222112058_Feza Raffa Arnanda_Penugasan Praktikum 11

November 2, 2023

Penugasan Praktikum 11

Nama : Feza Raffa Arnadna

NIM : 222112058

Kelas : 3SD2

0.1 A. Penyiapan Library dan Dataset

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# untuk pre-processing teks
import re, string
# bag of words
from sklearn.feature_extraction.text import TfidfVectorizer
#untuk pembangunan model
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, f1_score, accuracy_score,␣
 ↪confusion_matrix
```

Berikutnya, kita bisa menyiapkan data training yang diperlukan. Kali ini kita akan menggunakan dataset analisis sentimen pada tweet (Koto and Rahmaningtyas, 2017). Tugas kita adalah memprediksi apakah suatu tweet memiliki sentimen positif (kode 1) atau negatif (kode 0).

[ ]:
```
# Uncomment baris-baris berikut jika file data training disimpan di komputer
import os
os.chdir('C:/Users/FEZA/My Drive/00. Drive PC/1.STIS/5. Semester 5/Information␣
 ↪Retrieval [IR] P/Pertemuan 10')
df_train=pd.read_csv('train0.csv')
print(df_train.shape)
df_train.head()
```

```
(3638, 2)
```

[ ]:

	sentence	sentiment
0	Kangen NaBil @RealSyahnazS @bangbily RaGa @Raf…	1
1	Doa utk orang yg mberi makan: Ya Allah! Berila…	1

```
2  Setiap kali HP aku bunyi, aku selalu berharap …          1
3  Belum pernah sedekat ini wawancara dgn Afgan S…          1
4  Dulu masa first pergi award show amatlah malas…          1
```

[ ]:
```
df_test=pd.read_csv('test0.csv')
df_test.head()
```

[ ]:
```
                                             sentence  sentiment
0  #Sports Perempuan Golkar Makassar Dibekali Ilm…          1
1  Se-jauh"nya, Se-kenal"nya, Se-pisah"nya, Se-cu…          1
2  Sekedar Shared Ucapan Terimakasih Charles Hono…          1
3  Wah pak Jokowi sudah mendapat nilai positif di…          1
4  Penelpon : raffi ahmad oh raffi ahmad… *bu…          1
```

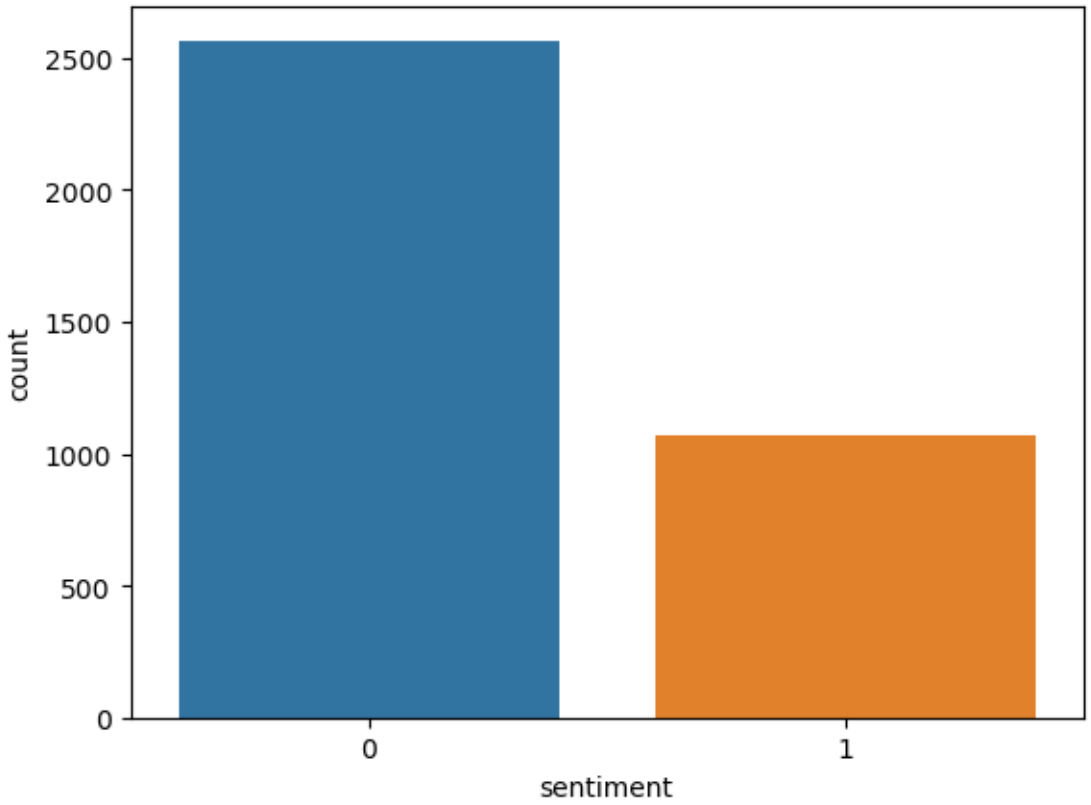
0.2 B. Analisis Data Eksploratif

Class Distribution Data Train

[ ]:
```
# CLASS DISTRIBUTION
# mengecek apakah dataset yang digunakan balance atau tidak
x=df_train['sentiment'].value_counts()
print(x)
sns.barplot(x=x.index, y=x)
```

