BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

Dalam penelitian ini, digunakan empat metode yang terdiri dari Naïve Bayes, K-Nearest Neighbor, Naïve Bayes+PSO, dan K-Nearest Neighbor+PSO. Data yang digunakan dalam penelitian ini adalah data komentar pada aplikasi ID Express Customer pada Google Playstore dengan jumlah total 51.250 data. Dilakukan tahap preprocessing pada data tersebut sehingga jumlah data yang siap diolah menjadi 42.594. Data sampel ini akan digunakan untuk menghitung probabilitasnya, mencari tingkat akurasi tertinggi, dan mengimplementasikannya pada platform Google Colab dengan bahasa python.

4.1.1 Import Data

Dataset yang digunakan pada penelitan ini, berupa data dengan format .csv yang telah siap dan sudah melalui tahap *preprocessing* akan di *import* ke dalam Google Colab dengan bahasa python. Dengan mengambil dataset dari penyimpanan google drive menggukan *source code* seperti pada gambar 4.1.

```
[ ]  from google.colab import drive
     drive.mount("/content/drive")
```

Gambar 4.1 Import Dataset

Setelah menghubungkan Google Drive ke sesi Colab, kita dapat mengakses file dan direktori di Google Drive. Selanjutnya, kita akan mengubah direktori kerja saat ini ke direktori utama di Google Drive.

Selanjutnya kita akan menarik data dari direktori dan menampilkan data yang akan digunakan. Untuk menarik data kita menggunakan *source code* seperti pada gambar 4.2.

```
[ ]  import pandas as pd
     path = '/content/drive/MyDrive/Cendy/Data_Sentimen.csv'
     pd.set_option('display.max_colwidth', None)
     df = pd.read_csv(path)
     df = df.drop(df.columns[0], axis=1)
     df
```

Gambar 4.2 *Source Code* untuk menampilkan dataset

Setelaha menjalankan perintah diatas, maka akan terlihat dataset yang nantinya akan digunakan, dapat dilihat pada gambar 4.3.

	cln_content	Subjectivity	Polarity	Sentiment
0	bad service	0.666667	-0.700000	Negative
1	aja	0.125000	0.000000	Neutral
2	kecewa proses pengiriman kurir customer udh sampe andong rumah pengiriman kurim lambat banget	0.756667	-0.346667	Negative
3	mesan barang shopee pengiriman id express udah 4 gak dikirim kerumah statusnya udah sampe kota	0.000000	0.000000	Neutral
4	aplikasi nya bagus pelayanannya terbaik memuaskan jasa pengiriman nya super cepet kurir nya rapi sopan semangat id express the best pkonya patut coba pakai jasa expedisi	0.509524	0.576190	Positive
...	...	...	...	...
223	mengecewakan	0.700000	-0.600000	Negative
224	kirim kamis 16 2 dr palembang sabtu 18 2 menjelang sore paket sdh chat email online berkali kali no respon minimal respon robot kek gk bad banget no solusi pakai idexpress	0.583333	0.425000	Positive
225	instal aplikasi nyuri data google	0.000000	0.000000	Neutral
226	paket tertahan udah 2 sekota loh	0.000000	0.000000	Neutral
227	kurir ajar	0.000000	0.000000	Neutral

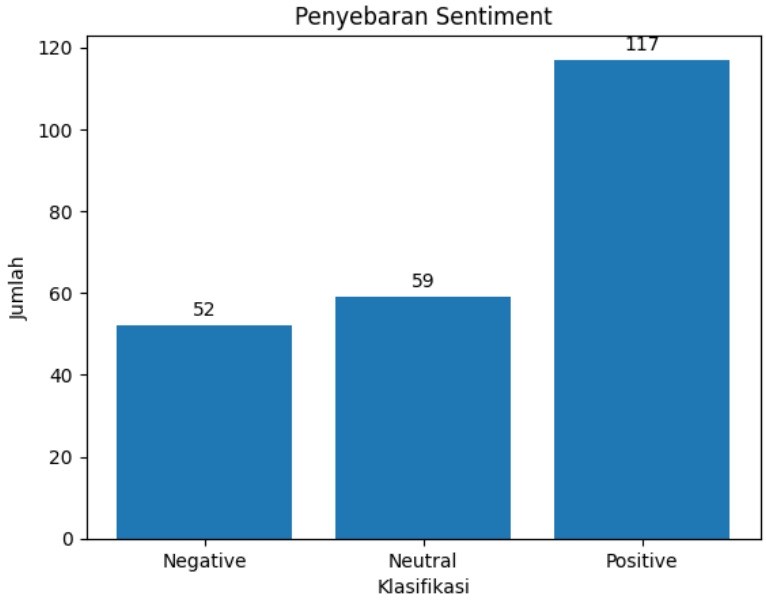
228 rows × 4 columns

Gambar 4.3 Tampilan dataset pada Google Colab

Dari dataset tersebut, kita dapat melihat penyebaran sentiment yang ada pada dataset dengan memvisualisaikannya menggunakan *Source Code* pada gambar 4.4. Dan pada gambar 4.5 merupakan hasil dari visualisai tersebut.

```
from collections import Counter
import matplotlib.pyplot as plt

target_cnt = Counter(df['Sentiment'])
label = target_cnt.keys()
values = target_cnt.values()

fig, ax = plt.subplots()

bar_container = ax.bar(label, values)

ax.set(ylabel="Jumlah", xlabel="Klasifikasi", title="Penyebaran Sentiment")

# Adding labels on top of each bar
for rect in bar_container:
    height = rect.get_height()
    ax.annotate('{}'.format(height), xy=(rect.get_x() + rect.get_width() / 2, height), xytext=(0, 3),
                textcoords="offset points", ha='center', va='bottom')

plt.show()
```

Gambar 4.4 Source Code untuk Visualisai Penyebaran Sentiment Dataset



Gambar 4.5 Visualisasi Penyebaran Sentiment Dataset

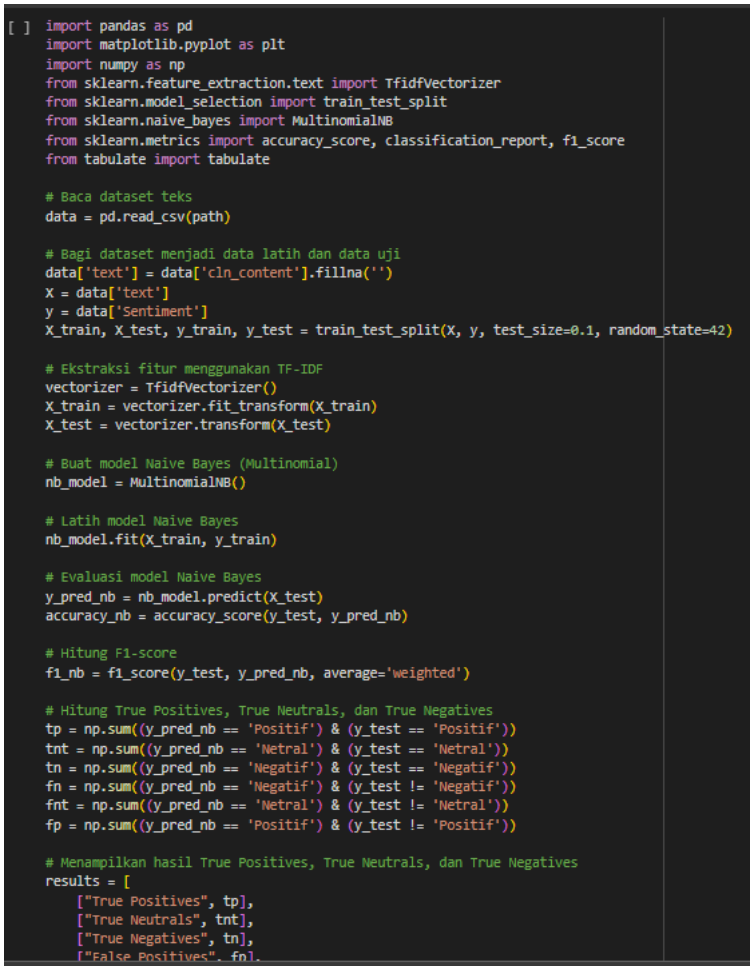
Pada data visualisai diatas, kita dapat lihat bawah pada dataset yang digunakan terdapat penyebaran data sentiment. Dimana data dengan nilai Negatif bejumlah 52 data teks, Positif dengan jumlah 117 data teks dan Netral sejumlah 59 data teks.

