BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Bab ini berfokus pada analisis Perspektif pengguna *Mobile Banking* Sumsel Babel dengan menggunakan metode *K-Means* dan *Naïve Bayes*. Penelitian ini dilakukan dengan mengambil data ulasan aplikasi *Mobile Banking* Sumsel Babel di Play Store. Pengumpulan data pada penelitian ini dengan cara *scrapping* data ulasan pengguna di google play store. *Scraping* data dilakukan pada tanggal 30 April 2024.

4.2 HASIL PENELITIAN

Scraping data di lakukan pada tanggal 30 April 2024, dan mendapatkan hasil 2985 data ulasan pengguna.

4.2.1 Hasil Pengumpulan Data

Pengumpulan data menggunakan web scraping dengan menggunakan library *'google_play_scraper'*, untuk mengotomatiskan pengambilan ulasan pengguna dari *google play store*. Hasil dari *scrapping* data ini menghasilkan 2985 data dan terdapat 2 kolom yaitu kolom score dan content yang di simpan dalam bentuk csv untuk mempermudah dalam *preprosesing* data.

Tabel 2 Contoh Data Ulasan

No	Rating	Content
1	1	Mohon di perbaiki, masa karna melepaskan kartu sim card d hp dan d pasang lagi ,bangking ny minta aktivasi ulang dan ke blokir , cape bolak balik bank.
2	3	Secara default aplikasi ini membutuhkan perizinan akses ke kamera, kontak, lokasi, dan panggilan telepon. Agak gimana jadinyaðŸ¥¶.
3	5	Sudah baik untuk kelas bank daerah. Semoga kedepannya mobile banking bank sumselbabel bisa transaksi pembelian token listrik.

4	1	Selalu dan selalu terkendala "Aktivasi tidak tersedia saat ini" ribet amat ya ðŸ˜*f*
5	5	Aplikasi ini sangat membantu kemudahan transaksi. Semoga makin lancar jaya â˜•
6	2	Apaan sih ga jelas banget, tiap mau top up e wallet atau qris suka banget gagal dan duit kepotong, nunggu pengembalian lama banget pula , benerin dulu lah sistem nya , masa selalu aja kena begini
7	3	Sejujurnya sangat membantu, tetapi sayang tidak dilengkapi fitur keluhan atau komplain di aplikasinya. Sudah beberapa x transaksi pembelian atau pembayaran diganggu error tetapi saldo terpotong. Tolong diperhatikan hal ini.
8	5	Aplikasinya keren untuk sekelas BUMD. Warna menarik dan ramah Terhadap pengelihatan. Serta fitur fitur mudah diengerti. Walau terkadang sering ribet OTP. Tolong dong untung masalah mbanking terblokir bisa via telpon call center saja supaya lebih praktis. Terimakasih
9	1	Udah banyak update, kirain udah berubah. Ternyata tambah parah. Scan qris katanya gagal, tapi saldo kepotong.
10	3	Sinyal bagus kok susah masuk katanya jaringan anda gak stabil

Tabel 2 data ulasan yang diperoleh masih memiliki banyak ulasan yang tidak relevan untuk analisis sentimen. Beberapa ulasan mengandung banyak karakter yang tidak penting, sehingga perlu dilakukan *preprocessing* data untuk memastikan hasil analisis yang lebih tepat.

4.3 HASIL PREPROCESSING

Setelah mendapatkan data ulasan dari *google play store*, selanjutnya yaitu tahap *preprocessing* tahap ini digunakan untuk membersihkan dan pengolahan data yang diperlukan untuk diolah pada tahap selanjutnya. Berikut adalah tahap *preprocessing* data:

4.3.1 Case Folding

Case Folding yaitu proses mengubah karakter huruf kapital menjadi huruf kecil. Berikut kode yang digunakan dalam *case folding* .

Pada Gambar 4.1 menunjukan kode program yang menggunakan parameter case folding yang menerima parameter content dengan tujuan untuk mengubah semua kata yang ada pada dokumen menjadi huruf kecil seperti pada gambar 4.2.

```
def case_folding(text):
      lower_word = text.lower()
      return lower_word
df_data['case_folding']= df_data['content'].str.lower()

print(df_data['case_folding'])
```

Gambar 4.1 Case Folding

```
0       aplikasi sering keluar sendiri. baru mau sudah...
1       tolong di perbaiki lagi aplikasinya, hapir set...
2       ribet, aktivasi harus ke bank padahal bni mobi...
3       2024, transfer rempong, kalo lagi tf pembayara...
4       m-banking sering eror minta aktivasi ulang dan...
                              ...
2980                                          cukup keren
2981                                     mudah aplikasinya
2982    tidak bisam daftar aktivasi padahal sudah masu...
2983                                        susah download
2984                                                  sip
Name: case_folding, Length: 2985, dtype: object
```

Gambar 4.2 Hasil

4.3.2 Filtering

Langkah ini bertujuan untuk menghapus karakter-karakter yang tidak relevan dan membersihkan teks dari elemen-elemen yang tidak diperlukan. Berikut adalah tahap-tahap filtering dalam preprocessing data:

Special removal merupakan suatu proses pembersihan teks pada ulasan dengan cara menghapus karakter khusus dan simbol tertentu yang tidak diperlukan. Pada gambar 4.3 merupakan kode program *special removal*, yang

mana mana terdapat fungsi remove_play special yang bertujuan membersihkan teks ulasan.

```
def remove_play_special(case_folding):
    if isinstance(case_folding, float):
        return " "
    else:
      text = case_folding.replace('\\t'," ").replace('\\n'," ").replace('\\'," ")
      text = text.encode('ascii', 'replace').decode('ascii')
      text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
      return text.replace("http://", " ").replace("https://"," ")
df_data['clean'] = df_data['case_folding'].apply(remove_play_special)
```

Gambar 4.3 *Special removal*

1. *Number removal*

   Pada gambar 4.4 *number removal* digunakan untuk penghapusan angka yang terdapat dalam teks.

```
def remove_number(clean):
    text = re.sub(r"\d+", " ", clean)
    return text
df_data['clean'] = df_data['clean'].apply(remove_number)
```

Gambar 4.4 *Number removal*

2. *Punctuation Removal*

   Pada gambar 4.5 *punctuation removal* digunakan untuk delete karakter khusus seperti simbol mata uang, tanda pagar (#), tanda bintang (*), dan sebagainya yang tidak memiliki pengaruh signifikan dengan menggunakan fungsi remove_punc.

```
def remove_punc(clean):
    clean_spcl = re.compile('[/(){}\[\]\|@,;]')
    clean_symbol = re.compile('[^0-9a-z]')
    text = clean_spcl.sub(' ', clean)
    text = clean_symbol.sub(' ', clean)
    return text
df_data['clean'] = df_data['clean'].apply(remove_punc)
```

Gambar 4.5 *Punctuation Removal*

3. *Whitespaces Removal*

Pada gambar 4.6 terdapat fungsi yang diberi nama remove_whitespace yang digunakan untuk delete spasi yang terdapat di awal dan di akhir dalam sebuah kalimat.

```
def remove_whitespace(clean):
    corrected = str(clean)
    corrected = re.sub(r"//t",r"\t", corrected)
    corrected = re.sub(r"()\1+",r"\1", corrected)
    corrected = re.sub(r"(\n)\1+","\1", corrected)
    corrected = re.sub(r"(\r)\1+",r"\1", corrected)
    corrected = re.sub(r"(\t)\1","\1", corrected)
    return corrected.strip(" ")
df_data['clean'] = df_data['clean'].apply(remove_whitespace)
```