```
sentiment
0    2567
1    1071
Name: count, dtype: int64
```

[ ]: <Axes: xlabel='sentiment', ylabel='count'>

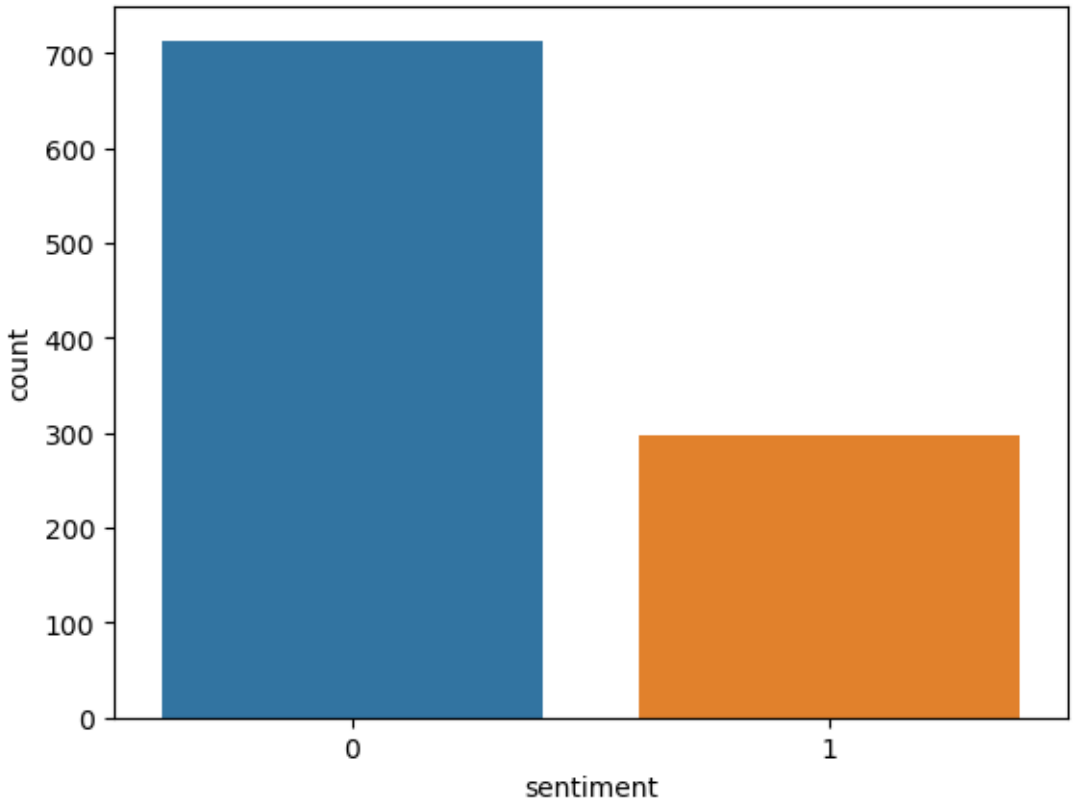


Class Distribution Data Test

[ ]:
```
# CLASS DISTRIBUTION
# mengecek apakah dataset yang digunakan balance atau tidak
x=df_test['sentiment'].value_counts()
print(x)
sns.barplot(x=x.index, y=x)
```

```
sentiment
0    713
1    298
Name: count, dtype: int64
```

[ ]: <Axes: xlabel='sentiment', ylabel='count'>



Cek Missing Values Data Train

[ ]:
```
# Memeriksa missing values
df_train.isna().sum()
```

[ ]:
```
sentence     0
sentiment    0
dtype: int64
```

Cek Missing Values Data Test

[ ]:
```
# Memeriksa missing values
df_test.isna().sum()
```

[ ]:
```
sentence     0
sentiment    0
dtype: int64
```

Word Count, Character Count, dan Unique Word Count Data Train

[ ]:
```
#1. WORD-COUNT
print("Word Count")
```

```
df_train['word_count'] = df_train['sentence'].apply(lambda x:len(str(x).
 ↪split()))
print(df_train[df_train['sentiment']==1]['word_count'].mean()) #Positive
print(df_train[df_train['sentiment']==0]['word_count'].mean()) #Negative
#2. CHARACTER-COUNT
print("\nCharacter Count")
df_train['char_count'] = df_train['sentence'].apply(lambda x: len(str(x)))
print(df_train[df_train['sentiment']==1]['char_count'].mean()) #Positive
print(df_train[df_train['sentiment']==0]['char_count'].mean()) #Negative
#3. UNIQUE WORD-COUNT
print("\nUnique Word Count")
df_train['unique_word_count'] = df_train['sentence'].apply(lambda x:
 ↪len(set(str(x).split())))
print(df_train[df_train['sentiment']==1]['unique_word_count'].mean()) #Positive
print(df_train[df_train['sentiment']==0]['unique_word_count'].mean()) #Negative
```

```
Word Count
16.985060690943044
16.684456564082588

Character Count
121.1484593837535
111.01051811453058

Unique Word Count
16.166199813258636
15.502532138683287
```

Word Count, Character Count, dan Unique Word Count Data Test

[ ]:
```
#1. WORD-COUNT
print("Word Count")
df_test['word_count'] = df_test['sentence'].apply(lambda x:len(str(x).split()))
print(df_test[df_test['sentiment']==1]['word_count'].mean()) #Positive
print(df_test[df_test['sentiment']==0]['word_count'].mean()) #Negative
#2. CHARACTER-COUNT
print("\nCharacter Count")
df_test['char_count'] = df_test['sentence'].apply(lambda x: len(str(x)))
print(df_test[df_test['sentiment']==1]['char_count'].mean()) #Positive
print(df_test[df_test['sentiment']==0]['char_count'].mean()) #Negative
#3. UNIQUE WORD-COUNT
print("\nUnique Word Count")
df_test['unique_word_count'] = df_test['sentence'].apply(lambda x:
 ↪len(set(str(x).split())))
print(df_test[df_test['sentiment']==1]['unique_word_count'].mean()) #Positive
print(df_test[df_test['sentiment']==0]['unique_word_count'].mean()) #Negative
```

```
Word Count
```

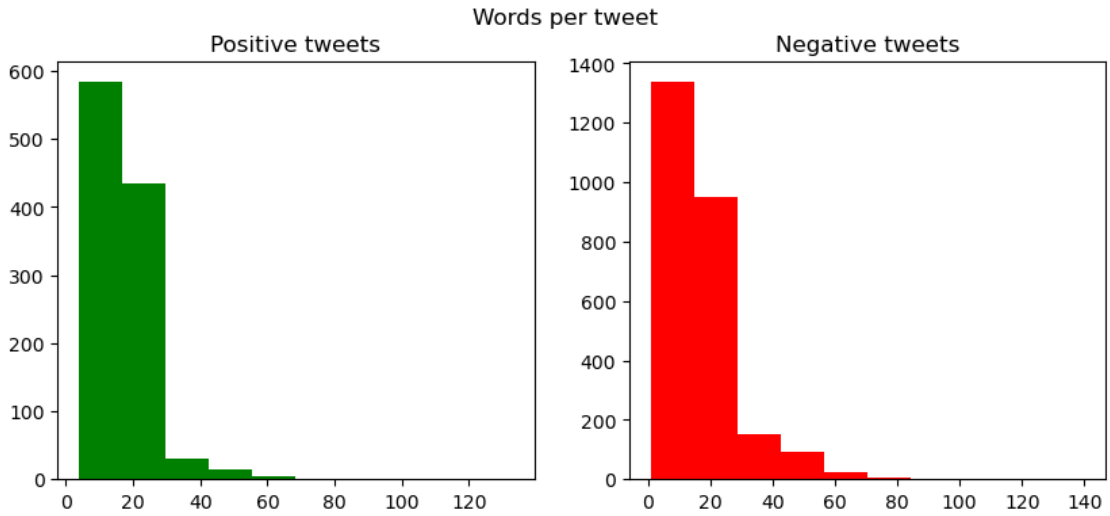
```
17.456375838926174
16.593267882187938