Adapun, kata-kata yang sering muncul pada dataset menunjukan keberagaram kata pada data. Data tersebut divisualisaikan menggunakan fitur *WordCloud* yang dapat kita lihat hasil visualisainya pada gambar 4.6.

Gambar 4.6 Visualisasi *WordCloud* yang paling banyak muncul

4.1.2 Medote *Naïve Bayes*

Pada tahap ini, dataset yang digunakan sudah di import kedalam Google Colab dan akan dilakukan klasifikasi dengan menggunakan model algoritma *Naïve Bayes.* Berikut souce code untuk metode algoritma *Naïve Bayes* menggunakan Bahasa pemograman *Pyton.*



```
[ ]  import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.model_selection import train_test_split
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.metrics import accuracy_score, classification_report, f1_score
     from tabulate import tabulate

     # Baca dataset teks
     data = pd.read_csv(path)

     # Bagi dataset menjadi data latih dan data uji
     data['text'] = data['cln_content'].fillna('')
     X = data['text']
     y = data['Sentiment']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

     # Ekstraksi fitur menggunakan TF-IDF
     vectorizer = TfidfVectorizer()
     X_train = vectorizer.fit_transform(X_train)
     X_test = vectorizer.transform(X_test)

     # Buat model Naive Bayes (Multinomial)
     nb_model = MultinomialNB()

     # Latih model Naive Bayes
     nb_model.fit(X_train, y_train)

     # Evaluasi model Naive Bayes
     y_pred_nb = nb_model.predict(X_test)
     accuracy_nb = accuracy_score(y_test, y_pred_nb)

     # Hitung F1-score
     f1_nb = f1_score(y_test, y_pred_nb, average='weighted')

     # Hitung True Positives, True Neutrals, dan True Negatives
     tp = np.sum((y_pred_nb == 'Positif') & (y_test == 'Positif'))
     tnt = np.sum((y_pred_nb == 'Netral') & (y_test == 'Netral'))
     tn = np.sum((y_pred_nb == 'Negatif') & (y_test == 'Negatif'))
     fn = np.sum((y_pred_nb == 'Negatif') & (y_test != 'Negatif'))
     fnt = np.sum((y_pred_nb == 'Netral') & (y_test != 'Netral'))
     fp = np.sum((y_pred_nb == 'Positif') & (y_test != 'Positif'))

     # Menampilkan hasil True Positives, True Neutrals, dan True Negatives
     results = [
         ["True Positives", tp],
         ["True Neutrals", tnt],
         ["True Negatives", tn],
         ["False Positives", fp],
```

Gambar 4.7 *Source Code* Algoritma *Naïve Bayes*

Pada gambar 4.7, menunjukan program mengimpor pustaka yang diperlukan seperti *pandas* untuk membaca dan memanipulasi data, *matplotlib.pyplot* untuk membuat plot, *numpy* untuk operasi numerik, serta modul-modul dari sklearn untuk pemrosesan teks dan pembuatan model klasifikasi. Dataset teks dibaca dari file CSV menggunakan *pd.read_csv(path*), di mana path adalah lokasi file CSV yang berisi data. Data teks dibagi menjadi data latih dan data uji menggunakan *train_test_split*. Kolom 'text' digunakan sebagai fitur (X) dan kolom ‘Sentiment’ sebagai label (y).

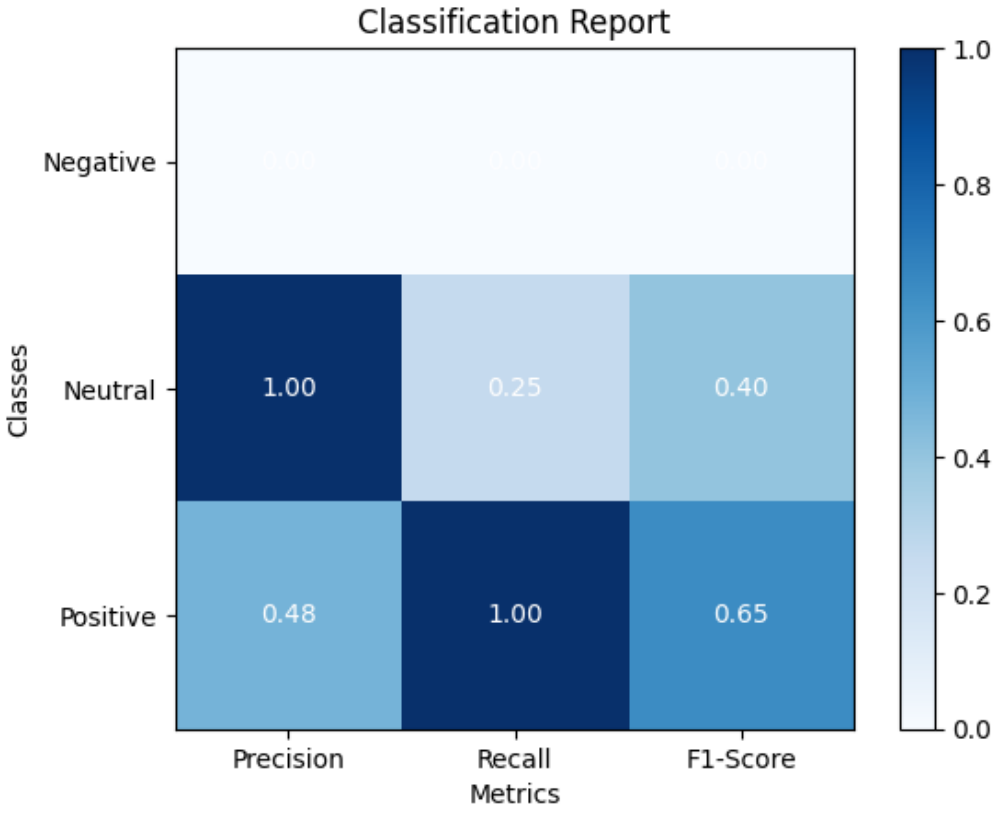
Kemudian, ekstraksi fitur teks menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*) dengan menggunakan *TfidfVectorizer* dari *sklearn*. TF-IDF sendiri mengukur pentingnya kata dalam suatu dokumen dalam konteks koleksi dokumen. Dimana, X_train dan X_test akan berisi representasi TF-IDF dari teks dalam data latih dan data uji. Model klasifikasi *Naïve Bayes* dibuat menggunakan *Naïve Bayes* dari *sklearn*. Model tersebut dilatih menggunakan data latih (X_train dan y_train) dengan memanggil fit pada objek

model. Lalu, model *Naïve Bayes* dievaluasi menggunakan data uji. Prediksi klasifikasi dibuat dengan memanggil *predict* pada model yang sudah dilatih. Akurasi model dihitung menggunakan *accuracy_score* yang membandingkan label aktual (y_test) dengan prediksi (y_pred_nb). F1-score dihitung menggunakan *f1_score* dengan rata-rata tertimbang (*weighted*) untuk memperhitungkan ketidakseimbangan kelas.

Adapun hasil yang didapat dari uji coba menggunakan metode *Naïve Bayes* adalah sebagai berikut :

a) Hasil Uji Klasifikasi dengan perbandingan data 90:10)

Penerapan metode *Naïve Bayes* dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 90:10 dapat dilihat *Classification Report* yang diperolah pada gambar 4.8.



Gambar 4.8 *Classification Report* Metode *Naïve Bayes* (90:10)

Tabel 4.1 Matrik Konfusi Metode *Naïve Bayes* (90:10)

	Predicted Classification		
Classes	Negatif	Netral	Positif

Actual Classification				
	Negatif	0	0	0
	Netral	0	2	0
	Positif	11	11	10

Berdasarkan gambar diatas, pada pembagian data sampling 90% dan data testing 10% menggunakan metode *Naïve Bayes* maka didapatkan akurasi sebesar 52,17%. Dimana hasil ini didapatkan dari perhitungan seperti berikut:

Rumus: $Akurasi = \frac{TP+TN}{TP+TN+FP+FN}$

Keterangan :

- TP (True Positive): Jumlah data dengan kelas yang benar diprediksi sebagai positif.
- TN (True Negative): Jumlah data dengan kelas yang benar diprediksi sebagai negatif.
- FP (False Positive): Jumlah data dengan kelas yang salah diprediksi sebagai positif.
- FN (False Negative): Jumlah data dengan kelas yang salah diprediksi sebagai negatif.