Gambar 4.6 Whitespaces Removal

4. *Single Char Removal*

Gambar 4.7 merupakan kode program untuk menghapus karakter tunggal atau karakter yang hanya terdiri dari satu huruf dalam teks data. Contoh karakter tunggal yang umum dihapus termasuk huruf yang berdiri sendiri (misalnya, "a", "b", "c").

```
def remove_singl_char(clean):
    singl = re.sub(r"\b[a-zA-Z]\b", " ",clean)
    return singl
df_data['clean'] = df_data['clean'].apply(remove_singl_char)
```

Gambar 4.7 *Single Char Removal*

5. *Remove Repeated Letters*

Pada gambar 4.8 terdapat fungsi *remove_repeated_letters* dengan tujuan untuk menghapus atau mengurangi pengulangan huruf yang berlebihan dalam kata-kata misalnya "baguuuuus", "buruuuuk", "yeeeessss".

```python
def remove_repeated_letters (clean):
#mencocokban pola kata-kata dengan huruf yang diulang minimal 3 kali
    pattern = r'\b\w*(\w)\1{2,}\w*\b'
    processed_text = re.sub(pattern, '', clean)
    return processed_text
df_data['filtering'] = df_data['clean'].apply(remove_repeated_letters)
```

Gambar 4.8 *Remove Repeated Letters*

Gambar 4.9 di bawah merupakan hasil dari proses filtering yang mana data telah dibersihkan dengan menghapus karakter-karakter yang tidak relevan. Berikut adalah hasil dari perintah dalam proses filtering data.

```
0       aplikasi sering keluar sendiri  baru mau sudah...
1       tolong di perbaiki lagi aplikasinya  hapir set...
2       ribet  aktivasi harus ke bank padahal bni mobi...
3       transfer rempong  kalo lagi tf pembayarannyo s...
4         banking sering eror minta aktivasi ulang dan...
                              ...
2980                                          cukup keren
2981                                       mudah aplikasinya
2982    tidak bisam daftar aktivasi padahal sudah masu...
2983                                        susah download
2984                                                   sip
Name: filtering, Length: 2985, dtype: object
```

Gambar 4.9 Hasil Filtering

4.3.3 Tokenizing

Tokenizing merupakan pemecahan atau pemisahan teks menjadi bagian-bagian yang lebih kecil yang disebut "token". Token ini biasanya berupa kata-kata atau kalimat.

Gambar 4.10 terdapat fungsi *tokenizing* yang mana untuk melakukan pemisahan karakter pada kalimat sehingga bisa disimpan menjadi setiap kata dalam dokumen atau *term*.

```python
from nltk.tokenize import word_tokenize

def tokenizing(filtering):
    tokens = word_tokenize(filtering)
    return tokens

df_data['tokenize'] = df_data['filtering'].apply(tokenizing)
print(df_data['tokenize'])
```

Gambar 4.10 *Tokenizing*

Gambar 4.11 di bawah merupakan hasil dari *tokenizing* yang mana karakter pada kalimat telah dipisahkan.

```
0       [aplikasi, sering, keluar, sendiri, baru, mau,...
1       [tolong, di, perbaiki, lagi, aplikasinya, hapi...
2       [ribet, aktivasi, harus, ke, bank, padahal, bn...
3       [transfer, rempong, kalo, lagi, tf, pembayaran...
4       [banking, sering, eror, minta, aktivasi, ulang...
                              ...
2980                                      [cukup, keren]
2981                                  [mudah, aplikasinya]
2982    [tidak, bisam, daftar, aktivasi, padahal, suda...
2983                                  [susah, download]
2984                                               [sip]
Name: tokenize, Length: 2985, dtype: object
```

Gambar 4.11 hasil *Tokenizing*

4.3.4 Stopword Removal

Stopword Removal yaitu tahap menghapus kata-kata umum yang tidak memiliki makna penting dalam analisis teks. Contohnya yaitu (dan, yang, di, ke, dari, adalah) dan lain sebagainya. Berikut adalah kode program dari *stopword removal.*

Kode pada gambar 4.12 di atas bertujuan untuk menghapus kata-kata umum yang tidak memiliki makna penting (*stopwords*) dari teks yang ada dalam *data frame* df_data.

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory

factory = StopWordRemoverFactory()
stopword = factory.create_stop_word_remover()

def stopword_removal (tokens):
  if isinstance(tokens, str):
    stopwords = stopword.remove(tokens)

  else:
    stopwords = [stopword.remove(tokenize) for tokenize in tokens]
  return stopwords

df_data['stopword'] = df_data['tokenize'].apply(stopword_removal)

print(df_data['stopword'])
```

Gambar 4.12 Stopword Removal

Gambar 4.13 merupakan output dari stopword removal yang mana teks menjadi lebih bersih dan fokus pada kata-kata yang lebih bermakna dan informatif.

```
0       [aplikasi, sering, keluar, sendiri, baru, mau,...
1       [, , perbaiki, , aplikasinya, hapir, , login, ...
2       [ribet, aktivasi, , , bank, padahal, bni, mobi...
3       [transfer, rempong, kalo, , tf, pembayarannyo,...
4       [banking, sering, eror, minta, aktivasi, ulang...
                              ...
2980                                      [cukup, keren]
2981                                   [mudah, aplikasinya]
2982    [, bisam, daftar, aktivasi, padahal, , masukan...
2983                                    [susah, download]
2984                                                [sip]
Name: stopword, Length: 2985, dtype: object
```

Gambar 4.13 Hasil Stopword Removal

4.3.5 Stemming

Stemming merupakan tahap untuk menghilangkan imbuhan kata dan untuk mengubah kata-kata ke bentuk dasarnya.

Kode pada gambar 4.14 tersebut menggunakan *library* Sastrawi untuk stemming teks yang telah dihapus stopwordsnya. Fungsi *stemming* dibuat untuk mengubah setiap kata menjadi bentuk dasarnya. Kemudian fungsi ini diterapkan pada kolom *stopword* di DataFrame df_data, sehingga setiap token diubah ke bentuk dasar dan hasilnya disimpan dalam kolom baru stemmer.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def stemming(stopwords):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    return stemmer.stem(stopwords)
df_data['stemmer'] = df_data['stopword'].apply(lambda x: [stemming(token) for token in x])
print(df_data['stemmer'])
```

Gambar 4.14 Stemming

Gambar 4.15 merupakan hasil output dari stemming yang mana setiap kata telah menjadi kata dasar.

```
0       [aplikasi, sering, keluar, sendiri, baru, mau,...
1       [, , baik, , aplikasi, hapir, , login, sulit, ...
2       [ribet, aktivasi, , , bank, padahal, bni, mobi...
3       [transfer, rempong, kalo, , tf, pembayarannyo,...
4       [banking, sering, eror, minta, aktivasi, ulang...
                              ...
2980                                      [cukup, keren]
2981                                   [mudah, aplikasi]
2982    [, bisam, daftar, aktivasi, padahal, , masuk, ...
2983                                    [susah, download]
2984                                                [sip]
Name: stemmer, Length: 2985, dtype: object
```

Gambar 4.15 Hasil Stemming

4.3.6 Normalisasi

Normalisasi adalah tahap untuk mengubah data teks yang mengalami kesalahan penulisan atau penggunaan kata yang tidak baku.