Character Count
124.76510067114094
111.3492286115007

Unique Word Count
16.496644295302012
15.539971949509116
```

Plotting word-count per tweet Data Train

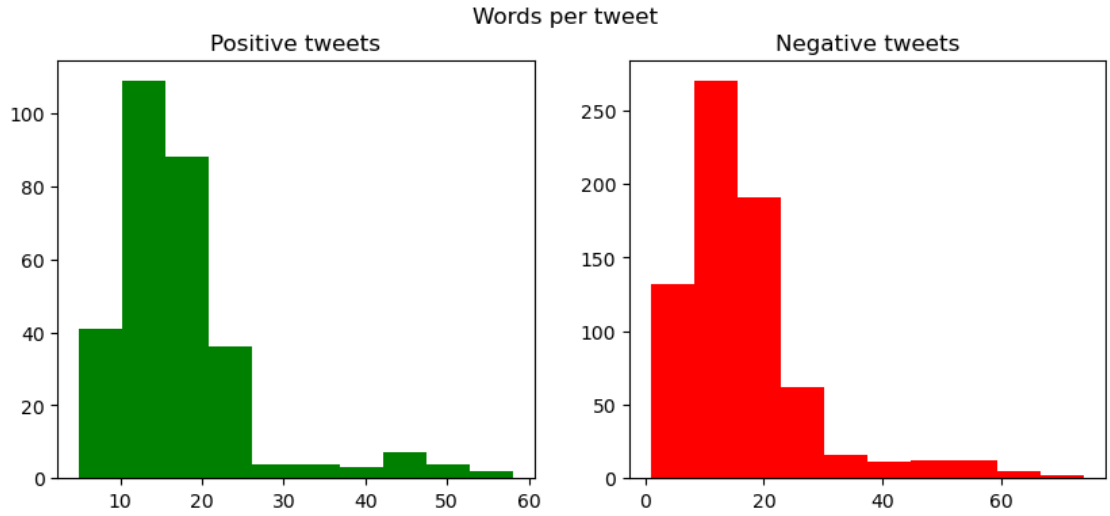
[ ]:
```
# Plotting word-count per tweet
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,4))
train_words=df_train[df_train['sentiment']==1]['word_count']
ax1.hist(train_words,color='green')
ax1.set_title('Positive tweets')
train_words=df_train[df_train['sentiment']==0]['word_count']
ax2.hist(train_words,color='red')
ax2.set_title('Negative tweets')
fig.suptitle('Words per tweet')
plt.show()
df_train=df_train.drop(columns=['word_count','char_count','unique_word_count'])
```



Plotting word-count per tweet Data Test

[ ]:
```
# Plotting word-count per tweet
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,4))
train_words=df_test[df_test['sentiment']==1]['word_count']
```

```
ax1.hist(train_words,color='green')
ax1.set_title('Positive tweets')
train_words=df_test[df_test['sentiment']==0]['word_count']
ax2.hist(train_words,color='red')
ax2.set_title('Negative tweets')
fig.suptitle('Words per tweet')
plt.show()
df_test=df_test.drop(columns=['word_count','char_count','unique_word_count'])
```



0.3 C. Teks Pre-processing

[ ]:
```
# untuk pre-processing teks
#1. Common text preprocessing
text = "@user Teks ini mau dibersihkan. Ada beberapa karakter seperti: <br>, ?,␣
 ↪:, '' spasi berlebih dan tab . "
# mengubah ke huruf kecil (lowercase) dan menghapus tanda baca, karakter aneh␣
 ↪dan strip
def preprocess(text):
    text = text.lower() #lowercase text
    text=text.strip() #Menghapus leading/trailing whitespace
    text = re.sub('@[^\s]+','atUser',text) #mengubah @user menjadi atUser
    text = re.sub(r'#([^\s]+)', r'\1', text) #menghapus hashtag di depan suatu␣
 ↪kata
    text= re.compile('<.*?>').sub('', text) #Menghapus HTML tags/markups
    text = re.compile('[%s]' % re.escape(string.punctuation)).sub(' ', text)
    #Replace punctuation with space. Careful since punctuation can sometime be␣
 ↪useful
    text = re.sub('\s+', ' ', text) #Menghapus extra space dan tabs
```

```
    text = re.sub(r'\[[0-9]*\]',' ',text) #[0-9] matches any digit (0 to10000...
↪)
    text= re.sub(r'[^\w\s]', '', str(text).strip())
    text = re.sub(r'\d',' ',text) #matches any digit from 0 to 100000...,␣
↪\Dmatches non-digits
    text = re.sub(r'\s+',' ',text) #\s matches any whitespace, \s+␣
↪matchesmultiple whitespace, \S matches non-whitespace
    return text


preprocess(text)
```

[ ]:
```
'atUser teks ini mau dibersihkan ada beberapa karakter seperti spasi berlebih
dan tab'
```

[ ]:
```
def tokenisasi(text):
    tokens = text.split(" ")
    return tokens
#STOPWORD ELIMINATION DAN STEMMING
def stemming(text, stemmer):
    # stemming process
    output = stemmer.stem(text)
    return output
def stemming_stopword_elim(text, stopwords, stemmer):
    output = ""
    for token in tokenisasi(text):
        if not token in stopwords:
            output = output + stemming(token, stemmer) + " "
    return output[:-1]
```

[ ]:
```
#FINAL PREPROCESSING
from spacy.lang.id import Indonesian
import spacy
nlp = Indonesian() # use directly
nlp = spacy.blank('id') # blank instance'
stopwords = nlp.Defaults.stop_words
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()
```

[ ]:
```
# def finalpreprocess(string, stopwords, stemmer):
#  return stemming_stopword_elim(preprocess(string), stopwords, stemmer)
```

[ ]:
```
def finalpreprocess(string):
  return preprocess(string)
```

Data Train

```
[ ]: #df_train['clean_text'] = df_train['sentence'].apply(lambda x:␣
      ↪finalpreprocess(x, stopwords, stemmer))

     df_train['clean_text'] = df_train['sentence'].apply(lambda x:finalpreprocess(x))
     df_train.head()
```

```
[ ]:                                          sentence  sentiment  \
     0  Kangen NaBil @RealSyahnazS @bangbily RaGa @Raf…          1
     1  Doa utk orang yg mberi makan: Ya Allah! Berila…          1
     2  Setiap kali HP aku bunyi, aku selalu berharap …          1
     3  Belum pernah sedekat ini wawancara dgn Afgan S…          1
     4  Dulu masa first pergi award show amatlah malas…          1

                                          clean_text
     0  kangen nabil atUser atUser raga atUser atUser …
     1  doa utk orang yg mberi makan ya allah berilah …
     2  setiap kali hp aku bunyi aku selalu berharap i…
     3  belum pernah sedekat ini wawancara dgn afgan s…
     4  dulu masa first pergi award show amatlah malas…
```

Data Test

