BAB 4

HASIL PENELITIAN

4.1 IMPLEMENTASI SISTEM

4.1.1 Data Training

Proses pertama pada saat pengujian data training, perlu melakukan import library pandas dan openpyxl untuk membaca data. Adapun proses melakukan install pandas kemudian membaca data seperti gambar 4.1 berikut :

```
# memasang library pandas
import pandas as pd
```

```
# pip install openpyxl
```

```
#membaca data

df = pd.read_excel (r'pelabelan_manual.xlsx', sheet_name='Sheet1')
df=pd.DataFrame(df)
df
```

	no	tanggal	username	tweet	label	kelas
0	1	2012-03-07 17:32:43+00:00	agentaruhanbola	account transfer main bola situs taruh bola li...	Positif	1
1	2	2010-06-20 09:15:32+00:00	baskaramp	aduh aduh kangen liga inggris spanyol itali ni...	Positif	1
2	3	2021-04-15 16:30:00+00:00	Bolanet	akhir spekulasi barcelona segera umum transfer...	Negatif	0
3	4	2021-04-15 10:28:14+00:00	M88Indo	akhir spekulasi barcelona segera umum transfer...	Negatif	0
4	5	2020-08-30 15:51:35+00:00	_fireshare	aku masuk opini kalau macam insentif bisnis sp...	Negatif	0
...	...	...	...	...	...	...
595	596	2021-04-19 02:10:14+00:00	anggaimaginer	wabilkhusus barcelona bikin hutang sebenernya ...	Negatif	0
596	597	2021-08-25 11:23:58+00:00	minggussenin	waduuuhh bahaya nih transfer main atlet musim ...	Negatif	0
597	598	2012-09-01 14:35:00+00:00	febriantopw	weseet edan tenan psg i moso keluar transfer m...	Negatif	0
598	599	2011-06-30 16:04:19+00:00	maudyannisa	yeye tidak masuk d spesial transfer main binta...	Negatif	0
599	600	2020-09-26 12:20:01+00:00	bolaskorcom	zinedine zidane nanti kejut transfer real madrid	Negatif	0

600 rows × 6 columns

Gambar 4.1 import library pandas dan membaca data

Kemudian setelah dilakukan pemanggilan data, dilakukan pengujian TF-IDF pada dokumen. Adapun kode pengujian data TF-IDF seperti gambar 4.2 berikut :

```
from sklearn.feature_extraction.text import TfidfVectorizer

d1 = "real madrid lineup ganti raphael varane tengah minat transfer manchester united"
d2 = "real madrid tumbal vinicius junior transfer kylian mbappe"
d3 = "update transfer main sergio aguero sepakat gabung barcelona kontrak tahun depan"
d4 = "rumor transfer sergio aguero sepakat gabung barcelona musim depan"

vect = TfidfVectorizer()
X = vect.fit_transform([d1, d2, d3, d4])

X.toarray()
```

```
array([[0.        , 0.        , 0.        , 0.        , 0.32417842,
        0.        , 0.        , 0.        , 0.32417842, 0.25558599,
        0.        , 0.32417842, 0.        , 0.32417842, 0.        ,
        0.32417842, 0.25558599, 0.        , 0.        , 0.        ,
        0.        , 0.32417842, 0.16916975, 0.        , 0.32417842,
        0.        , 0.32417842, 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.39176533, 0.        , 0.39176533, 0.        , 0.30887228,
        0.        , 0.        , 0.39176533, 0.        , 0.        ,
        0.        , 0.30887228, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.2044394 , 0.39176533, 0.        ,
        0.        , 0.        , 0.39176533],
       [0.2787129 , 0.2787129 , 0.2787129 , 0.2787129 , 0.        ,
        0.        , 0.35351198, 0.        , 0.        , 0.        ,
        0.35351198, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.2787129 , 0.2787129 ,
        0.35351198, 0.        , 0.18447721, 0.        , 0.        ,
        0.35351198, 0.        , 0.        ],
       [0.32181737, 0.32181737, 0.32181737, 0.32181737, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.40818453,
        0.        , 0.        , 0.40818453, 0.32181737, 0.32181737,
        0.        , 0.        , 0.21300762, 0.        , 0.        ,
        0.        , 0.        , 0.        ]])
```