Pada gamabar 4.16 bertujuan untuk melakukan normalisasi teks dengan Kode mengimpor kelas WordNetLemmatizer dari *library* NLTK untuk melakukan lemmatisasi, yaitu mengonversi kata-kata dalam teks menjadi bentuk dasarnya. Selanjutnya, dictionary norm_dict digunakan untuk menggantikan kata-kata tertentu dalam teks dengan kata yang telah ditentukan sebelumnya. Dalam fungsi normalizing, setiap token dalam teks dilemmatisasi terlebih dahulu, kemudian setiap token diperiksa dalam dictionary.

```
def normalizing(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    normalized_tokens = [norm_dict.get(token, token) for token in lemmatized_tokens]
    normalized_text = " ".join(normalized_tokens)
    return normalized_text

df_data['normalizing'] = df_data['stemmer'].apply(normalizing)

print(df_data['normalizing'])
```

Gambar 4.16 Normalisasi

Gambar 4.17 menunjukan bahwa data ulasan teks yang telah diubah menjadi bentuk yang lebih standar dan umum atau baku

```
0       aplikasi sering keluar sendiri baru mau   inga...
1         baik  aplikasi hapir  login sulit minta akti...
2       ribet aktivasi   bank padahal bni mobile brimo...
3       transfer rempong kalo  transfer pembayarannyo ...
4       banking sering eror minta aktivasi ulang  pros...
                              ...
2980                                          cukup keren
2981                                       mudah aplikasi
2982     bisa daftar aktivasi padahal  masuk nomor yan...
2983                                       susah download
2984                                                  sip
Name: normalizing, Length: 2985, dtype: object
```

Gambar 4.17 Hasil Normalisasi

4.4 EKSTRAKSI FITUR

4.4.1 TF-IDF

Kode yang disajikan pada gambar 4.18 menggunakan TfidfVectorizer dari library sklearn untuk mengubah teks yang sudah dinormalisasi menjadi representasi berbasis TF-IDF.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
stopwords = stopwords.words('english')

df_data = df_data.dropna()
df_data ['normalizing'] = df_data['normalizing'].astype(str)
tfidf = TfidfVectorizer(
    min_df=5,
    max_df=0.95,
    max_features=10000,
    stop_words=stopwords)

text = tfidf.fit(df_data.normalizing)
text = tfidf.transform(df_data.normalizing)

print(text)
```

Gambar 4.18 Kode TF IDF

Gambar 4.19 merupakan output dari TF-IDF menunjukkan indeks kata dan nilai pentingnya dalam dokumen. Misalnya, (0, 625) berarti kata pada indeks 625 dalam dokumen pertama memiliki nilai penting 0.2216. Nilai ini menunjukkan seberapa penting kata tersebut dalam dokumen dibandingkan dengan seluruh kumpulan dokumen. Semakin tinggi nilai TF-IDF, semakin penting kata itu dalam dokumen.

```
(0, 625)      0.22158232667943062
(0, 606)      0.19099012426733497
(0, 538)      0.1703299314224059
(0, 537)      0.471650009138615
(0, 392)      0.20107280072047187
(0, 390)      0.18242744165110117
(0, 374)      0.14453248447823358
(0, 345)      0.23055721941303517
(0, 280)      0.22495936278715387
(0, 279)      0.2768056321214634
(0, 269)      0.24309357936575984
(0, 232)      0.2827777543718692
(0, 60)       0.1897758103690501
(0, 55)       0.2896720635437971
(0, 50)       0.13669397955478485
(0, 33)       0.10390172427323349
```

Gambar 4.19 Output TF-IDF

4.5 K-MEANS

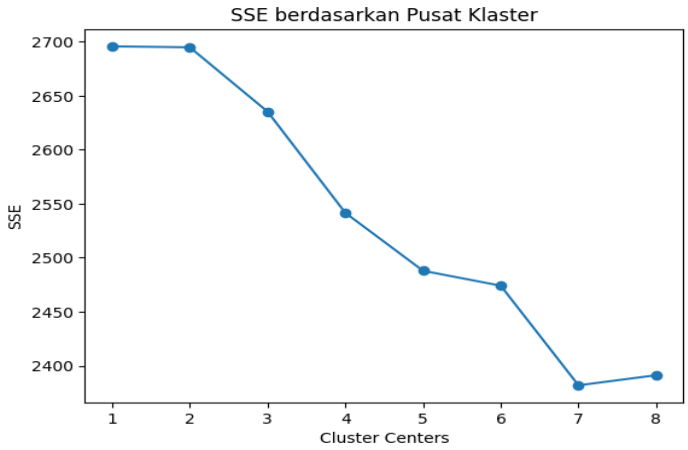
Dengan menggunakan metode elbow, dapat menentukan jumlah klaster yang optimal secara visual berdasarkan grafik nilai inertia (SSE). Jumlah *cluster* yang ditetapkan yaitu titik yang membentuk siku pada grafik.

Kode program pada gambar 4.20 di bawah merupakan untuk menentukan jumlah *cluster* yang optimal.

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.cluster import MiniBatchKMeans

def find_optimal_clusters(data, max_k):
    iters = range(1, max_k+1, 1)

    sse = []
    for k in iters:
        sse.append(MiniBatchKMeans(n_clusters=k,
                                   init_size=1024,
                                   batch_size=256,
                                   random_state=11).fit(data).inertia_)
        print('Fit {} klaster'.format(k))

    f, ax = plt.subplots(1, 1)
    ax.plot(iters, sse, marker='o')
    ax.set_xlabel('Cluster Centers')
    ax.set_xticks(iters)
    ax.set_xticklabels(iters)
    #SSE mengukur seberapa baik titik data dalam klaster tertentu dikelompokkan di sekitar pusat klaster (centroid).
    ax.set_ylabel('SSE')
    #Sum of Squared Errors (SSE)
    ax.set_title('SSE berdasarkan Pusat Klaster')
    plt.show()

find_optimal_clusters(text, 8)
```

Gambar 4.20 Kode Program Elbow Method

Gambar 4.21 di bawah jumlah *cluster* ada di titik grafik nomor 4 karena menunjukan bentuk sebuah siku. Meskipun ada beberapa titik yang menunjukkan bentuk siku namun titik ke-4 lebih jelas membentuk sudut siku.



Gambar 4.21 Grafik Elbow

kode yang disajikan pada gambar 4.22 di bawah ini yaitu menggunakan algoritma MiniBatchKMeans untuk mengelompokkan data teks ke dalam 4 klaster.

```
from sklearn.cluster import MiniBatchKMeans

#Menghasilkan kluster menggunakan MiniBatchKMeans
clusters = MiniBatchKMeans(n_clusters=4, init_size=1025, batch_size=2049,
                           random_state=11).fit_predict(text)

#Membuat DataFrame dari data teks dan kluster
text_clusters ={'text': text, 'cluster': clusters}
df_data = pd.DataFrame(text_clusters, columns=['text', 'cluster'])
text = tfidf.inverse_transform(text) # mengubah kembali ke bentuk string
text = [' '.join(t) for t in text] # menggabungkan token menjadi kalimat
text_clusters = {'text': text, 'cluster': clusters}
df_data = pd.DataFrame (text_clusters, columns=['text', 'cluster'])

print(df_data)

df_data['cluster'].value_counts()
```

Gambar 4.22 Clustering

Pada gambar 4.23 merupakan output dari pengelompokan data yang terdapat 4 cluster yaitu *cluster* 0, *cluster* 1, *cluster* 2, *cluster* 3.

```
                                                   text  cluster
0     terus sering sendiri mohon mobile mau lot kelu...        3
1     yang ujung transfer sulit padahal nomer minta ...        3
2     udah sama ribet padahal mobile mandiri bni ban...        3
3     virtual transfer transaksi sering sedot saldo ...        3
4     ulang sulit sering sangat prose nasabah minta ...        3
...                                                 ...      ...
2863                                      keren cukup        3
2864                                    mudah aplikasi       0
2865  yang temu rekening pake padahal nomor masuk in...        3
2866                                    susah download       3
2867                                                         3

[2868 rows x 2 columns]

3    1704
1     601
0     428
2     135
Name: cluster, dtype: int64
```