```
[ ]: #df_test['clean_text'] = df_test['sentence'].apply(lambda x:finalpreprocess(x,␣
      ↪stopwords, stemmer))

     df_test['clean_text'] = df_test['sentence'].apply(lambda x:finalpreprocess(x))
     df_test.head()
```

```
[ ]:                                          sentence  sentiment  \
     0  #Sports Perempuan Golkar Makassar Dibekali Ilm…          1
     1  Se-jauh"nya, Se-kenal"nya, Se-pisah"nya, Se-cu…          1
     2  Sekedar Shared Ucapan Terimakasih Charles Hono…          1
     3  Wah pak Jokowi sudah mendapat nilai positif di…          1
     4  Penelpon : raffi ahmad oh raffi ahmad… *bu…          1

                                          clean_text
     0  sports perempuan golkar makassar dibekali ilmu…
     1  se jauh nya se kenal nya se pisah nya se cuek …
     2  sekedar shared ucapan terimakasih charles hono…
     3  wah pak jokowi sudah mendapat nilai positif di…
     4  penelpon raffi ahmad oh raffi ahmad bukannya s…
```

0.4 D. Ekstraksi Feature dari Data Teks

[ ]:
```
X_train = df_train['clean_text']
y_train = df_train['sentiment']
X_test = df_test['clean_text']
y_test = df_test['sentiment']
# TF-IDF
# Konversi x_train ke vector karena model hanya dapat memproses angka, bukan␣
↪kata/karakter
tfidf_vectorizer = TfidfVectorizer(use_idf=True)
X_train_vectors_tfidf = tfidf_vectorizer.fit_transform(X_train)
# tfidf digunakan pada kalimat yang belum ditokenisasi, berbeda dengan word2vec
# Hanya men-transform x_test (bukan fit dan transform)
X_test_vectors_tfidf = tfidf_vectorizer.transform(X_test)
# Jangan melakukan fungsi fit() TfidfVectorizer ke data testing karena hal itu␣
↪akan
# mengubah indeks kata & bobot sehingga sesuai dengan data testing. Sebaliknya,␣
↪lakukan
# fungsi fit pada data training, lalu gunakan hasil model pada data training␣
↪tadi pada
# data testing untuk menunjukkan fakta bahwa Anda menganalisis data testing␣
↪hanya
# berdasarkan apa yang dipelajari tanpa melihat data testing itu sendiri␣
↪sebelumnya
```

0.5 E. Pembangunan Model Klasifikasi Teks dengan Naive Bayes

[ ]:
```
"""#### NB (tf-idf)"""
nb_tfidf = MultinomialNB()
nb_tfidf.fit(X_train_vectors_tfidf, y_train) #model
#Melakukan prediksi nilai y pada dataset testing
y_predict_nb = nb_tfidf.predict(X_test_vectors_tfidf)
y_prob = nb_tfidf.predict_proba(X_test_vectors_tfidf)[:,1]
```

0.6 F. Evaluasi Model Klasifikasi

[ ]:
```
print("Evaluasi Naive Bayes ",classification_report(y_test,y_predict_nb))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_nb))
```

```
Evaluasi Naive Bayes                precision    recall  f1-score   support

           0       0.79      0.96      0.87       713
           1       0.80      0.39      0.52       298

    accuracy                           0.79      1011
   macro avg       0.80      0.67      0.69      1011
weighted avg       0.79      0.79      0.76      1011
```

```
Confusion Matrix: [[685  28]
 [183 115]]
```

Kesimpulan : 1. *Precision (presisi):* - Precision untuk kelas 0 (Negative): 0.79 - Precision untuk kelas 1 (Positive): 0.77

Precision mengukur sejauh mana model benar-benar memprediksi kelas tertentu dengan benar. Dalam hal ini, untuk kelas 0 (Negatif), model benar-benar memprediksi dengan benar sekitar 79% dari semua prediksi yang dilakukan untuk kelas tersebut. Sedangkan, untuk kelas 1 (Positif), model benar-benar memprediksi dengan benar sekitar 77% dari semua prediksi yang dilakukan untuk kelas tersebut.

2. *Recall (recall):*

   - Recall untuk kelas 0 (Negative): 0.95
   - Recall untuk kelas 1 (Positive): 0.41

   Recall mengukur sejauh mana model dapat mendeteksi semua instance yang benar-benar termasuk dalam kelas tertentu. Dalam hal ini, untuk kelas 0 (Negatif), model dapat mendeteksi sekitar 95% dari semua instance yang seharusnya masuk ke dalam kelas tersebut. Namun, untuk kelas 1 (Positif), model hanya dapat mendeteksi sekitar 41% dari semua instance yang seharusnya masuk ke dalam kelas tersebut.

3. *F1-Score (f1-score):*

   - F1-Score untuk kelas 0 (Negative): 0.86
   - F1-Score untuk kelas 1 (Positive): 0.54

   F1-Score adalah perpaduan antara precision dan recall, yang berguna untuk mengukur keseluruhan kinerja model. Untuk kelas 0 (Negatif), F1-Score adalah 0.86, dan untuk kelas 1 (Positif), F1-Score adalah 0.54. F1-Score yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara precision dan recall.

4. *Akurasi (accuracy):*

   - Akurasi secara keseluruhan: 0.79

   Akurasi adalah rasio prediksi yang benar secara keseluruhan. Dalam hal ini, akurasi adalah sekitar 79%, yang berarti model ini benar sekitar 79% dari waktu saat memprediksi kelas dari keseluruhan data. 79% persen yang diprediksi benar dari keseluruhan.

5. *Macro Average dan Weighted Average:*

   - Macro Average mengambil rata-rata metrik untuk setiap kelas tanpa memperhatikan distribusi kelas. Macro Average F1-Score adalah 0.70.
   - Weighted Average memberi bobot metrik berdasarkan jumlah sampel dalam setiap kelas. Weighted Average F1-Score adalah 0.77.

   Macro Average berguna jika kelas-kelas memiliki distribusi yang seimbang, sementara Weighted Average lebih akurat ketika kelas-kelas memiliki distribusi yang tidak seimbang. Weighted Average lebih tinggi karena memberikan lebih banyak bobot pada kelas mayoritas (kelas 0) yang memiliki lebih banyak sampel.

Secara keseluruhan, model memiliki kinerja yang baik dalam memprediksi kelas 0 (Negatif), tetapi memiliki kinerja yang lebih rendah dalam memprediksi kelas 1 (Positif) berdasarkan nilai recall dan

F1-Score yang lebih rendah. Akurasi model adalah 0.79, yang berarti model ini benar sekitar 79%.

Precisison : 79% tweet yang bersentimen negatif itu benar dari semua tweet yang diprediksi bersentimen negatif

Recall : 96% tweet yang benar bersentimen negatif itu benar dari semua tweet yang aktual/sebenarnya negatif

Precisison : 80% tweet yang bersentimen positif itu benar dari semua tweet yang diprediksi bersentimen positif

Recall : 39% tweet yang benar bersentimen positif itu benar dari semua tweet yang aktual/sebenarnya positif

1 Praktikum Pertemuan 11

1.1 A. Pembangunan Model Klasifikasi Teks dengan Rocchio Classification

[ ]:
```
from sklearn.neighbors import NearestCentroid
rocchio_tfidf = NearestCentroid()
rocchio_tfidf.fit(X_train_vectors_tfidf, y_train) #model
#Melakukan prediksi nilai y pada dataset testing
y_predict_rocchio = rocchio_tfidf.predict(X_test_vectors_tfidf)
```

[ ]:
```
# gatau bener apa gak wkwkw