Dari laporan klasifikasi di atas, kita memiliki:

- TP (Positif) = 10
- TNT (Netral) = 2
- TN (Negatif) = 0
- FP (Positif) = 11
- FNT (Netral) = 0
- FN (Negatif) = 0

Maka dapat dihitung akurasinya:

$$Akurasi = \frac{10+2+0}{10+2+0+11+0+0}$$
$$= \frac{12}{23}$$
$$= 52.17\%$$

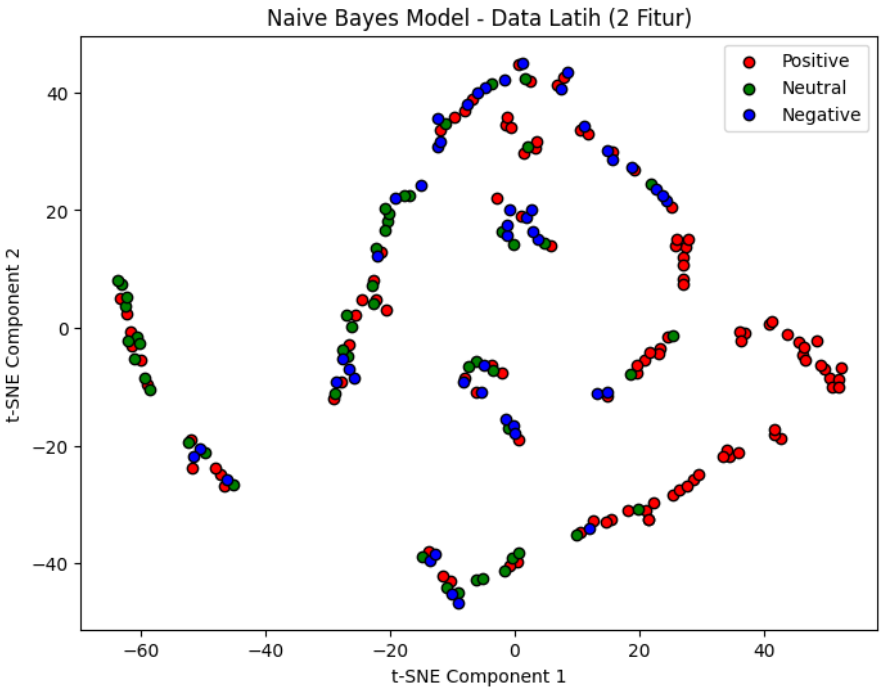
Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 42.17%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

Rumus F1-score:

$$\frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.00\times5+0.40\times8+0.65\times10}{23} = 42.17\%$$

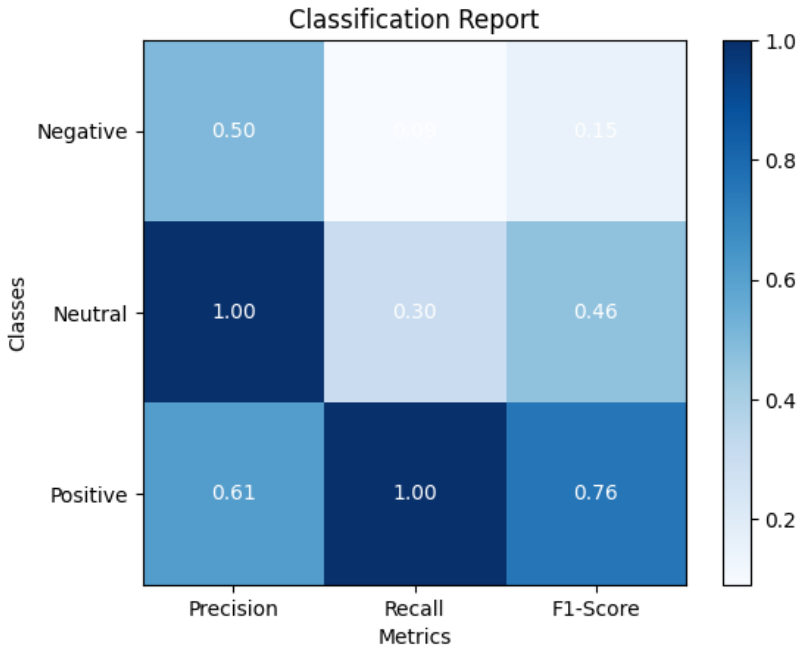
Adapun penyebaran klasifiksai data latih dari penerapan metode *Naïve Bayes* pada pembagian data sampling 90% dan data testing 10% dapat dilihat pada gambar 4.9.



Gambar 4.9 Penyebaran Klasifikasi Data Latih Metode *Naïve Bayes* (90:10)

b) Hasil Uji Klasifikasi dengan perbandingan data 80:20

Penerapan metode *Naïve Bayes* dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 80:20 dapat dilihat *Classification Report* yang diperolah pada gambar 4.10.



Gambar 4.10 *Classification Report* Metode *Naïve Bayes* (80:20)

Tabel 4.2 Matrik Konfusi Metode *Naïve Bayes* (80:20)

		Predicted Classification		
	Classes	Negatif	Netral	Positif
Actual Classification	Negatif	1	0	0
	Netral	0	3	0
	Positif	16	16	25

Berdasarkan gambar diatas, pada pembagian data sampling 80% dan data testing 20% menggunakan metode *Naïve Bayes* maka didapatkan akurasi sebesar 63,04%. Dimana hasil ini didapatkan dari perhitungan seperti berikut:

Rumus: $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN}$$

Dimana:

- TP (True Positive): Jumlah data dengan kelas yang benar diprediksi sebagai positif.
- TN (True Negative): Jumlah data dengan kelas yang benar diprediksi sebagai negatif.
- FP (False Positive): Jumlah data dengan kelas yang salah diprediksi sebagai positif.
- FN (False Negative): Jumlah data dengan kelas yang salah diprediksi sebagai negatif.

Dari laporan klasifikasi di atas, kita memiliki:

- TP (Positif) = 25
- TNT (Netral) = 3
- TN (Negatif) = 1
- FP (Positif) = 16
- FNT (Netral) = 0
- FN (Negatif) = 0

Maka dapat dihitung akurasinya:

$$Akurasi = \frac{25+3+1}{25+3+1+16+0+0}$$

$$= \frac{29}{45} = 64{,}44\%$$

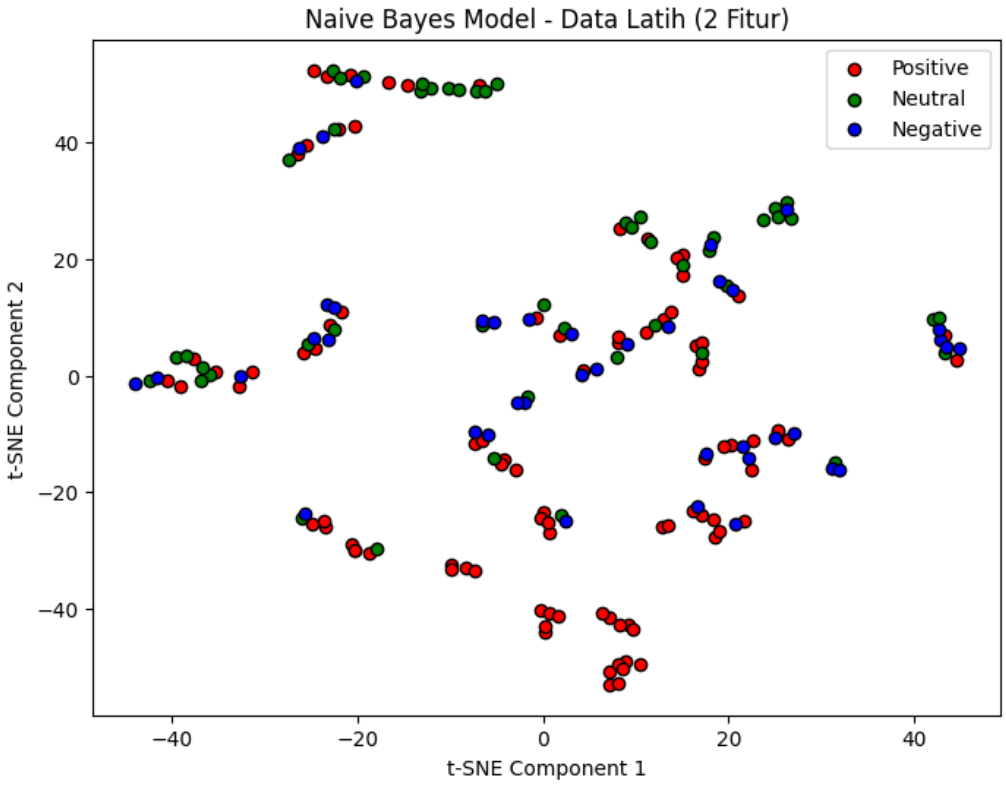
Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 54,89%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