Gambar 4.23 Hasil Clustering

Wordcloud pada gambar 4.24 merupakan untuk memahami tema yang dibahas dalam setiap cluster sehingga data tersebut bisa dianalisis apa yang sedang di perbincangkan.

```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# Tentukan jumlah klaster
num_clusters = df_data['cluster'].nunique()
# Tentukan ukuran grid untuk 4 klaster
rows, cols = 2, 2
# Buat figure dan axes untuk subplots
fig, axes = plt.subplots(rows, cols, figsize=(15, 10))
# Pastikan axes adalah array 2D
axes = axes.flatten()
# Visualisasi dengan WordCloud untuk setiap klaster
for k in range(num_clusters):
    # Memfilter data untuk klaster tertentu
    cluster_data = df_data[df_data['cluster'] == k]
    # Menggabungkan teks dari semua dokumen dalam kluster menjadi satu string
    text = cluster_data['text'].str.cat(sep=' ')
    # Mengonversi teks menjadi lowercase
    text = text.lower()
    # Membuat WordCloud
    wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(text)
    # Menampilkan WordCloud untuk setiap klaster
    axes[k].imshow(wordcloud, interpolation="bilinear")
    axes[k].set_title(f'Cluster: {k}')
    axes[k].axis("off")
# Hapus axes yang tidak terpakai jika jumlah klaster tidak tepat mengisi grid
for i in range(num_clusters, len(axes)):
    fig.delaxes(axes[i])
# Tampilkan plot
plt.tight_layout()
plt.show()
```

Gambar 4.24 Wordcloud

WordCloud pada gambar 4.25 menggambarkan kata-kata yang paling sering muncul dalam teks atau dokumen yang termasuk dalam klaster tersebut. Kata-kata yang muncul dengan frekuensi yang lebih tinggi akan ditampilkan dengan ukuran yang lebih besar dalam WordCloud, sementara kata-kata yang jarang muncul akan lebih kecil atau bahkan tidak muncul sama sekali.

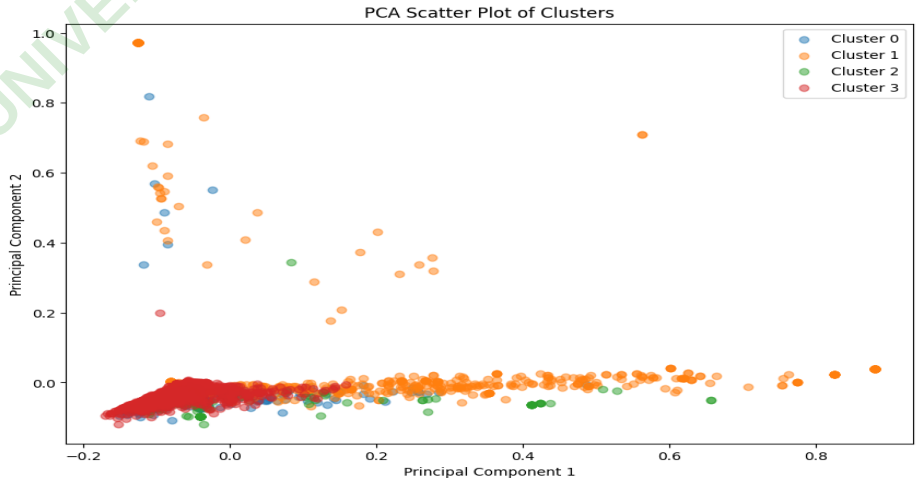
Gambar 4.25 Hasil Wordcloud

Pada gambar 4.26 merupakan kode untuk melakukan analisis dan visualisasi data teks. Kode ini membuat scatter plot untuk menggambarkan distribusi data dalam dua dimensi, dengan setiap titik mewakili dokumen yang dikelompokkan berdasarkan klaster yang ditentukan.

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Tentukan jumlah klaster
num_clusters = df_data['cluster'].nunique()

# Mengonversi teks ke fitur TF-IDF
tfidf = TfidfVectorizer(max_features=10000, stop_words='english')
X_tfidf = tfidf.fit_transform(df_data['text'])

# Menggunakan PCA untuk mereduksi dimensi
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_tfidf.toarray())

# Membuat scatter plot
plt.figure(figsize=(10, 7))

for k in range(num_clusters):
    cluster_data = X_pca[df_data['cluster'] == k]
    plt.scatter(cluster_data[:, 0], cluster_data[:, 1], label=f'Cluster {k}', alpha=0.5)

plt.title('PCA Scatter Plot of Clusters')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```

Gambar 4.26 Scatter plot

Gambar 4.27 ini menunjukkan hasil PCA dari kluster data teks. Titik-titik berwarna mewakili dokumen dalam kluster yang berbeda: biru (kluster 0), oranye (kluster 1), hijau (kluster 2), dan merah muda (kluster 3). Kluster 1 tersebar luas, sementara kluster 0 dan 2 lebih terkonsentrasi. Sebagian besar titik berkumpul di sekitar pusat plot.



Gambar 4.27 Hasil Scatter plot

Tabel 3 Analisis Pembahasan

Cluster	Kata Yang Sering Muncul	Hasil Analisis
Cluster 0	aplikasi, aktivasi, transaksi, buka, terus, tidak, blokir, eror, ribet	Pengguna sering menghadapi kesulitan dalam mengakses aplikasi, melakukan aktivasi akun, dan proses transaksi. Keluhan tentang aplikasi yang sering mengalami error, ribet, atau tidak berfungsi dengan baik juga umum.
Cluster 1	mantap, sangat, bantu, transaksi, bagus, mudah, good, manfaat, aman	Pengguna merasa puas dengan kemudahan penggunaan aplikasi dalam melakukan transaksi. Mereka menyoroti keamanan yang baik dan manfaat yang diperoleh dari penggunaan aplikasi.
Cluster 2	baik, aplikasi, sangat, layanan, mohon, tingkat, susah, aman, bantu	Pengguna menganggap layanan aplikasi sudah baik, tetapi ada harapan untuk peningkatan lebih lanjut. Permintaan untuk bantuan lebih intens dalam mengatasi masalah dan memastikan keamanan aplikasi.
Cluster 3	transfer, transaksi, Aktivasi, sering, gagal, masuk, error, ribet, update	Mencakup masalah operasional dan pengalaman pengguna. Pengguna menghadapi masalah teknis seperti kegagalan dalam transfer dan transaksi, serta proses aktivasi yang sering tidak berhasil. Masalah aplikasi yang sering error dan memerlukan update teratur untuk meningkatkan kinerja.

Pada tabel 3 di atas merupakan hasil frekuensi kata yang sering muncul dari setiap cluster.

4.6 NAÏVE BAYES

Pada tahap ini menjelaskan untuk memperoleh hasil klasifikasi akan di lalukan pelabelan pada setiap cluster, label ini berupa kategori seperti "Positif", "Netral", atau "Negatif" berdasarkan skor sentimen yang diberikan. Pembagian data (split data) yang mana proses membagi dataset menjadi dua subset: data latih

(training data) dan data uji (testing data). Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk mengevaluasi kinerja model. Yang terakhir yaitu Klasifikasi dengan *Naive Bayes* model *Naive Bayes* dilatih menggunakan data latih, kemudian digunakan untuk memprediksi label data uji. Hasil prediksi dievaluasi menggunakan metrik evaluasi yang relevan.