print("Evaluasi Rocchio\n",classification_report(y_test,y_predict_rocchio))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_rocchio))
```

```
Evaluasi Rocchio
              precision    recall  f1-score   support

           0       0.89      0.58      0.70       713
           1       0.45      0.84      0.59       298

    accuracy                           0.66      1011
   macro avg       0.67      0.71      0.65      1011
weighted avg       0.76      0.66      0.67      1011

Confusion Matrix: [[414 299]
 [ 49 249]]
```

1.2 B. Pembangunan Model Klasifikasi Teks dengan kNN

[ ]:
```
from sklearn.neighbors import KNeighborsClassifier
n_neighbors=5
knn_tfidf = KNeighborsClassifier(n_neighbors, weights='distance')
knn_tfidf.fit(X_train_vectors_tfidf, y_train) #model
#Melakukan prediksi nilai y pada dataset
testingy_predict = knn_tfidf.predict(X_test_vectors_tfidf)
```

```
y_predict_kNN = testingy_predict
```

[ ]:
```
# gatau bener apa gak wkwkw

print("Evaluasi kNN\n",classification_report(y_test,y_predict_kNN))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_kNN))
```

```
Evaluasi kNN
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       713
           1       0.73      0.66      0.69       298

    accuracy                           0.83      1011
   macro avg       0.79      0.78      0.78      1011
weighted avg       0.82      0.83      0.82      1011

Confusion Matrix: [[639  74]
 [102 196]]
```

1.3 C. Pembangunan Model Klasifikasi Teks dengan SVM

[ ]:
```
from sklearn import svm
svm_tfidf = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
svm_tfidf.fit(X_train_vectors_tfidf, y_train) #model
#Melakukan prediksi nilai y pada dataset testing
y_predict_svm = svm_tfidf.predict(X_test_vectors_tfidf)
```

[ ]:
```
print("Evaluasi SVM\n",classification_report(y_test,y_predict_svm))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_svm))
```

```
Evaluasi SVM
              precision    recall  f1-score   support

           0       0.88      0.89      0.89       713
           1       0.74      0.70      0.72       298

    accuracy                           0.84      1011
   macro avg       0.81      0.80      0.80      1011
weighted avg       0.84      0.84      0.84      1011

Confusion Matrix: [[638  75]
 [ 89 209]]
```

1.4 D. Mencari Parameter Terbaik dengan Grid Search

C besar : seakin banyak support vector yang digunakan untuk membuat hyperplan

Gamma besar, semakin tinggi bias, dan rendah variance

[ ]:
```
from sklearn.model_selection import GridSearchCV
# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['linear', 'rbf']}
grid = GridSearchCV(svm.SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train_vectors_tfidf, y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[CV 1/5] END …C=0.1, gamma=1, kernel=linear;, score=0.720 total time=   1.1s
[CV 2/5] END …C=0.1, gamma=1, kernel=linear;, score=0.717 total time=   2.0s
[CV 3/5] END …C=0.1, gamma=1, kernel=linear;, score=0.717 total time=   1.9s
[CV 4/5] END …C=0.1, gamma=1, kernel=linear;, score=0.713 total time=   1.9s
[CV 5/5] END …C=0.1, gamma=1, kernel=linear;, score=0.719 total time=   1.8s
[CV 1/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.705 total time=   2.5s
[CV 2/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.706 total time=   2.3s
[CV 3/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.706 total time=   2.3s
[CV 4/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.706 total time=   2.0s
[CV 5/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.706 total time=   2.0s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.720 total time=   1.7s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.717 total time=   2.0s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.717 total time=   2.0s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.713 total time=   2.0s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.719 total time=   1.9s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.705 total time=   1.9s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.706 total time=   2.4s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.706 total time=   2.6s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.706 total time=   1.9s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.706 total time=   1.3s
[CV 1/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.720 total time=   1.5s
[CV 2/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.717 total time=   1.6s
[CV 3/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.717 total time=   1.7s
[CV 4/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.713 total time=   1.6s
[CV 5/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.719 total time=   1.5s
[CV 1/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.705 total time=   2.5s
[CV 2/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.706 total time=   1.6s
[CV 3/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.706 total time=   1.8s
[CV 4/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.706 total time=   2.0s
[CV 5/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.706 total time=   1.3s
[CV 1/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.720 total time=   1.3s
[CV 2/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.717 total time=   1.9s
[CV 3/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.717 total time=   2.3s
[CV 4/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.713 total time=   1.5s
[CV 5/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.719 total time=   1.7s
[CV 1/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.705 total time=   1.3s
[CV 2/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.1s
```