Rumus F1-score:

$$= \frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.15\times11+0.46\times10+0.76\times25}{46} = 54{,}89\%$$

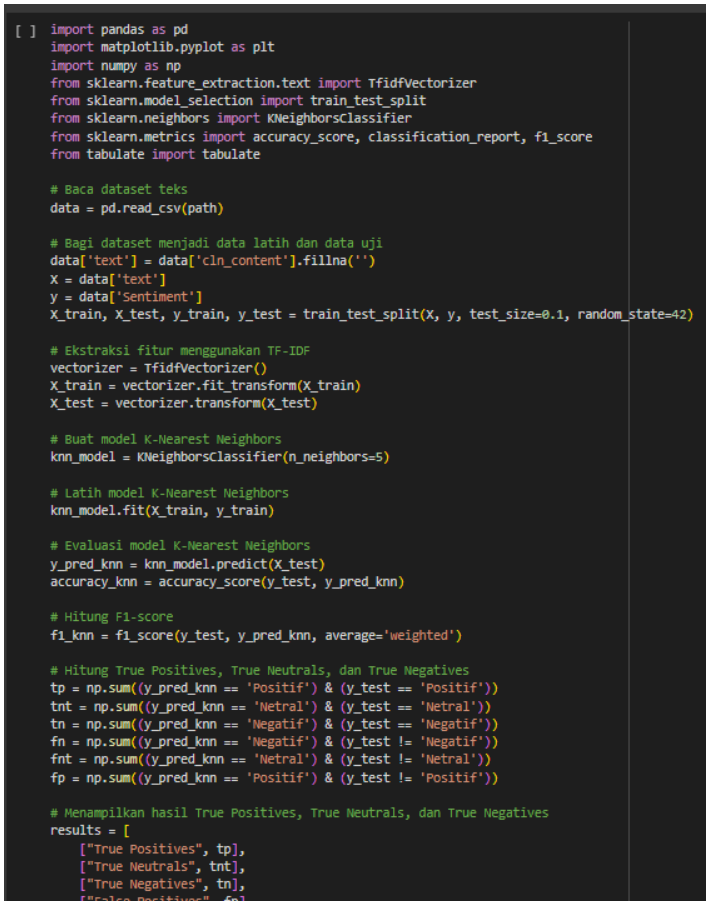
Adapun penyebaran klasifiksai data latih dari penerapan metode *Naïve Bayes* pada pembagian data sampling 80% dan data testing 20% dapat dilihat pada gambar 4.12.



Gambar 4.11 Penyebaran Klasifikasi Data Latih Metode *Naïve Bayes* (80:20)

4.1.3 Metode *K- Nearest Neighbor*

Pada tahap ini, dataset yang digunakan sudah di import kedalam Google Colab dan akan dilakukan klasifikasi dengan menggunakan model algoritma *K-Nearest Neighbor*. Berikut souce code untuk metode algoritma *K-Nearest Neighbor* menggunakan Bahasa pemograman *Pyton.*



```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.model_selection import train_test_split
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import accuracy_score, classification_report, f1_score
    from tabulate import tabulate

    # Baca dataset teks
    data = pd.read_csv(path)

    # Bagi dataset menjadi data latih dan data uji
    data['text'] = data['cln_content'].fillna('')
    X = data['text']
    y = data['Sentiment']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

    # Ekstraksi fitur menggunakan TF-IDF
    vectorizer = TfidfVectorizer()
    X_train = vectorizer.fit_transform(X_train)
    X_test = vectorizer.transform(X_test)

    # Buat model K-Nearest Neighbors
    knn_model = KNeighborsClassifier(n_neighbors=5)

    # Latih model K-Nearest Neighbors
    knn_model.fit(X_train, y_train)

    # Evaluasi model K-Nearest Neighbors
    y_pred_knn = knn_model.predict(X_test)
    accuracy_knn = accuracy_score(y_test, y_pred_knn)

    # Hitung F1-score
    f1_knn = f1_score(y_test, y_pred_knn, average='weighted')

    # Hitung True Positives, True Neutrals, dan True Negatives
    tp = np.sum((y_pred_knn == 'Positif') & (y_test == 'Positif'))
    tnt = np.sum((y_pred_knn == 'Netral') & (y_test == 'Netral'))
    tn = np.sum((y_pred_knn == 'Negatif') & (y_test == 'Negatif'))
    fn = np.sum((y_pred_knn == 'Negatif') & (y_test != 'Negatif'))
    fnt = np.sum((y_pred_knn == 'Netral') & (y_test != 'Netral'))
    fp = np.sum((y_pred_knn == 'Positif') & (y_test != 'Positif'))

    # Menampilkan hasil True Positives, True Neutrals, dan True Negatives
    results = [
        ["True Positives", tp],
        ["True Neutrals", tnt],
        ["True Negatives", tn],
```

Gambar 4.12 *Source Code* Algoritma *K-Nearest Neighbor*

Pada gambar 4.12 menunjukan program mengimpor pustaka yang diperlukan seperti *pandas* untuk membaca dan memanipulasi data, *matplotlib.pyplot* untuk membuat plot, *numpy* untuk operasi numerik, serta modul-modul dari sklearn untuk pemrosesan teks dan pembuatan model klasifikasi. Dataset teks dibaca dari file CSV menggunakan *pd.read_csv(path*), di mana path adalah lokasi file CSV yang berisi data. Data teks dibagi menjadi data latih dan data uji menggunakan *train_test_split*. Kolom 'text' digunakan sebagai fitur (X) dan kolom ‘Sentiment’ sebagai label (y).

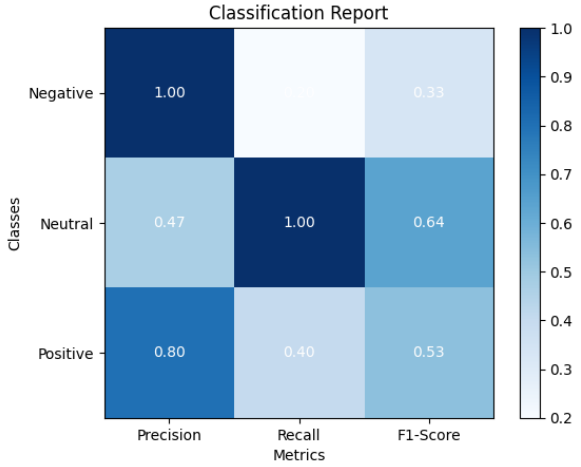
Kemudian, ekstraksi fitur teks menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*) dengan menggunakan *TfidfVectorizer* dari *sklearn*. TF-IDF sendiri mengukur pentingnya kata dalam suatu dokumen dalam konteks koleksi dokumen. Dimana, X_train dan X_test akan berisi representasi TF-IDF dari teks dalam data latih dan data uji. Model klasifikasi *K-Nearest Neighbor* dibuat menggunakan *K-Nearest Neighbor* dari *sklearn*. Model tersebut dilatih menggunakan data latih (X_train dan y_train) dengan memanggil fit pada objek model. Lalu, model *K-Nearest Neighbor* dievaluasi

menggunakan data uji. Prediksi klasifikasi dibuat dengan memanggil *predict* pada model yang sudah dilatih. Akurasi model dihitung menggunakan *accuracy_score* yang membandingkan label aktual (y_test) dengan prediksi (y_pred_nb). F1-score dihitung menggunakan *f1_score* dengan rata-rata tertimbang (*weighted*) untuk memperhitungkan ketidakseimbangan kelas.

Adapun hasil yang didapat dari uji coba menggunakan metode *K-Nearest Neighbor* adalah sebagai berikut :

a) Hasil Uji Klasifikasi dengan perbandingan data 90:10

Penerapan metode *K-Nearest Neighbor* dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 90:10 dapat dilihat *Classification Report* yang diperolah pada gambar 4.13.