Kode program di bawah merupakan kode dari pelabelan untuk cluster 0 Kode ini secara keseluruhan berfungsi untuk mengelompokkan data dalam cluster 0 berdasarkan skor/rating ulasan dan memberikan label yang sesuai (‘Positif’, ‘Netral’, atau ‘Negatif’). Langkah-langkah yang dilakukan meliputi membaca data dari file CSV, mendefinisikan fungsi untuk memberi label berdasarkan skor ulasan, menerapkan fungsi pelabelan pada data dalam cluster 0, memisahkan data cluster 0 ke dalam DataFrame baru, menghitung jumlah data dalam cluster 0, menghitung jumlah masing-masing label (Positif, Netral, Negatif), menampilkan beberapa contoh ulasan dari cluster 0 beserta labelnya, dan menyimpan hasil data cluster 0 yang telah diberi label ke dalam file CSV baru.

1. Cluster 0

```
import pandas as pd
# Membaca data dari file CSV atau dari data yang sudah ada
df_data = pd.read_csv('cluster.csv', delimiter=';')  # Ubah
'cluster.csv' dengan nama file CSV Anda
# Fungsi untuk memberi label berdasarkan skor
def label_review(score):
    if score == 5:
        return 'Positif'
    elif score >= 3:
        return 'Netral'
    else:
        return 'Negatif'
# Memberi label pada cluster 0
df_data.loc[df_data['cluster'] == 0, 'label'] =
df_data[df_data['cluster'] == 0]['score'].apply(label_review)
# Memilih hanya data dari cluster 0 untuk disimpan
cluster_0_data = df_data[df_data['cluster'] == 0]

# Menghitung jumlah data dalam cluster 0
num_data_cluster_0 = len(cluster_0_data)
```

```
print(f"Jumlah data dalam cluster 0: {num_data_cluster_0}")
# Menghitung jumlah masing-masing label
label_counts = cluster_0_data['label'].value_counts()
num_positif = label_counts.get('Positif', 0)
num_netral = label_counts.get('Netral', 0)
num_negatif = label_counts.get('Negatif', 0)
print(f"Jumlah Positif: {num_positif}")
print(f"Jumlah Netral: {num_netral}")
print(f"Jumlah Negatif: {num_negatif}")
print(f"Cluster 0 with assigned labels:")
print(cluster_0_data[['text', 'score', 'label']].head())

# Menyimpan hasil dari cluster 0 ke dalam file CSV
cluster_0_data.to_csv('label_cluster0.csv', index=False)
```

Hasil gambar 4.28 menunjukkan bahwa dari 428 ulasan dalam cluster 0, sebanyak 87 ulasan dinilai ‘Positif’, 24 ulasan dinilai ‘Netral’, dan 317 ulasan dinilai ‘Negatif’. Label-label ini diberikan berdasarkan skor ulasan yang telah ditentukan sebelumnya untuk setiap ulasan dalam cluster tersebut.

```
Jumlah data dalam cluster 0: 428
Jumlah Positif: 87
Jumlah Netral: 24
Jumlah Negatif: 317
Cluster 0 with assigned labels:
                                                text  score    label
6                              transaksi mudah aplikasi    5.0  Positif
13  sudah mau masuk mana keluar kata buka bagai ap...    2.0  Negatif
15       update terus paket masak data bulan aplikasi    3.0   Netral
16                    sering sendiri blokir aplikasi    1.0  Negatif
17  sistem masalah lumayan kelas ganggu baik bagus...    4.0   Netral
```

Gambar 4.28 Hasil Pelabelan Cluster 0

Program pada gambar 4.29 ini untuk membagi data yang sudah diberi label dari file CSV ‘label_cluster0.csv’ menjadi dua bagian menggunakan fungsi train_test_split dari library scikit-learn. Data dibagi dengan proporsi 80% untuk data latih (training set) dan 20% untuk data uji (testing set). Pembagian ini memungkinkan untuk melatih model pada data latih dan menguji performanya pada data uji.

```python
#membagi data menjadi data training dan testing dengan test_size = 0.20 dan random state nya 0
from sklearn.model_selection import train_test_split

# Membaca data yang sudah diberi label dari file CSV
df_data = pd.read_csv('label_cluster0.csv')

X = df_data['text']
y = df_data['label']

# Split the data: 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Display the split data sizes
print(f'Training set size: {len(X_train)}')
print(f'Test set size: {len(X_test)}')

# Optional: Create a DataFrame for training and test sets for easy access
train_data = pd.DataFrame({'text': X_train, 'label': y_train})
test_data = pd.DataFrame({'text': X_test, 'label': y_test})

# Display a few samples from training and test sets
print("Training set samples:")
print(train_data.head())
print("\nTest set samples:")
print(test_data.head())
```

Gambar 4.29 Split Data Cluster 0

Pada gambar 4.30 data Cluster 0 telah dibagi menjadi dua bagian, data latih dengan 342 sampel dan data uji dengan 86 sampel. Data latih digunakan untuk melatih model klasifikasi, sedangkan data uji digunakan untuk menguji seberapa baik model tersebut dapat memprediksi label pada ulasan yang belum pernah dilihat sebelumnya.

```
Training set size: 342
Test set size: 86
Training set samples:
                                                  text    label
64   untk otp mudah lebih klu hapus depan biar baya...  Positif
235         uang transaksi payah masuk keluar aplikasi  Negatif
158          tolol tidak niat kalau hapus aplikasi aja  Negatif
42   ulang registrasi mulu mintak mau kalo kali jgn...  Negatif
191       update tunggu buka bug baru banking aplikasi  Negatif

Test set samples:
                                                  text    label
261             transfer lengkap fitur bayar aplikasi  Negatif
85                user sangat keren friendly aplikasi  Positif
422                    versi upgrade baru aplikasi ak  Negatif
95                         mudah mantab guna aplikasi  Positif
167  transaksi tambah susah sinyal sering salah pad...  Negatif
```

Gambar 4.30 Hasil Split Data Cluster 0

Kode gambar 4.31 tersebut digunakan untuk membuat pipeline yang menggabungkan TF-IDF Vectorizer dengan model klasifikasi Naive Bayes

Multinomial untuk melakukan klasifikasi pada data teks. Setelah melatih model menggunakan data latih (X_train, y_train), model dievaluasi dengan menggunakan data uji (X_test, y_test) untuk menghasilkan laporan evaluasi yang mencakup precision, recall, f1-score, dan akurasi secara keseluruhan.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score

# Buat pipeline dengan TF-IDF dan Multinomial Naive Bayes
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Prediksi menggunakan data uji
y_pred = model.predict(X_test)

# Evaluasi model
print("Hasil Evaluasi Model cluster 0:")
print(classification_report(y_test, y_pred))
print(f"Akurasi Model cluster 0: {accuracy_score(y_test, y_pred)}")
```

Gambar 4.31 Klasifikasi

Hasil evaluasi model untuk cluster 0 pada gambar 4.32 menunjukkan bahwa model memiliki precision yang baik untuk kelas 'Negatif' (0.83), recall yang tinggi (0.98), dan akurasi keseluruhan sebesar 83.7%. Namun, kelas 'Netral' memiliki performa yang rendah dengan precision dan recall keduanya adalah 0.00.

```
Hasil Evaluasi Model cluster 0:
              precision    recall  f1-score   support

     Negatif       0.83      0.98      0.90        64
      Netral       0.00      0.00      0.00         5
     Positif       0.90      0.53      0.67        17

    accuracy                           0.84        86
   macro avg       0.58      0.50      0.52        86
weighted avg       0.79      0.84      0.80        86

Akurasi Model cluster 0: 0.8372093023255814
```

Gambar 4.32 Hasil Klasifikasi Cluster 0

2. Cluster 1

Kode program di bawah ini digunakan untuk memberi label pada ulasan yang terdapat dalam cluster 1 berdasarkan skor/rating ulasan yang sudah ada sebelumnya. Setelah diberi label, data yang termasuk dalam cluster 1 disimpan dalam file CSV dengan nama ‘label_cluster1.csv’. Hasilnya menunjukkan bahwa dari jumlah total ulasan dalam cluster 1, terdapat sejumlah ulasan yang dinilai sebagai ‘Positif’, ‘Netral’, dan ‘Negatif’, dengan jumlah masing-masing label yang tertampil dalam keluaran.