```
[CV 3/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 4/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 5/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 1/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.720 total time=   1.4s
[CV 2/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.717 total time=   1.6s
[CV 3/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.717 total time=   2.0s
[CV 4/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.713 total time=   2.0s
[CV 5/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.719 total time=   3.0s
[CV 1/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.705 total time=   1.6s
[CV 2/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.5s
[CV 3/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.9s
[CV 4/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.5s
[CV 5/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.3s
[CV 1/5] END …C=1, gamma=1, kernel=linear;, score=0.846 total time=   2.0s
[CV 2/5] END …C=1, gamma=1, kernel=linear;, score=0.837 total time=   2.3s
[CV 3/5] END …C=1, gamma=1, kernel=linear;, score=0.850 total time=   2.3s
[CV 4/5] END …C=1, gamma=1, kernel=linear;, score=0.839 total time=   2.6s
[CV 5/5] END …C=1, gamma=1, kernel=linear;, score=0.856 total time=   1.5s
[CV 1/5] END …C=1, gamma=1, kernel=rbf;, score=0.815 total time=   3.3s
[CV 2/5] END …C=1, gamma=1, kernel=rbf;, score=0.821 total time=   2.5s
[CV 3/5] END …C=1, gamma=1, kernel=rbf;, score=0.828 total time=   2.0s
[CV 4/5] END …C=1, gamma=1, kernel=rbf;, score=0.818 total time=   2.8s
[CV 5/5] END …C=1, gamma=1, kernel=rbf;, score=0.828 total time=   2.6s
[CV 1/5] END …C=1, gamma=0.1, kernel=linear;, score=0.846 total time=   1.4s
[CV 2/5] END …C=1, gamma=0.1, kernel=linear;, score=0.837 total time=   1.3s
[CV 3/5] END …C=1, gamma=0.1, kernel=linear;, score=0.850 total time=   1.5s
[CV 4/5] END …C=1, gamma=0.1, kernel=linear;, score=0.839 total time=   1.2s
[CV 5/5] END …C=1, gamma=0.1, kernel=linear;, score=0.856 total time=   1.2s
[CV 1/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.751 total time=   1.3s
[CV 2/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.769 total time=   1.3s
[CV 3/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.772 total time=   1.4s
[CV 4/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.733 total time=   1.3s
[CV 5/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.761 total time=   1.7s
[CV 1/5] END …C=1, gamma=0.01, kernel=linear;, score=0.846 total time=   1.9s
[CV 2/5] END …C=1, gamma=0.01, kernel=linear;, score=0.837 total time=   2.3s
[CV 3/5] END …C=1, gamma=0.01, kernel=linear;, score=0.850 total time=   2.3s
[CV 4/5] END …C=1, gamma=0.01, kernel=linear;, score=0.839 total time=   2.0s
[CV 5/5] END …C=1, gamma=0.01, kernel=linear;, score=0.856 total time=   2.2s
[CV 1/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.705 total time=   2.1s
[CV 2/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.706 total time=   1.8s
[CV 3/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.706 total time=   1.7s
[CV 4/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.706 total time=   1.6s
[CV 5/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.706 total time=   2.0s
[CV 1/5] END …C=1, gamma=0.001, kernel=linear;, score=0.846 total time=   2.3s
[CV 2/5] END …C=1, gamma=0.001, kernel=linear;, score=0.837 total time=   1.7s
[CV 3/5] END …C=1, gamma=0.001, kernel=linear;, score=0.850 total time=   1.4s
[CV 4/5] END …C=1, gamma=0.001, kernel=linear;, score=0.839 total time=   1.4s
[CV 5/5] END …C=1, gamma=0.001, kernel=linear;, score=0.856 total time=   1.3s
```

```
[CV 1/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.705 total time=   1.5s
[CV 2/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.6s
[CV 3/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.8s
[CV 4/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.9s
[CV 5/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.4s
[CV 1/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.846 total time=   1.3s
[CV 2/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.837 total time=   1.5s
[CV 3/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.850 total time=   1.3s
[CV 4/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.839 total time=   1.4s
[CV 5/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.856 total time=   1.4s
[CV 1/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.705 total time=   1.3s
[CV 2/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 3/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.0s
[CV 4/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.0s
[CV 5/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.0s
[CV 1/5] END …C=10, gamma=1, kernel=linear;, score=0.834 total time=   1.4s
[CV 2/5] END …C=10, gamma=1, kernel=linear;, score=0.835 total time=   1.4s
[CV 3/5] END …C=10, gamma=1, kernel=linear;, score=0.839 total time=   1.4s
[CV 4/5] END …C=10, gamma=1, kernel=linear;, score=0.850 total time=   1.4s
[CV 5/5] END …C=10, gamma=1, kernel=linear;, score=0.839 total time=   1.4s
[CV 1/5] END …C=10, gamma=1, kernel=rbf;, score=0.839 total time=   1.7s
[CV 2/5] END …C=10, gamma=1, kernel=rbf;, score=0.838 total time=   1.9s
[CV 3/5] END …C=10, gamma=1, kernel=rbf;, score=0.841 total time=   1.7s
[CV 4/5] END …C=10, gamma=1, kernel=rbf;, score=0.835 total time=   1.7s
[CV 5/5] END …C=10, gamma=1, kernel=rbf;, score=0.853 total time=   2.5s
[CV 1/5] END …C=10, gamma=0.1, kernel=linear;, score=0.834 total time=   2.5s
[CV 2/5] END …C=10, gamma=0.1, kernel=linear;, score=0.835 total time=   1.8s
[CV 3/5] END …C=10, gamma=0.1, kernel=linear;, score=0.839 total time=   2.1s
[CV 4/5] END …C=10, gamma=0.1, kernel=linear;, score=0.850 total time=   1.7s
[CV 5/5] END …C=10, gamma=0.1, kernel=linear;, score=0.839 total time=   1.6s
[CV 1/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.849 total time=   2.2s
[CV 2/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.842 total time=   1.7s
[CV 3/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.848 total time=   1.6s
[CV 4/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.856 total time=   1.9s
[CV 5/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.868 total time=   2.1s
[CV 1/5] END …C=10, gamma=0.01, kernel=linear;, score=0.834 total time=   2.1s
[CV 2/5] END …C=10, gamma=0.01, kernel=linear;, score=0.835 total time=   2.3s
[CV 3/5] END …C=10, gamma=0.01, kernel=linear;, score=0.839 total time=   1.9s
[CV 4/5] END …C=10, gamma=0.01, kernel=linear;, score=0.850 total time=   1.9s
[CV 5/5] END …C=10, gamma=0.01, kernel=linear;, score=0.839 total time=   1.4s
[CV 1/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.760 total time=   1.1s
[CV 2/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.782 total time=   1.9s
[CV 3/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.777 total time=   1.8s
[CV 4/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.751 total time=   2.5s
[CV 5/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.769 total time=   3.1s
[CV 1/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.834 total time=   2.2s
[CV 2/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.835 total time=   1.6s
[CV 3/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.839 total time=   1.5s
```