Gambar 4.13 *Classification Report* Metode *K-Nearest Neighbor* (90:10)

Tabel 4.3 Matrik Konfusi Metode *K-Nearest Neihgbor* (90:10)

		Predicted Classification		
	Classes	Negatif	Netral	Positif
Actual Classification	Negatif	1	0	0
	Netral	9	8	9
	Positif	1	1	4

Berdasarkan gambar diatas, pada pembagian data sampling 90% dan data testing 10% menggunakan metode *K-Nearest Neighbor* maka didapatkan akurasi sebesar 56,52%. Dimana hasil ini didapatkan dari perhitungan seperti berikut:

Rumus: $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN}$$

Dimana:

- TP (True Positive): Jumlah data dengan kelas yang benar diprediksi sebagai positif.
- TN (True Negative): Jumlah data dengan kelas yang benar diprediksi sebagai negatif.
- FP (False Positive): Jumlah data dengan kelas yang salah diprediksi sebagai positif.
- FN (False Negative): Jumlah data dengan kelas yang salah diprediksi sebagai negatif.

Dari laporan klasifikasi di atas, kita memiliki:

- TP (Positif) = 4
- TNT (Netral) = 8
- TN (Negatif) = 1
- FP (Positif) = 1
- FNT (Netral) = 9
- FN (Negatif) = 0

Maka dapat dihitung akurasinya:

$$Akurasi = \frac{4+8+1}{4+8+1+1+9+0}$$
$$= \frac{13}{23}$$
$$= 56.52\%$$

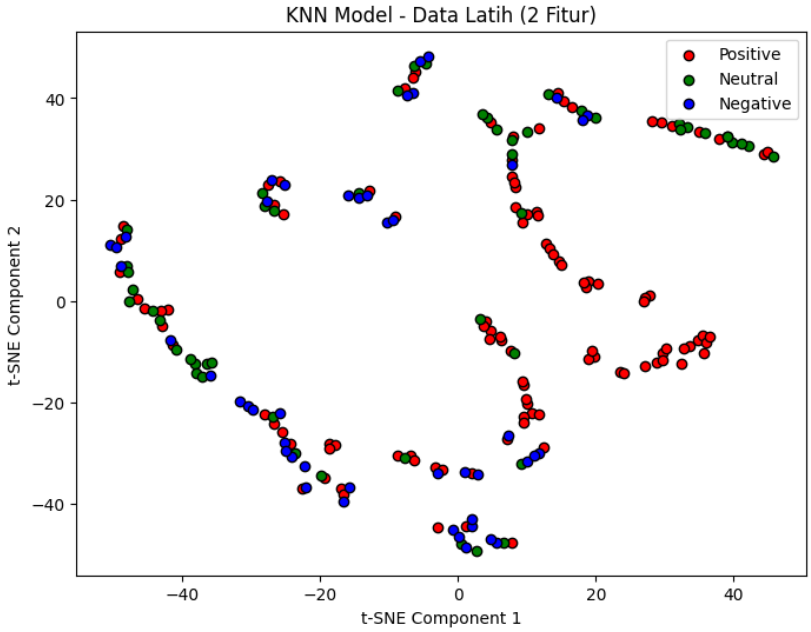
Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 52,47%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

Rumus F1-score:

$$= \frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.33\times5+0.64\times8+0.53\times10}{23} = 52.47\%$$

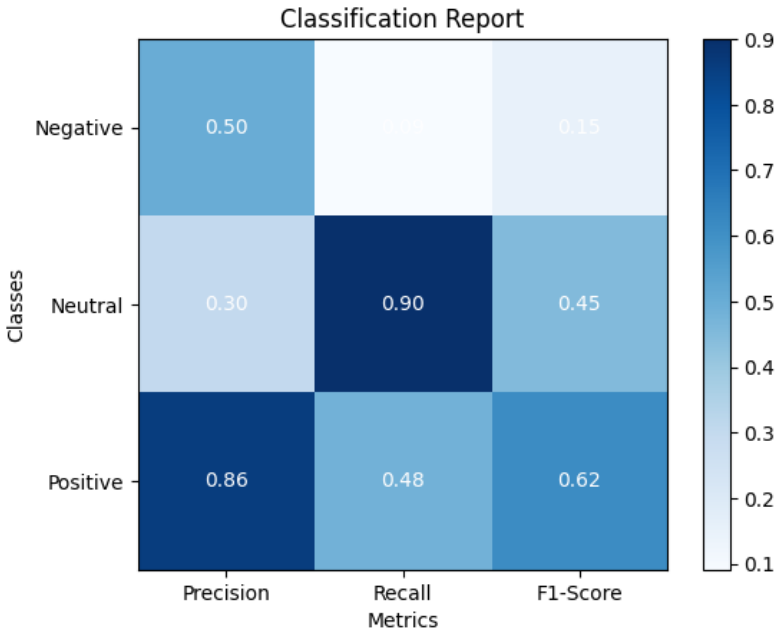
Adapun penyebaran klasifiksai data latih dari penerapan metode *K-Nearest Neighbor* pada pembagian data sampling 90% dan data testing 10% dapat dilihat pada gambar 4.14.



Gambar 4.14 Penyebaran Klasifikasi Data Latih Metode *K-Nearest Neighbor* (90:10)

b) Hasil Uji Klasifikasi dengan perbandingan data 80:20

Penerapan metode *K-Nearest Neighbor* dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 80:20 dapat dilihat *Classification Report* yang diperolah pada gambar 4.15.



Gambar 4.15 *Classification Report* Metode *K-Nearest Neighbor* (80:20)

Tabel 4.4 Matrik Konfusi Metode *K-Nearest Neighbor* (80:20)

		Predicted Classification		
	Classes	Negatif	Netral	Positif
Actual Classification	Negatif	1	0	0
	Netral	21	9	21
	Positif	2	2	12

Berdasarkan gambar diatas, pada pembagian data sampling 80% dan data testing 20% menggunakan metode *K-Nearest Neighbor* maka didapatkan akurasi sebesar 48,88%. Dimana hasil ini didapatkan dari perhitungan seperti berikut:

Rumus: $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN}$$

Dimana:

- TP (True Positive): Jumlah data dengan kelas yang benar diprediksi sebagai positif.
- TN (True Negative): Jumlah data dengan kelas yang benar diprediksi sebagai negatif.
- FP (False Positive): Jumlah data dengan kelas yang salah diprediksi sebagai positif.
- FN (False Negative): Jumlah data dengan kelas yang salah diprediksi sebagai negatif.

Dari laporan klasifikasi di atas, kita memiliki:

- TP (Positif) = 12
- TNT (Netral) = 9
- TN (Negatif) = 1
- FP (Positif) = 2
- FNT (Netral) = 21
- FN (Negatif) = 0

Maka dapat dihitung akurasinya:

$$Akurasi = \frac{12+9+1}{12+9+1+2+21+0}$$
$$= \frac{22}{45}$$
$$= 48.88\%$$

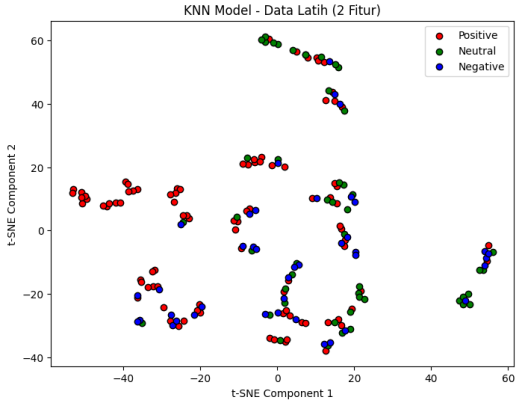
Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 47,06%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

Rumus F1-score:

$$= \frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.15\times11+0.45\times10+0.62\times25}{46} = 47{,}06\%$$

Adapun penyebaran klasifikasi data latih dari penerapan metode *K-Nearest Neighbor* pada pembagian data sampling 80% dan data testing 20% dapat dilihat pada gambar 4.16.