```
import pandas as pd
def label_review(score):
    if score == 5:
        return 'Positif'
    elif score >= 3:
        return 'Netral'
    else:
        return 'Negatif'
# Membaca data dari file CSV atau dari data yang sudah ada
df_data = pd.read_csv('cluster.csv', delimiter=';')  # Ubah
'Clusternan.csv' dengan nama file CSV Anda
# Memberi label pada cluster 1
df_data.loc[df_data['cluster'] == 1, 'label'] =
df_data[df_data['cluster'] == 1]['score'].apply(label_review)
# Memilih hanya data dari cluster 1 untuk disimpan
cluster_1_data = df_data[df_data['cluster'] == 1]
# Menghitung jumlah data dalam cluster 1
num_data_cluster_1 = len(cluster_1_data)
print(f"Jumlah data dalam cluster 1: {num_data_cluster_1}")
# Menghitung jumlah masing-masing label
label_counts = cluster_1_data['label'].value_counts()
num_positif = label_counts.get('Positif', 0)
num_netral = label_counts.get('Netral', 0)
num_negatif = label_counts.get('Negatif', 0)

print(f"Jumlah Positif: {num_positif}")
print(f"Jumlah Netral: {num_netral}")
print(f"Jumlah Negatif: {num_negatif}")
# Menampilkan beberapa ulasan dari cluster 1 dengan label
print(f"Cluster 1 with assigned labels:")
print(cluster_1_data[['text', 'score', 'label']].head())
# Menyimpan hasil dari cluster 1 ke dalam file CSV
cluster_1_data.to_csv('label_cluster1.csv', index=False)
```

Dari hasil gambar 4..33 pada pelabelan cluster 1, terdapat total 600 ulasan yang telah dinilai berdasarkan skornya. Dari jumlah tersebut, sebanyak 537 ulasan dikategorikan sebagai 'Positif', sedangkan terdapat 42 ulasan yang dinilai ‘Netral’, dan hanya 21 ulasan yang masuk ke dalam kategori ‘Negatif’. Proses pemberian label dilakukan berdasarkan skor ulasan awal, di mana ulasan dengan skor 5 diberi label ‘Positif’. Hal ini menunjukkan bahwa mayoritas ulasan dalam cluster 1 memberikan penilaian positif terhadap topik yang dibahas.

```
Jumlah data dalam cluster 1: 600
Jumlah Positif: 537
Jumlah Netral: 42
Jumlah Negatif: 21
Cluster 1 with assigned labels:
                                text  score    label
7                              bagus    5.0  Positif
34                             bagus    5.0  Positif
42                            mantap    5.0  Positif
43                            mantap    5.0  Positif
46  yang sekali sangat bantu aplikasi    5.0  Positif
```

Gambar 4.33 Hasil Pelabelan Cluster 1

Gambar 4.34 merupakan tahap split data pada cluster 1 dilakukan untuk membagi dataset menjadi dua bagian: data training dan data testing. Data ini dibagi dengan proporsi 80% untuk training set dan 20% untuk test set dari total data yang ada. Proses ini menggunakan fungsi train_test_split dari library sklearn.model_selection, yang membagi data X (fitur teks) dan y (label yang sudah ditentukan) berdasarkan parameter test_size=0.2 untuk menentukan proporsi test set sebesar 20%. Penggunaan random_state=42 memastikan bahwa pembagian data ini bersifat deterministik dan dapat direproduksi, sementara stratify=y digunakan untuk memastikan bahwa distribusi kelas pada data training dan test set tetap seimbang sesuai dengan label y. Setelah pembagian, ukuran dari training set dan test set ditampilkan untuk memverifikasi proses pembagian data.

```python
# Membaca data yang sudah diberi label dari file CSV
df_data = pd.read_csv('label_cluster1.csv')

# Memisahkan data menjadi fitur (X) dan label (y)
X = df_data['text']  # Variabel teks sebagai fitur
y = df_data['label']  # Label yang sudah ditentukan

# Split the data: 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Display the split data sizes
print(f'Training set size: {len(X_train)}')
print(f'Test set size: {len(X_test)}')

# Optional: Create a DataFrame for training and test sets for easy access
train_data = pd.DataFrame({'text': X_train, 'label': y_train})
test_data = pd.DataFrame({'text': X_test, 'label': y_test})

# Display a few samples from training and test sets
print("Training set samples:")
print(train_data.head())
print("\nTest set samples:")
print(test_data.head())
```

Gambar 4.34 Split Data Cluster 1

Gambar 4.35 dari proses split data cluster 1 menunjukkan bahwa dataset telah berhasil dibagi menjadi dua bagian: data training dengan ukuran 480 sampel dan data testing dengan ukuran 120 sampel. Data ini telah dibagi dengan proporsi 80% untuk training set dan 20% untuk test set dari total 600 sampel yang tersedia. Setiap bagian dari data training dan test set juga telah ditampilkan beberapa sampelnya untuk memvisualisasikan bagaimana struktur dataset setelah pembagian.

```
Training set size: 480
Test set size: 120
Training set samples:
                text    label
6       mudah mantap  Positif
532  transaksi mudah  Positif
403  utama good aman  Positif
377     sangat bantu  Positif
580             good  Positif

Test set samples:
                                text    label
183         sangat bantu aplikasi  Positif
55                         mantap  Positif
161  transaksi sangat laku bantu  Positif
35          sangat praktis bantu  Positif
1                          bagus  Positif
```

Gambar 4.35 Hasil Split Data Cluster 1

Kode program gambar 4.36 tersebut menggunakan pipeline yang terdiri dari TF-IDF Vectorizer dan model Multinomial Naive Bayes untuk mengolah dan

mengklasifikasikan teks. Model dilatih dengan data training (X_train dan y_train), kemudian digunakan untuk memprediksi label pada data uji (X_test). Evaluasi dilakukan dengan mencetak laporan klasifikasi dan menghitung akurasi model terhadap data uji.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score

# Buat pipeline dengan TF-IDF dan Multinomial Naive Bayes
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Prediksi menggunakan data uji
y_pred = model.predict(X_test)

# Evaluasi model
print("Hasil Evaluasi Model cluster 1:")
print(classification_report(y_test, y_pred))
print(f"Akurasi Model cluster 1: {accuracy_score(y_test, y_pred)}")
```

Gambar 4.36 Klasifikasi

Hasil klasifikasi untuk cluster 1 pada gambar 4.37 menunjukkan bahwa model memiliki tingkat akurasi sebesar 89.17%. Dari hasil klasifikasi, diketahui bahwa model memiliki tingkat presisi yang rendah untuk kategori Negatif dan Netral, dengan nilai 0.00. Namun, model berhasil dengan baik dalam mengidentifikasi ulasan positif, dengan nilai presisi mencapai 0.90 dan recall sebesar 1.00. Hal ini menunjukkan bahwa model cenderung lebih baik dalam mengklasifikasikan ulasan sebagai Positif dibandingkan dengan kategori lainnya.

```
Hasil Evaluasi Model cluster 1:
              precision    recall  f1-score   support

     Negatif       0.00      0.00      0.00         4
      Netral       0.00      0.00      0.00         9
     Positif       0.90      1.00      0.95       107

    accuracy                           0.89       120
   macro avg       0.30      0.33      0.32       120
weighted avg       0.80      0.89      0.84       120

Akurasi Model cluster 1: 0.8916666666666667
```