```
[CV 4/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.850 total time=   1.5s
[CV 5/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.839 total time=   1.4s
[CV 1/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.705 total time=   1.1s
[CV 2/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 3/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.0s
[CV 4/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.3s
[CV 5/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.706 total time=   1.2s
[CV 1/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.834 total time=   1.2s
[CV 2/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.835 total time=   1.3s
[CV 3/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.839 total time=   1.3s
[CV 4/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.850 total time=   1.2s
[CV 5/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.839 total time=   1.2s
[CV 1/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.705 total time=   1.0s
[CV 2/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.6s
[CV 3/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 4/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.2s
[CV 5/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.2s
[CV 1/5] END …C=100, gamma=1, kernel=linear;, score=0.837 total time=   1.9s
[CV 2/5] END …C=100, gamma=1, kernel=linear;, score=0.838 total time=   1.8s
[CV 3/5] END …C=100, gamma=1, kernel=linear;, score=0.837 total time=   1.4s
[CV 4/5] END …C=100, gamma=1, kernel=linear;, score=0.847 total time=   1.5s
[CV 5/5] END …C=100, gamma=1, kernel=linear;, score=0.831 total time=   1.4s
[CV 1/5] END …C=100, gamma=1, kernel=rbf;, score=0.839 total time=   2.1s
[CV 2/5] END …C=100, gamma=1, kernel=rbf;, score=0.838 total time=   1.5s
[CV 3/5] END …C=100, gamma=1, kernel=rbf;, score=0.841 total time=   1.6s
[CV 4/5] END …C=100, gamma=1, kernel=rbf;, score=0.835 total time=   1.5s
[CV 5/5] END …C=100, gamma=1, kernel=rbf;, score=0.853 total time=   1.6s
[CV 1/5] END …C=100, gamma=0.1, kernel=linear;, score=0.837 total time=   1.9s
[CV 2/5] END …C=100, gamma=0.1, kernel=linear;, score=0.838 total time=   1.4s
[CV 3/5] END …C=100, gamma=0.1, kernel=linear;, score=0.837 total time=   1.8s
[CV 4/5] END …C=100, gamma=0.1, kernel=linear;, score=0.847 total time=   1.5s
[CV 5/5] END …C=100, gamma=0.1, kernel=linear;, score=0.831 total time=   1.8s
[CV 1/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.839 total time=   1.8s
[CV 2/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.839 total time=   2.2s
[CV 3/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.852 total time=   2.1s
[CV 4/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.856 total time=   2.4s
[CV 5/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.849 total time=   2.9s
[CV 1/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.837 total time=   2.1s
[CV 2/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.838 total time=   2.1s
[CV 3/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.837 total time=   1.9s
[CV 4/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.847 total time=   1.8s
[CV 5/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.831 total time=   2.0s
[CV 1/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.845 total time=   1.8s
[CV 2/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.842 total time=   1.9s
[CV 3/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.848 total time=   1.9s
[CV 4/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.856 total time=   2.0s
[CV 5/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.867 total time=   1.8s
[CV 1/5] END .C=100, gamma=0.001, kernel=linear;, score=0.837 total time=   2.3s
```

```
[CV 2/5] END .C=100, gamma=0.001, kernel=linear;, score=0.838 total time=   1.6s
[CV 3/5] END .C=100, gamma=0.001, kernel=linear;, score=0.837 total time=   1.5s
[CV 4/5] END .C=100, gamma=0.001, kernel=linear;, score=0.847 total time=   1.4s
[CV 5/5] END .C=100, gamma=0.001, kernel=linear;, score=0.831 total time=   1.5s
[CV 1/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.764 total time=   1.1s
[CV 2/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.782 total time=   1.2s
[CV 3/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.780 total time=   1.1s
[CV 4/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.751 total time=   1.3s
[CV 5/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.769 total time=   1.6s
[CV 1/5] END C=100, gamma=0.0001, kernel=linear;, score=0.837 total time=   1.3s
[CV 2/5] END C=100, gamma=0.0001, kernel=linear;, score=0.838 total time=   1.5s
[CV 3/5] END C=100, gamma=0.0001, kernel=linear;, score=0.837 total time=   1.9s
[CV 4/5] END C=100, gamma=0.0001, kernel=linear;, score=0.847 total time=   1.9s
[CV 5/5] END C=100, gamma=0.0001, kernel=linear;, score=0.831 total time=   1.3s
[CV 1/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.705 total time=   1.4s
[CV 2/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 3/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 4/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.0s
[CV 5/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.706 total time=   1.1s
[CV 1/5] END …C=1000, gamma=1, kernel=linear;, score=0.837 total time=   1.4s
[CV 2/5] END …C=1000, gamma=1, kernel=linear;, score=0.838 total time=   1.2s
[CV 3/5] END …C=1000, gamma=1, kernel=linear;, score=0.837 total time=   1.2s
[CV 4/5] END …C=1000, gamma=1, kernel=linear;, score=0.847 total time=   1.2s
[CV 5/5] END …C=1000, gamma=1, kernel=linear;, score=0.831 total time=   1.2s
[CV 1/5] END …C=1000, gamma=1, kernel=rbf;, score=0.839 total time=   1.5s
[CV 2/5] END …C=1000, gamma=1, kernel=rbf;, score=0.838 total time=   1.5s
[CV 3/5] END …C=1000, gamma=1, kernel=rbf;, score=0.841 total time=   1.5s
[CV 4/5] END …C=1000, gamma=1, kernel=rbf;, score=0.835 total time=   1.5s
[CV 5/5] END …C=1000, gamma=1, kernel=rbf;, score=0.853 total time=   1.5s
[CV 1/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.837 total time=   1.2s
[CV 2/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.838 total time=   1.2s
[CV 3/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.837 total time=   1.2s
[CV 4/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.847 total time=   1.2s
[CV 5/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.831 total time=   1.2s
[CV 1/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.839 total time=   1.3s
[CV 2/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.843 total time=   1.3s
[CV 3/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.849 total time=   1.3s
[CV 4/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.856 total time=   1.3s
[CV 5/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.847 total time=   1.4s
[CV 1/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.837 total time=   1.2s
[CV 2/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.838 total time=   1.3s
[CV 3/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.837 total time=   1.3s
[CV 4/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.847 total time=   1.3s
[CV 5/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.831 total time=   1.3s
[CV 1/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.835 total time=   2.6s
[CV 2/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.841 total time=   2.4s
[CV 3/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.837 total time=   3.2s
[CV 4/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.849 total time=   3.5s
```

```
[CV 5/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.828 total time=   2.0s
[CV 1/5] END C=1000, gamma=0.001, kernel=linear;, score=0.837 total time=   1.5s
[CV 2/5] END C=1000, gamma=0.001, kernel=linear;, score=0.838 total time=   1.8s
[CV 3/5] END C=1000, gamma=0.001, kernel=linear;, score=0.837 total time=   1.7s
[CV 4/5] END C=1000, gamma=0.001, kernel=linear;, score=0.847 total time=   1.9s
[CV 5/5] END C=1000, gamma=0.001, kernel=linear;, score=0.831 total time=   1.5s
[CV 1/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.845 total time=   1.1s
[CV 2/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.842 total time=   1.3s
[CV 3/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.848 total time=   1.3s
[CV 4/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.856 total time=   1.3s
[CV 5/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.867 total time=   1.3s
[CV 1/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.837 total time=
1.2s
[CV 2/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.838 total time=
1.3s
[CV 3/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.837 total time=
1.3s
[CV 4/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.847 total time=
1.3s
[CV 5/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.831 total time=
1.4s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.764 total time=   1.1s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.782 total time=   1.1s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.780 total time=   1.1s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.751 total time=   1.1s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.769 total time=   1.1s
```

[ ]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['linear', 'rbf']},
             verbose=3)
```