Gambar 4.16 Penyebaran Klasifikasi Data Latih Metode *K-Nearest Neighbor* (80:20)

4.1.4 Metode *Naïve Bayes* dan PSO

Pada tahap ini, dataset yang digunakan sudah di import kedalam Google Colab dan akan dilakukan klasifikasi dengan menggunakan model algoritma *Naïve Bayes* dan di optimasi menggunakan PSO. Berikut souce code untuk metode algoritma *Naïve Bayes* dan PSO menggunakan Bahasa pemograman *Pyton.*

```
[ ] import pandas as pd
    import numpy as np
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.metrics import accuracy_score, classification_report, f1_score
    from pyswarm import pso
    from tabulate import tabulate

    # Baca dataset teks
    data = pd.read_csv(path)

    # Bagi dataset menjadi data latih dan data uji
    data['text'] = data['cln_content'].fillna('')
    X = data['text']
    y = data['Sentiment']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

    # Ekstraksi fitur menggunakan TF-IDF
    vectorizer = TfidfVectorizer()
    X_train = vectorizer.fit_transform(X_train)
    X_test = vectorizer.transform(X_test)

    # Fungsi tujuan untuk optimasi PSO
    def objective_function(params):
        alpha = params[0]
        nb_model = MultinomialNB(alpha=alpha)
        nb_model.fit(X_train, y_train)
        y_pred = nb_model.predict(X_test)
        return 1 - accuracy_score(y_test, y_pred)

    # Batasan PSO
    lb = [0.0]  # Lower bound untuk alpha (smoothing)
    ub = [1.0]  # Upper bound untuk alpha (smoothing)

    # Optimasi PSO
    best_params, _ = pso(objective_function, lb, ub)

    # Buat model Naive Bayes dengan parameter terbaik
    best_alpha = best_params[0]
    optimized_nb_model = MultinomialNB(alpha=best_alpha)

    # Latih model Naive Bayes yang dioptimalkan
    optimized_nb_model.fit(X_train, y_train)

    # Evaluasi model Naive Bayes yang dioptimalkan
    y_pred_optimized_nb = optimized_nb_model.predict(X_test)
    accuracy_optimized_nb = accuracy_score(y_test, y_pred_optimized_nb)
```

Gambar 4.17 *Source Code* Algoritma *Naïve Bayes*+PSO

Pada gambar 4.17 menunjukan program mengimpor pustaka yang diperlukan seperti *pandas* untuk membaca dan memanipulasi data, *matplotlib.pyplot* untuk membuat plot, *numpy* untuk operasi numerik, serta modul-modul dari sklearn untuk pemrosesan teks dan pembuatan model klasifikasi. Dataset teks dibaca dari file CSV menggunakan *pd.read_csv(path*), di mana path adalah lokasi file CSV yang berisi data. Data teks dibagi menjadi data latih dan data uji menggunakan *train_test_split*. Kolom 'text' digunakan sebagai fitur (X) dan kolom ‘Sentiment’ sebagai label (y).

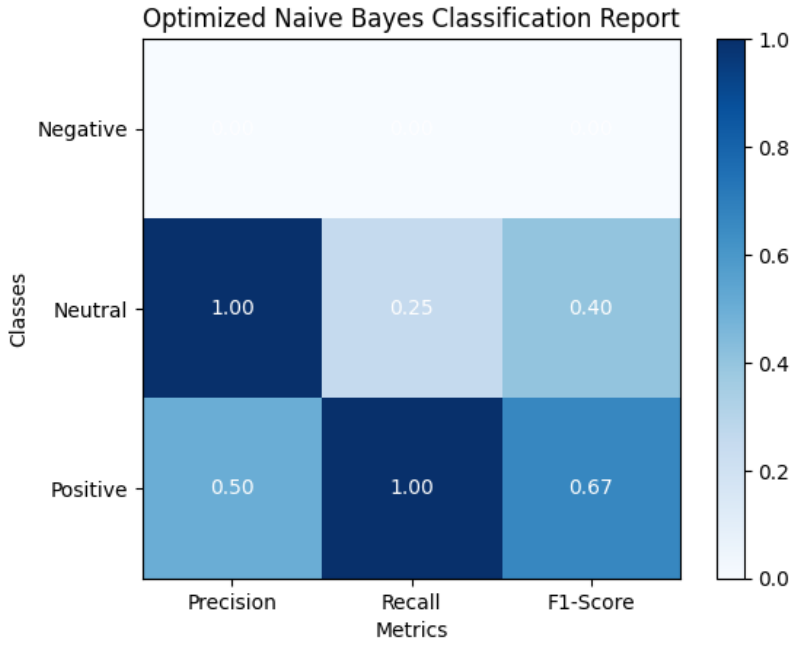
Kemudian, ekstraksi fitur teks menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*) dengan menggunakan *TfidfVectorizer* dari *sklearn*. TF-IDF sendiri mengukur pentingnya kata dalam suatu dokumen dalam

konteks koleksi dokumen. Dimana, X_train dan X_test akan berisi representasi TF-IDF dari teks dalam data latih dan data uji. Model klasifikasi *Niave Bayes* dibuat menggunakan *Naïve Bayes* dari *sklearn*. Model tersebut dilatih menggunakan data latih (X_train dan y_train) dengan memanggil fit pada objek model. Lalu, model *Naïve Bayes* dievaluasi menggunakan data uji. Prediksi klasifikasi dibuat dengan memanggil *predict* pada model yang sudah dilatih. Akurasi model dihitung menggunakan *accuracy_score* yang membandingkan label aktual (y_test) dengan prediksi (y_pred_nb). F1-score dihitung menggunakan *f1_score* dengan rata-rata tertimbang (*weighted*) untuk memperhitungkan ketidakseimbangan kelas.

Adapun hasil yang didapat dari uji coba menggunakan metode *Naïve Bayes*+PSO adalah sebagai berikut :

a) Hasil Uji Klasifikasi dengan perbandingan data 90:10

   Penerapan metode *Naïve Bayes*+PSO dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 90:10 dapat dilihat *Classification Report* yang diperoleh pada gambar 4.18.



Gambar 4.18 *Classification Report* Metode *Naïve Bayes*+PSO(90:10)

Berdasarkan gambar diatas, pada pembagian data sampling 90% dan data testing 10% menggunakan metode *Naïve Bayes+PSO* maka didapatkan akurasi sebesar 52,17%.

Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 43,04%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

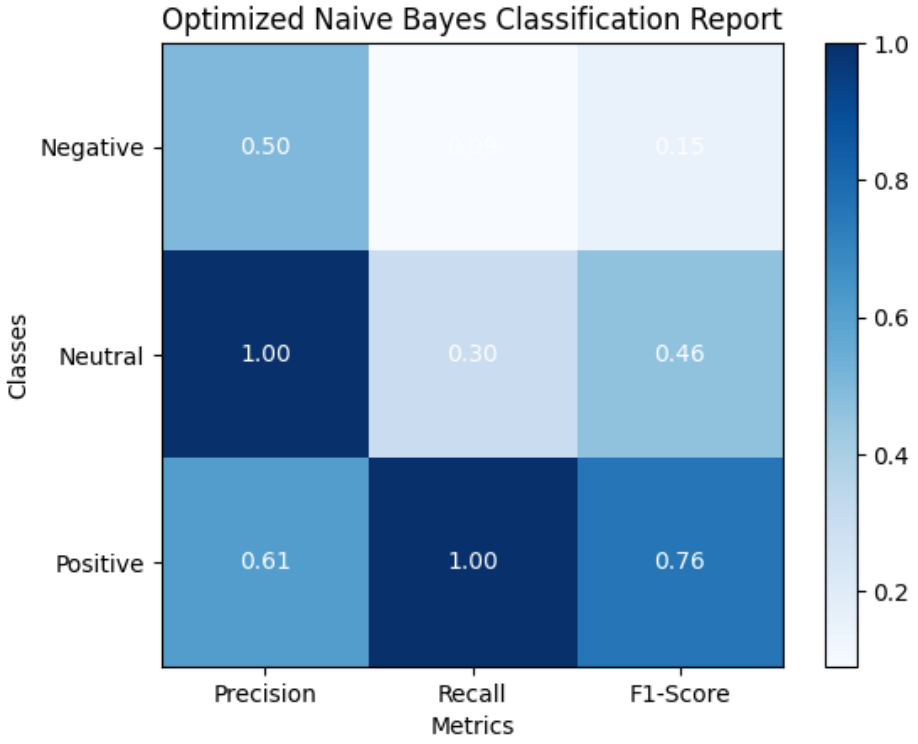
Rumus F1-score:

$$= \frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.00\times5+0.40\times8+0.67\times10}{23} = 43{,}04\%$$

b) Hasil Uji Klasifikasi dengan perbandingan data 80:20

Penerapan metode *Naïve Bayes*+PSO dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 80:20 dapat dilihat *Classification Report* yang diperolah pada gambar 4.19.



Gambar 4.19 *Classification Report* Metode *Naïve Bayes*+PSO (80:20)

Berdasarkan gambar diatas, pada pembagian data sampling 80% dan data testing 20% menggunakan metode *Naïve Bayes*+PSO maka didapatkan akurasi sebesar 63,04%.

Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 55%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan

seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

Rumus F1-score:

$$= \frac{F1 - score\ Negatif\ X\ Support\ Negatif\ +\ F1 - score\ Netral\ x\ Support\ Netral + F1 - score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1 - score = \frac{0.15\times11+0.46\times10+0.76\times25}{46} = 54{,}89\%$$

4.1.5 Metode *K-Nearest Neigbor* dan PSO

Pada tahap ini, dataset yang digunakan sudah di import kedalam Google Colab dan akan dilakukan klasifikasi dengan menggunakan model algoritma *K-Nearest Neighbor*. Berikut souce code untuk metode algoritma *K-Nearest Neighbor*+PSO menggunakan Bahasa pemograman *Pyton.*

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, f1_score
from sklearn.neighbors import KNeighborsClassifier
from pyswarm import pso
from tabulate import tabulate

# Baca dataset teks
data = pd.read_csv(path)

# Bagi dataset menjadi data latih dan data uji
data['text'] = data['cln_content'].fillna('')
X = data['text']
y = data['Sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# Ekstraksi fitur menggunakan TF-IDF
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Fungsi tujuan untuk optimasi PSO
def objective_function(params):
    k = int(params[0])
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train, y_train)
    y_pred = knn_model.predict(X_test)
    return 1 - accuracy_score(y_test, y_pred)