Gambar 4.37 Hasil Klasifikasi

3. Cluster 2

Kode di bawah merupakan Kode untuk mengolah data dari file CSV bernama ‘Clusternan.csv’ dan memberikan label pada ulasan dalam cluster 2 berdasarkan skornya. Ulasan dengan skor 5 diberi label ‘Positif’, skor antara 3 dan 4 diberi label ‘Netral’, dan skor di bawah 3 diberi label ‘Negatif’. Setelah proses pelabelan, data dalam cluster 2 disimpan dalam file CSV baru bernama ‘label_cluster2.csv’.

```
import pandas as pd
df_data = pd.read_csv('Clusternan.csv', delimiter=';')
# Fungsi untuk memberi label berdasarkan skor
def label_review(score):
    if score == 5:
        return 'Positif'
    elif score >= 3:
        return 'Netral'
    else:
        return 'Negatif'

# Memberi label pada cluster 2
df_data.loc[df_data['cluster'] == 2, 'label'] =
df_data[df_data['cluster'] == 2]['score'].apply(label_review)
# Memilih hanya data dari cluster 2 untuk disimpan
cluster_2_data = df_data[df_data['cluster'] == 2]
# Menghitung jumlah data dalam cluster 2
num_data_cluster_2 = len(cluster_2_data)
print(f"Jumlah data dalam cluster 2: {num_data_cluster_2}")

# Menghitung jumlah masing-masing label
label_counts = cluster_2_data['label'].value_counts()
num_positif = label_counts.get('Positif', 0)
num_netral = label_counts.get('Netral', 0)
num_negatif = label_counts.get('Negatif', 0)

print(f"Jumlah Positif: {num_positif}")
print(f"Jumlah Netral: {num_netral}")
print(f"Jumlah Negatif: {num_negatif}")
# Menampilkan beberapa ulasan dari cluster 2 dengan label
print(f"Cluster 2 with assigned labels:")
print(cluster_2_data[['text', 'score', 'label']].head())
# Menyimpan hasil dari cluster 2 ke dalam file CSV
cluster_2_data.to_csv('label_cluster2.csv', index=False)
```

Hasil pelabelan gambar 4.38 dari cluster 2 menunjukkan bahwa dari total 135 ulasan, 88 ulasan diberi label ‘Positif’, 16 ulasan diberi label ‘Netral’, dan 31 ulasan diberi label Negatif’. Beberapa contoh ulasan dengan label yang diberikan mencakup ulasan positif seperti “baik” dan “mobile banking baik”, serta ulasan netral seperti “sangat manfaat bantu baik”.

```
Jumlah data dalam cluster 2: 135
Jumlah Positif: 88
Jumlah Netral: 16
Jumlah Negatif: 31
Cluster 2 with assigned labels:
                              text  score    label
20                            baik    5.0  Positif
32       sangat manfaat bantu baik    4.0   Netral
38             mobile banking baik    5.0  Positif
45  update makin kembang hari baik    5.0  Positif
49             kerja baik aplikasi    5.0  Positif
```

Gambar 4.38 Hasil Pelabelan

Gambar 4.39 kode program ini memproses data ulasan dari cluster 2 yang telah diberi label, membaginya menjadi data latih (80%) dan data uji (20%) dengan stratifikasi berdasarkan label untuk memastikan proporsi yang sama dalam kedua set tersebut.

```python
from sklearn.model_selection import train_test_split

# Membaca data yang sudah diberi label dari file CSV
df_data = pd.read_csv('label_cluster2.csv')

X = df_data['text']
y = df_data['label']

# Membagi data menjadi data latih (80%) dan data uji (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Menampilkan ukuran set latih dan uji
print(f'Training set size: {len(X_train)}')
print(f'Test set size: {len(X_test)}')

# Menampilkan beberapa sampel dari set latih dan uji
train_data = pd.DataFrame({'text': X_train, 'label': y_train})
test_data = pd.DataFrame({'text': X_test, 'label': y_test})

print("Training set samples:")
print(train_data.head())
print("\nTest set samples:")
print(test_data.head())
```

Gambar 4.39 Split Data Cluster 2

Dari hasil gambar 4.40 cluster 2 menunjukkan bahwa dari 135 ulasan, 108 ulasan dialokasikan untuk set latih (training set) dan 27 ulasan dialokasikan untuk set uji (test set). Set latih digunakan untuk melatih model klasifikasi, sementara set uji digunakan untuk menguji kinerja model tersebut.

```
Training set size: 108
Test set size: 27
Training set samples:
                                                  text    label
26                          transaksi sangat mudah baik  Positif
109                                      baik apknya  Negatif
51   versi upgrade ubah tak sering saldo pulsa moho...   Netral
76                                              baik   Netral
132                                             baik  Negatif

Test set samples:
                                                  text    label
134                                     smga lbh baik  Positif
97   yang susah sangat salah pin masuk kantor kali ...  Negatif
124  temu semua sangat perlu parah pagi nasabah moh...  Negatif
4                                 kerja baik aplikasi  Positif
63                                      sekali baik  Positif
```

Gambar 4.40 Hasil Split data

Pada gambar 4.41 di atas merupakan kode program perintah untuk melakukan klasifikasi teks.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score

# Buat pipeline dengan TF-IDF dan Multinomial Naive Bayes
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Prediksi menggunakan data uji
y_pred = model.predict(X_test)

# Evaluasi model
print("Hasil Evaluasi Model cluster 2:")
print(classification_report(y_test, y_pred))
print(f"Akurasi Model cluster 2: {accuracy_score(y_test, y_pred)}")
```

Gambar 4.41 Klasifikasi Cluster 2

Gambar 4.42 hasil dari klasifikasi pada cluster 2 menunjukkan bahwa model memiliki tingkat akurasi sebesar 66.67%. Model ini mampu memprediksi label ‘Positif’ dengan baik, mencapai nilai recall sebesar 1.00 untuk kelas tersebut, namun memiliki performa yang kurang memuaskan untuk kelas

‘Negatif’ dan ‘Netral’ dengan recall dan precision yang rendah, bahkan mencapai nol untuk beberapa kelas.

```
Hasil Evaluasi Model cluster 2:
              precision    recall  f1-score   support

     Negatif       0.00      0.00      0.00         6
      Netral       0.00      0.00      0.00         3
     Positif       0.67      1.00      0.80        18

    accuracy                           0.67        27
   macro avg       0.22      0.33      0.27        27
weighted avg       0.44      0.67      0.53        27

Akurasi Model cluster 2: 0.6666666666666666
```

Gambar 4.42 Hasil Klasifikasi cluster 2

4. Cluster 3

Kode program di bawah ini digunakan untuk memberi label pada ulasan yang terdapat dalam cluster 3 berdasarkan skor/rating ulasan yang sudah ada sebelumnya. Setelah diberi label, data yang termasuk dalam cluster 3 disimpan dalam file CSV dengan nama ‘label_cluster3.csv’. Hasilnya menunjukkan bahwa dari jumlah total ulasan dalam cluster 3, terdapat sejumlah ulasan yang dinilai sebagai ‘Positif’, ‘Netral’, dan ‘Negatif’, dengan jumlah masing-masing label yang tertampil dalam keluaran.