Parameter terbaik hasil percobaan dengan Grid Search didapatkan dengan:

[ ]:
```
# print best parameter after tuning
print(grid.best_params_)
# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
SVC(C=10, gamma=0.1)
```

[ ]:
```
from sklearn import svm
svm_tfidf = svm.SVC(C=0.1, kernel='rbf', degree=3, gamma='auto')
svm_tfidf.fit(X_train_vectors_tfidf, y_train) #model
#Melakukan prediksi nilai y pada dataset testing
y_predict_svm = svm_tfidf.predict(X_test_vectors_tfidf)
```

Selanjutnya, prediksi menggunakan parameter terbaik dapat dilakukan dengan kode berikut

[ ]:
```
y_grid_predictions = grid.predict(X_test_vectors_tfidf)
```

[ ]:
```
print("Evaluasi Grid SVM\n",classification_report(y_test,y_grid_predictions))
print('Confusion Matrix:',confusion_matrix(y_test, y_grid_predictions))
```

```
Evaluasi Grid SVM
              precision    recall  f1-score   support

           0       0.89      0.88      0.88       713
           1       0.71      0.74      0.73       298

    accuracy                           0.84      1011
   macro avg       0.80      0.81      0.80      1011
weighted avg       0.84      0.84      0.84      1011

Confusion Matrix: [[624  89]
 [ 77 221]]
```

1.5 E. Menggunakan k-Fold Cross Validation

Karena dataset yang Anda gunakan sebelumnya telah dibagi menjadi data training dan testing, maka gabungkan terlebih dahulu data tersebut dengan melakukan concat, baru kemudian melakukan cross validation terhadap keseluruhan data.

[ ]:
```
X_join = pd.concat([X_train, X_test])
y_join = pd.concat([y_train, y_test])
X_join_vectors_tfidf = tfidf_vectorizer.transform(X_join)
```

Berikut kode yang digunakan untuk melakukan 5-fold cross validation, misalnya untuk model Naive Bayes

[ ]:
```
from sklearn.model_selection import cross_val_score,␣
 ↪cross_val_predict,cross_validate
scores = cross_validate(nb_tfidf, X_join_vectors_tfidf, y_join, cv=5,␣
 ↪scoring=('accuracy', 'f1'), return_train_score=True)
predictions = cross_val_predict(nb_tfidf, X_join_vectors_tfidf, y_join, cv=5)

scores
```

[ ]:
```
{'fit_time': array([0.00499368, 0.00300121, 0.008564  , 0.00719142,
0.00956917]),
 'score_time': array([0.00905299, 0.01253605, 0.0030117 , 0.00299191,
0.00404882]),
 'test_accuracy': array([0.81290323, 0.80860215, 0.82150538, 0.81397849,
0.79440258]),
 'train_accuracy': array([0.87442861, 0.87335305, 0.87415972, 0.87684862,
0.88602151]),
```

```
 'test_f1': array([0.57971014, 0.57819905, 0.59708738, 0.57907543, 0.52605459]),
 'train_f1': array([0.73749297, 0.73583847, 0.73589165, 0.74065685,
0.76779847])}
```

1.6 Kompilasi Evaluation Matrix

[ ]:
```
print("Naive Bayes")
print(classification_report(y_test,y_predict_nb))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_nb))
print('\n')


print("Rocchio Classification")
print(classification_report(y_test,y_predict_rocchio))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_rocchio))
print('\n')


print("kNN")
print(classification_report(y_test,y_predict_kNN))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_kNN))
print('\n')


print("SVM")
print(classification_report(y_test,y_predict_svm))
print('Confusion Matrix:',confusion_matrix(y_test, y_predict_svm))
print('\n')

print("SVM Parameter dengan Grid Search")
print(classification_report(y_test,y_grid_predictions))
print('Confusion Matrix:',confusion_matrix(y_test, y_grid_predictions))
```

```
Naive Bayes
              precision    recall  f1-score   support

           0       0.79      0.96      0.87       713
           1       0.80      0.39      0.52       298

    accuracy                           0.79      1011
   macro avg       0.80      0.67      0.69      1011
weighted avg       0.79      0.79      0.76      1011

Confusion Matrix: [[685  28]
 [183 115]]


Rocchio Classification
```

```
              precision    recall  f1-score   support

           0       0.89      0.58      0.70       713
           1       0.45      0.84      0.59       298

    accuracy                           0.66      1011
   macro avg       0.67      0.71      0.65      1011
weighted avg       0.76      0.66      0.67      1011

Confusion Matrix: [[414 299]
 [ 49 249]]


kNN
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       713
           1       0.73      0.66      0.69       298

    accuracy                           0.83      1011
   macro avg       0.79      0.78      0.78      1011
weighted avg       0.82      0.83      0.82      1011

Confusion Matrix: [[639  74]
 [102 196]]


SVM
              precision    recall  f1-score   support

           0       0.88      0.89      0.89       713
           1       0.74      0.70      0.72       298

    accuracy                           0.84      1011
   macro avg       0.81      0.80      0.80      1011
weighted avg       0.84      0.84      0.84      1011

Confusion Matrix: [[638  75]
 [ 89 209]]


SVM Parameter dengan Grid Search
              precision    recall  f1-score   support

           0       0.89      0.88      0.88       713
           1       0.71      0.74      0.73       298

    accuracy                           0.84      1011
```

```
   macro avg       0.80      0.81      0.80      1011
weighted avg       0.84      0.84      0.84      1011

Confusion Matrix: [[624  89]
 [ 77 221]]
```

Berdasarkan hasil evaluasi klasifikasi teks dari Naive Bayes, Rocchio Classification, kNN, dan SVM, apabila dilihat dari nilai akurasinya (model klasifikasi tersebut mampu memprediksi kelas dari data teks dengan benar), nilai dengan akurasi tertinggi hingga terendah yaitu : 1. SVM dan SVM dengan Grid Search 2. kNN 3. Naive Bayes 4. Rocchio

Metode yang cocok di gunakan apabila untuk mengklasifikasikan ke dalam kelompok sentimen negatif metode yang cocok digunakan apabila dilihat dari f1-score adalah metode svm dengan tingkat kecocokanberdasarka f1-score sebesar 89%.

Sedangkan metode yang cocok di gunakan untuk mengklasifikasikan ke dalam kelompok sentimen positif adalah metode svm dengan grid search dengan tingkat kecocokan berdasarkan f1-score sebesar 73% atau svm dengan tingkat kecocokan berdasarkan f1-score sebesar 72%