingnya**

   Jawab:

```
[125] # Menaruh label kluster
      dfx_train['label'] = fits[1].labels_
```

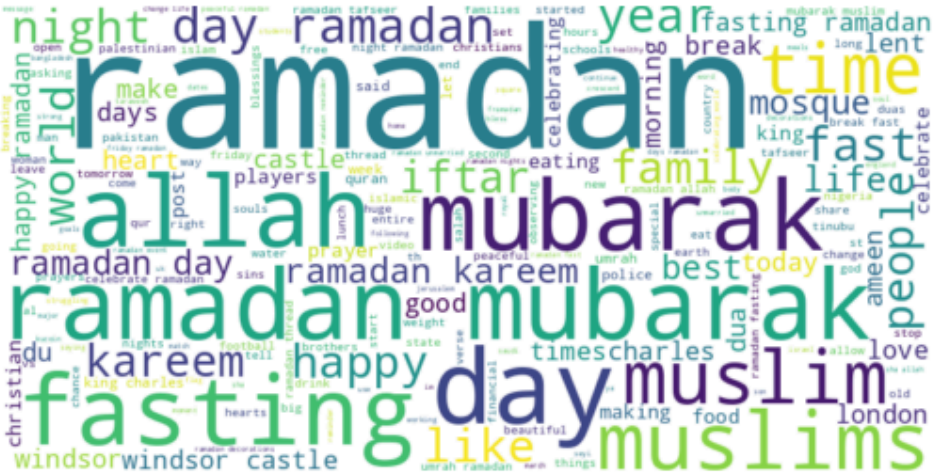
```
[129] # Membuat plot
      for cent in range(3):
        filt = (dfx_train['label'] == cent)
        term_sums = np.sum(dfx_train[filt], axis=0)
        feature_names = dfx_train[filt].drop(columns=['label']).columns

        term_freq = {feature_names[i]: term_sums[i] for i in range(len(feature_names))}

        # Membuat wordcloud
        wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(term_freq)

        # Display gambarnya
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis('off')
        plt.title(f'Cluster {cent+1}')
        plt.savefig(f'wordcloud_cluster_{cent}.png')
        plt.show()
```

b. Tulis/screenshot hasilnya

Jawab:

Cluster 2

Cluster 3



6. **Hitung klaster masing-masing record data**

   a. **Tuliskan/screenshotkan codingnya**

   Jawab:

```
K = range(2, 8)
fits = []
score = []

for k in K:
    # train model gunain k didalam loop
    model = KMeans(n_clusters = k, random_state = 0, n_init='auto').fit(X_train)

    # append model ke fits
    fits.append(model)

    # Append score inertia ke score
    score.append(model.inertia_)
```

```
[130] dfx_train = pd.DataFrame(X_train, columns=feature_names)

[131] # Menaruh label kluster
      dfx_train['label'] = fits[1].labels_
      dfx_train
```

   b. **Tuliskan/screenshotkan hasilnya**

   Jawab:

›rding	accounts	act	action	acts	additionally	...	ya allah	year	year old	years	yes	yesterday	york	young	zazzau	label
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
0.0	0.0	0.267967	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2