```python
import pandas as pd
# Membaca data dari file CSV atau dari data yang sudah ada
df_data = pd.read_csv('Clusternan.csv', delimiter=';')  # 
Ubah 'Clusternan.csv' dengan nama file CSV Anda

# Fungsi untuk memberi label berdasarkan skor
def label_review(score):
    if score == 5:
        return 'Positif'
    elif score >= 3:
        return 'Netral'
    else:
        return 'Negatif'

# Memberi label pada cluster 3
```

```
df_data.loc[df_data['cluster'] == 3, 'label'] =
df_data[df_data['cluster'] == 3]['score'].apply(label_review)

# Memilih hanya data dari cluster 3 untuk disimpan
cluster_3_data = df_data[df_data['cluster'] == 3]
# Menghitung jumlah data dalam cluster 3
num_data_cluster_3 = len(cluster_3_data)
print(f"Jumlah data dalam cluster 3: {num_data_cluster_3}")

# Menghitung jumlah masing-masing label
label_counts = cluster_3_data['label'].value_counts()
num_positif = label_counts.get('Positif', 0)
num_netral = label_counts.get('Netral', 0)
num_negatif = label_counts.get('Negatif', 0)

print(f"Jumlah Positif: {num_positif}")
print(f"Jumlah Netral: {num_netral}")
print(f"Jumlah Negatif: {num_negatif}")
# Menampilkan beberapa ulasan dari cluster 3 dengan label
print(f"Cluster 3 with assigned labels:")
print(cluster_3_data[['text', 'score', 'label']].head())

# Menyimpan hasil dari cluster 3 ke dalam file CSV
cluster_3_data.to_csv('label_cluster3.csv', index=False)
```

Hasil dari gambar 4.43 pada pelabelan cluster 3 menunjukkan bahwa sebagian besar ulasan di dalamnya cenderung negatif, dengan lebih dari dua pertiga dari total ulasan (1087 dari 1622) diberi label ‘Negatif’. Di sisi lain, terdapat 328 ulasan yang mendapat label ‘Positif’’ dan 207 ulasan yang mendapat label ‘Netral’.

```
Jumlah data dalam cluster 3: 1622
Jumlah Positif: 328
Jumlah Netral: 207
Jumlah Negatif: 1087
Cluster 3 with assigned labels:
                                                text  score    label
0  terus sering sendiri mohon mobile mau lot kelu...    2.0  Negatif
1  yang ujung transfer sulit padahal nomer minta ...    1.0  Negatif
2  udah sama ribet padahal mobile mandiri bni ban...    1.0  Negatif
3  virtual transfer transaksi sering sedot saldo ...    1.0  Negatif
4  ulang sulit sering sangat prose nasabah minta ...    3.0   Netral
```

Gambar 4.43 Hasil Pelabelan Cluster 3

Pada gambar 4.44 di bawah ini merupakan perintah untuk melakukan split data pada cluster 3.

```
from sklearn.model_selection import train_test_split

# Membaca data yang sudah diberi label dari file CSV
df_data = pd.read_csv('label_cluster3.csv')

X = df_data['text']
y = df_data['label']

# Membagi data menjadi data latih (80%) dan data uji (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Menampilkan ukuran set latih dan uji
print(f'Training set size: {len(X_train)}')
print(f'Test set size: {len(X_test)}')

# Menampilkan beberapa sampel dari set latih dan uji
train_data = pd.DataFrame({'text': X_train, 'label': y_train})
test_data = pd.DataFrame({'text': X_test, 'label': y_test})

print("Training set samples:")
print(train_data.head())
print("\nTest set samples:")
print(test_data.head())
```

Gambar 4.44 Split Data

Gambar 4.45 yang mana dengan ukuran training set sebanyak 1297 sampel dan test set sebanyak 325 sampel, data telah terbagi dengan label-label yang mencerminkan sentimen ulasan seperti positif, negatif, dan netral. Data pelatihan berisi contoh-contoh ulasan yang bervariasi, sedangkan data uji mencakup sampel-sampel yang digunakan untuk mengevaluasi performa model.

```
Training set size: 1297
Test set size: 325
Training set samples:
                                                  text     label
447   user uang tiba terus paksa pakek nya nasabah m...  Negatif
688   yang pulsa nomor ngak malah kirim hilang gopay...  Positif
216   yang tidak terus solusi sinyal sim sendiri rib...  Negatif
1569  terus susah padahal login loading jaring hp ba...  Negatif
1353         suruh mulu malah eror dong benerin aktivasi  Negatif

Test set samples:
                                                  text     label
1190  yang transaksi skrng mudah mobile lebih lanjut...   Netral
1111  ujung transaksi server sedang pa jelek jelas e...  Negatif
718                                        suka eror  Negatif
1054                                           jelek  Negatif
1292                                          daftar  Negatif
```

Gambar 4.45 Hasil

Pada gambar 4.46 di bawah merupakan kode program perintah untuk melakukan klasifikasi teks. Tahapannya meliputi pembuatan pipeline untuk menggabungkan TF-IDF dan Naive Bayes, pelatihan model menggunakan data latih, prediksi label menggunakan data uji, serta evaluasi performa model dengan mencetak laporan klasifikasi dan menghitung akurasi.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score

# Buat pipeline dengan TF-IDF dan Multinomial Naive Bayes
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Prediksi menggunakan data uji
y_pred = model.predict(X_test)

# Evaluasi model
print("Hasil Evaluasi Model cluster 3:")
print(classification_report(y_test, y_pred))
print(f"Akurasi Model cluster 3: {accuracy_score(y_test, y_pred)}")
```

Gambar 4.46 Klasifikasi Cluster 3

Gambar 4.47 menunjukan presisi baik untuk ulasan negatif (0.77, recall 0.99), untuk ulasan netral (presisi 1.00, recall 0.05) dan ulasan positif (presisi 0.82, recall 0.55). Akurasi total 0.78,

```
Hasil Evaluasi Model cluster 3:
              precision    recall  f1-score   support

     Negatif       0.77      0.99      0.87       218
      Netral       1.00      0.05      0.09        41
     Positif       0.82      0.55      0.65        66

    accuracy                           0.78       325
   macro avg       0.86      0.53      0.54       325
weighted avg       0.81      0.78      0.73       325

Akurasi Model cluster 3: 0.7815384615384615
```

Gambar 4.47 Hasil klasifikasi cluster 3

Tabel 4 Hasil Klasifikasi

No	Cluster	Hasil Klasifikasi
1	Cluster 0	Negatif
2	Cluster 1	Positif
3	Cluster 2	Positif
4	Cluster 3	Negatif

Berdasarkan tabel 4 analisis pada empat cluster yang telah dilakukan, ditemukan bahwa cluster 0 terkait Masalah Teknis dalam Penggunaan Aplikasi hasil sentimennya yaitu ‘negatif’ (-). Cluster 1 terkait Kepuasan dan Kemudahan Penggunaan hasil sentimennya yaitu ‘positif’ (+). Cluster 2 terkait Layanan dan Keamanan hasil sentimennya yaitu ‘positif’ (+). Cluster 3 terkait operasional dan pengalaman pengguna hasil sentimennya ‘negatif’ (-).

Berdasarkan penelitian ini, rekomendasi untuk BSB adalah fokus pada perbaikan pada Cluster 0 dan Cluster 3. Dimana cluster 0 rata-rata Pengguna sering menghadapi kesulitan dalam mengakses aplikasi, melakukan aktivasi akun, dan proses transaksi. Keluhan tentang aplikasi yang sering mengalami error, ribet, atau tidak berfungsi dengan baik juga umum. dan untuk cluster 3 kegagalan dalam transfer dan transaksi, serta proses aktivasi yang sering tidak berhasil. Masalah aplikasi yang sering error dan memerlukan update teratur untuk meningkatkan kinerja.