# Batasan PSO
lb = [1]  # Lower bound untuk jumlah tetangga (k)
ub = [10]  # Upper bound untuk jumlah tetangga (k)
```

Gambar 4.20 *Source Code* Algoritma *K-Nearest Neighbor*+PSO

Pada gambar 4.20 menunjukan program mengimpor pustaka yang diperlukan seperti *pandas* untuk membaca dan memanipulasi data, *matplotlib.pyplot* untuk membuat plot, *numpy* untuk operasi numerik, serta modul-modul dari sklearn untuk pemrosesan teks dan pembuatan model klasifikasi. Dataset teks dibaca dari file CSV menggunakan *pd.read_csv(path*), di mana path adalah lokasi file CSV yang berisi data. Data teks dibagi menjadi data latih dan data uji

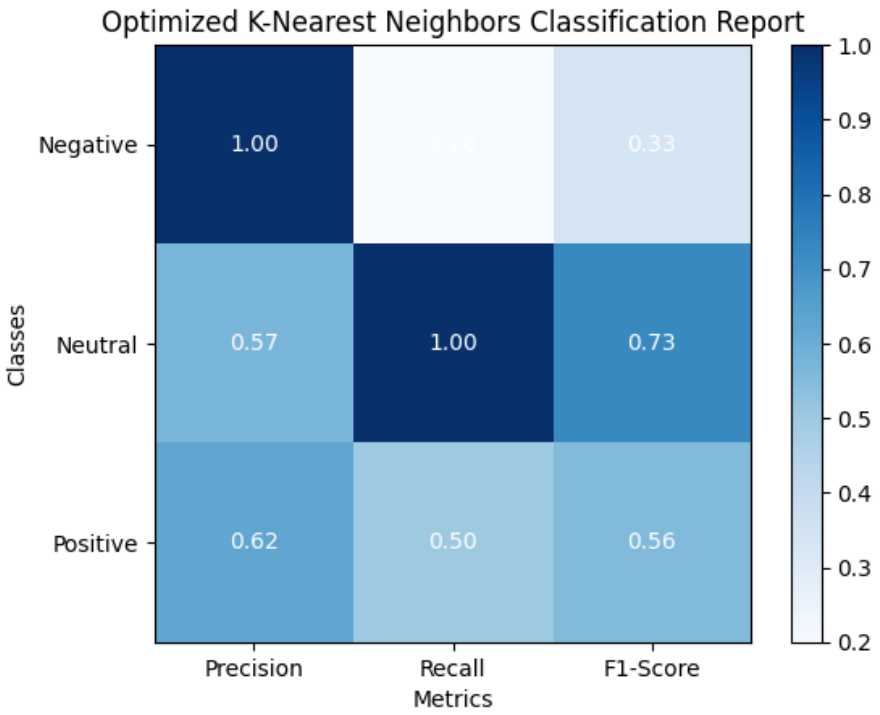
menggunakan *train_test_split*. Kolom 'text' digunakan sebagai fitur (X) dan kolom ‘Sentiment’ sebagai label (y).

Kemudian, ekstraksi fitur teks menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequenc*y) dengan menggunakan *TfidfVectorizer* dari *sklearn*. TF-IDF sendiri mengukur pentingnya kata dalam suatu dokumen dalam konteks koleksi dokumen. Dimana, X_train dan X_test akan berisi representasi TF-IDF dari teks dalam data latih dan data uji. Model klasifikasi *K-Nearest Neighbor*+PSO dibuat menggunakan *K-Nearest Neighbor* dari *sklearn*. Model tersebut dilatih menggunakan data latih (X_train dan y_train) dengan memanggil fit pada objek model. Lalu, model *K-Nearest Neighbor*+PSO dievaluasi menggunakan data uji. Prediksi klasifikasi dibuat dengan memanggil *predict* pada model yang sudah dilatih. Akurasi model dihitung menggunakan *accuracy_score* yang membandingkan label aktual (y_test) dengan prediksi (y_pred_nb). F1-score dihitung menggunakan *f1_score* dengan rata-rata tertimbang (*weighted*) untuk memperhitungkan ketidakseimbangan kelas.

Adapun hasil yang didapat dari uji coba menggunakan metode *K-Nearest Neighbor*+PSO adalah sebagai berikut :

a) Hasil Uji Klasifikasi dengan perbandingan data 90:10

   Penerapan metode *K-Nearest Neighbor*+PSO dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 90:10 dapat dilihat *Classification Report* yang diperolah pada gambar 4.21.



Gambar 4.21 *Classification Report* Metode *K-Nearest Neighbor*+PSO(90:10)

Tabel 4.5 Matrik Konfusi Metode *K-Nearest Neihgbor*+PSO (90:10)

		Predicted Classification		
	Classes	Negatif	Netral	Positif
Actual Classification	Negatif	1	0	0
	Netral	6	8	6
	Positif	3	3	5

Berdasarkan gambar diatas, pada pembagian data sampling 90% dan data testing 10% menggunakan metode *K-Nearest Neighbor* maka didapatkan akurasi sebesar 60,87%. Dimana hasil ini didapatkan dari perhitungan seperti berikut:

Rumus: $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN}$$

Dimana:

- TP (True Positive): Jumlah data dengan kelas yang benar diprediksi sebagai positif.
- TN (True Negative): Jumlah data dengan kelas yang benar diprediksi sebagai negatif.
- FP (False Positive): Jumlah data dengan kelas yang salah diprediksi sebagai positif.

- FN (False Negative): Jumlah data dengan kelas yang salah diprediksi sebagai negatif.

Dari laporan klasifikasi di atas, kita memiliki:

- TP (Positif) = 5
- TNT (Netral) = 8
- TN (Negatif) = 1
- FP (Positif) = 3
- FNT (Netral) = 6
- FN (Negatif) = 0

Maka dapat dihitung akurasinya:

$$Akurasi = \frac{5+8+1}{5+8+1+3+6+0}$$
$$= \frac{14}{23}$$
$$= 60{,}86\%$$

Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 56,91%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

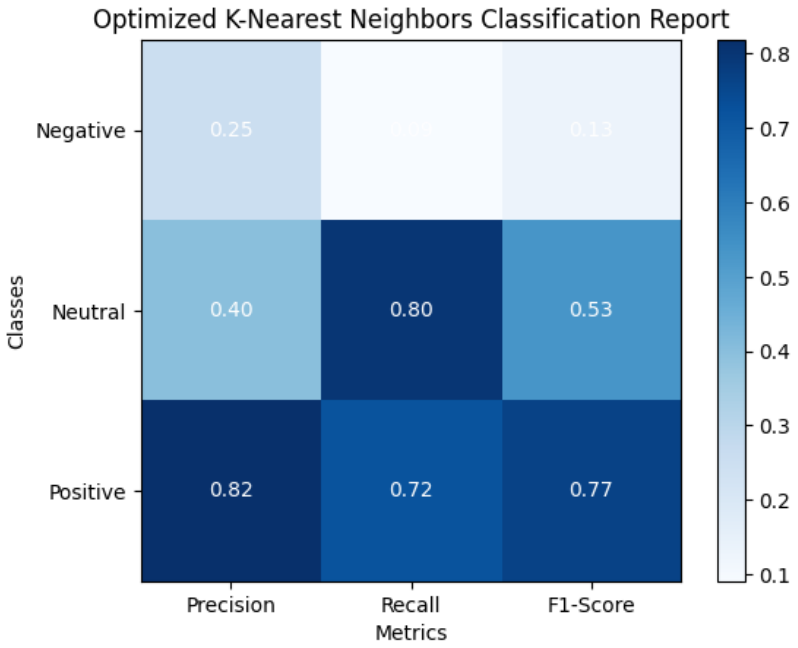
Rumus F1-score:

$$= \frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.33\times5+0.73\times8+0.56\times10}{23} = 56{,}91\%$$

b) Hasil Uji Klasifikasi dengan perbandingan data 80:20

Penerapan metode *K-Nearest Neighbor*+PSO dengan menggunakan split validation dengan nilai akurasi dan nilai F1-Score dengan pembagian data training dan data testing sebesar 80:20 dapat dilihat *Classification Report* yang diperolah pada gambar 4.15.



Gambar 4.22 *Classification Report* Metode *K-Nearest Neighbor* + PSO(80:20)

Tabel 4.6 Matrik Konfusi Metode *K-Nearest Neihgbor*+PSO (80:20)

		Predicted Classification		
	Classes	Negatif	Netral	Positif
Actual Classification	Negatif	1	3	3
	Netral	12	8	12
	Positif	4	4	18

Berdasarkan gambar diatas, pada pembagian data sampling 80% dan data testing 20% menggunakan metode *K-Nearest Neighbor*+PSO maka didapatkan akurasi sebesar 58,70%. Dimana hasil ini didapatkan dari perhitungan seperti berikut:

Rumus: $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN}$$

Dimana:

- TP (True Positive): Jumlah data dengan kelas yang benar diprediksi sebagai positif.
- TN (True Negative): Jumlah data dengan kelas yang benar diprediksi sebagai negatif.
- FP (False Positive): Jumlah data dengan kelas yang salah diprediksi sebagai positif.
- FN (False Negative): Jumlah data dengan kelas yang salah diprediksi sebagai negatif.

Dari laporan klasifikasi di atas, kita memiliki:

- TP (Positif) = 18
- TNT (Netral) = 8
- TN (Negatif) = 1
- FP (Positif) = 4
- FNT (Netral) = 12
- FN (Negatif) = 3

Maka dapat dihitung akurasinya:

$$Akurasi = \frac{18+8+1}{18+8+1+4+12+3}$$
$$= \frac{27}{46}$$
$$= 58{,}69\%$$

Sedangkan untuk nilai F1-score yang diperoleh dari data tersebut sebesar 56%. Nilai F1-score ini, didapatkan berdasarkan kinerja model dengan memperhitungkan seberapa besar setiap kelas berkontribusi terhadap hasil keseluruhan. Dan berikut merupakan perhitungannya:

Rumus F1-score:

$$= \frac{F1-score\ Negatif\ X\ Support\ Negatif\ +\ F1-score\ Netral\ x\ Support\ Netral + F1-score\ Positif\ xSupport\ Positif}{Total\ Support}$$

$$F1-score = \frac{0.13\times11+0.53\times10+0.77\times25}{46} = 56{,}47\%$$

4.2 Pembahasan Hasil Penelitian

Pembahasan hasil penelitian merupakan bahasan hasil dari penelitian yang telah dilakukan terhadap data uji. Penelitihan yang dilakukan yaitu melakukan klasifikasi data komentar pengguna aplikasi ID Express Cutomer yang didapatkan dari *Google Playstore* berupa teks terhadap beberapa metode algoritma. Dimana, metode algoritma yang digunakan yaitu *Naïve Bayes*, *K-Nearest Neighbor*, *Naïve Bayes+PSO*, *dan K-Nearest Neighbor+PSO*. Pada metode-metode tersebut, akan dilakukan uji klasifikasi data yang mana hasilnya dilakukan perbandingan akurasi. Dan dataset yang digunakan diterapkan split validation dimana perbandingan data training dan data testing sebesar 90:10 dan 80:20.

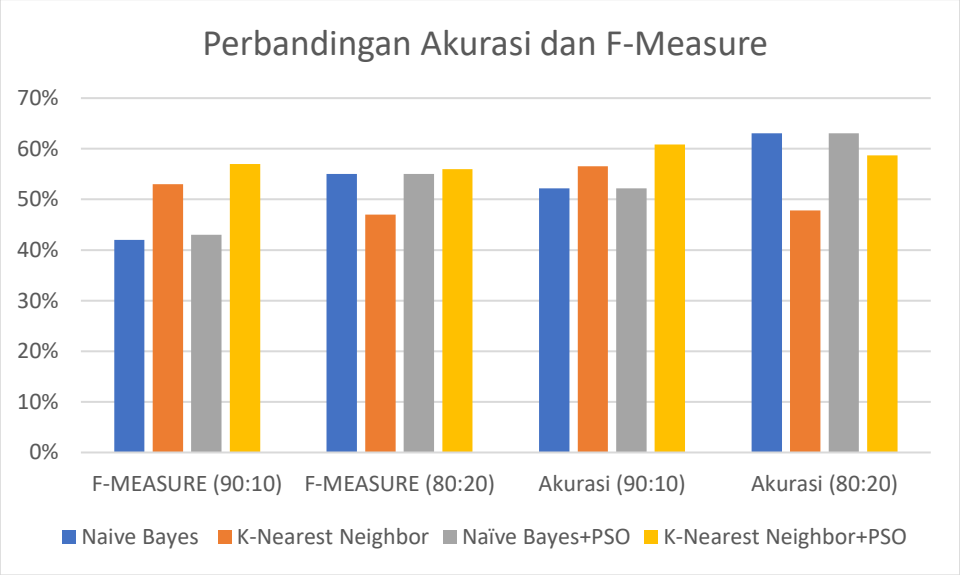
Untuk melihat perbandingan akurasi yang diperolah dari masing-masing metode, dengan penerapan split validation dataset dengan perbandingan data training dan data

testing sebesar 90:10 dan 80:20. Hasil penjelasan dan pemaparannya dapat kita lihat sebagai berikut.

Tabel 4.7 Perbandingan Hasil Metode

No	Metode	Akurasi (90:10)	Akurasi (80:20)	F-measure (90:10)	F-measure (80:20)
1	Naïve Bayes	52,17 %	63,04 %	0,42	0,55
2	K-Nearest Neighbor	56,52 %	47,83 %	0,53	0,47
3	Naïve bayes+PSO	52,17 %	63,04 %	0,43	0,55
4	K-Nearest Neighbor+PSO	60,87 %	58,70 %	0,57	0,56

Pada tabel 4.9, dapat dilihat bahwa dari percobaan menggunakan data training dan testing dengan perbandingan 90:10 dan 80:20 menghasilkan akurasi dan F-measure yang berbeda. Dari penelitian ini dapat disimpulkan bahwa mengambil akurasi yang terbaik yaitu dengan data training 80%, untuk metode *Naïve Bayes, Naïve Bayes+PSO.* Sedangkan, metode *K-Nearest Neighbor* dan *K-Nearest Neighbor+PSO* memiliki akurasi terbaik pada data training 90%. Perbandingan hasil penelitian ini dapat dilihat pada grafik dibawah ini.



Gambar 4.23 Grafik Perbandingan Akurasi dan F-measure

Pada gambar 4.23 dapat dilihat hasil perbandingan akurasi dan F-measure yang diperoleh dari penelitian. Dimana, data akurasi dan F-measure pada rasio 90:10 dan 80:20 digunakan menghasilkan tingkat akurasi dan nilai F-measure yang berbeda pada setiap metode. Pada metode *Naïve Bayes,* menunjukkan akurasi pada rasio 90:10 sebesar 52% dan pada rasi 80:20 lebih besar dengan nilai 63% dan nilai F-measure pasa

rasio 90:10 sebesar 42% dan rasio 80:20 lebih besar dengan nilai 55%. *K-Nearest Neighbor* menghasilkan akurasi 56,52% pada rasio 90:10 dan 47,83% pada 80:20, dengan F-measure pada 90:10 mencapai 53% pada dan rasio 80:20 mencapai 47% lebih kecil. *Naïve Bayes+PSO* memiliki akurasi 52% pada 90:10 dan 63% pada 80:20, dengan F-measure 43% dan 55% masing-masing. Dan metode *K-Nearest Neighbor*+PSO memiliki akurasi lebih tinggi 61% pada rasio 90:10 dari pada 59% pada rasio 80:20, namun F-measure tetap konsisten pada 57% dan 56%. Dalam keseluruhan hasil yang didapat, SVM menonjol dengan kinerja stabil dan baik pada kedua rasio, sedangkan Decision Tree dan KNN memiliki performa yang lebih rendah.