Gambar 4.2 TF-IDF Data Training

Setelah dilakukan proses perhitungan TF-IDF, maka selanjutnya proses pembagian data training dan data testing. Adapun proses pembagian data training dan data testing ditampilkan pada gambar 4.3 seperti berikut :

```
from sklearn.model_selection import train_test_split

X = df.tweet
y = df.kelas

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
# TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

vect = TfidfVectorizer(max_features=600, binary=True)
X_train_vect = vect.fit_transform(X_train)
```

Gambar 4.3 Pembagian data training dan testing

4.1.2 Naive Bayes Classifier

Pada tahap Naive Bayes Classifier perlu melakukan import library Naive Bayes Classifier untuk melihat hasil dari akurasi data training. Proses untuk memasang library Naive Bayes Classifier seperti gambar 4.4 berikut :

```
#Libary NBC
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()
nb.fit(X_train_res, y_train_res)
nb.score(X_train_res, y_train_res)
```

Gambar 4.4 Library Naive Bayes Classifier

Setelah melakukan import library, maka proses selanjutnya menghitung akurasi dan f1-score pada data training. Adapun proses menghitung akurasi dan f1-score pada data training ditampilkan pada gambar 4.5 berikut :

```
# mengukur akurasi dan f1-score

from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred) * 100))
print("\nF1 Score: {:.2f}".format(f1_score(y_test, y_pred, average='weighted') * 100))
print("\nCOnfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Gambar 4.5 Menghitung Akurasi

4.1.3 Cross Validation

Cross Validation merupakan metode untuk memperoleh hasil akurasi yang maksimal. Metode ini dilakukan percobaan sebanyak *k* kali untuk satu model dengan parameter yang sama. Dalam melakukan proses cross validation hal pertama yang perlu dilakukan adalah memasang library. Adapun library untuk melakukan proses cross validation ditunjukkan pada gambar 4.6 berikut :

```
# Library untuk membagi data training
from sklearn.model_selection import ShuffleSplit

X = df.tweet
y = df.kelas

ss = ShuffleSplit(n_splits=10, test_size=0.2)
sm = SMOTE()

accs = []
f1s = []
cms = []

for train_index, test_index in ss.split(X):

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit vectorizer and transform X train, then transform X test
    X_train_vect = vect.fit_transform(X_train)
    X_test_vect = vect.transform(X_test)

    # Oversample
    X_train_res, y_train_res = sm.fit_resample(X_train_vect, y_train)

    # Fit Naive Bayes on the vectorized X with y train labels,
    # then predict new y labels using X test
    nb.fit(X_train_res, y_train_res)
    y_pred = nb.predict(X_test_vect)

    # Determine test set accuracy and f1 score on this fold using the true y labels and predicted y labels
    accs.append(accuracy_score(y_test, y_pred))
    f1s.append(f1_score(y_test, y_pred, average='weighted'))
    cms.append(confusion_matrix(y_test, y_pred))

print("\nAverage accuracy across folds: {:.2f}%".format(sum(accs) / len(accs) * 100))
print("\nAverage F1 score across folds: {:.2f}%".format(sum(f1s) / len(f1s) * 100))
print("\nAverage Confusion Matrix across folds: \n {}".format(sum(cms) / len(cms)))
```

Gambar 4.6 Cross Validation

Kemudian dilakukan percobaan pengujiaan sebanyak 10 kali. Adapun source code pengujian seperti pada gambar 4.7 berikut :

```
#pengujian k fold sebanyak 10 kali

fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(16,9))

acc_scores = [round(a * 100, 1) for a in accs]
f1_scores = [round(f * 100, 2) for f in f1s]

x1 = np.arange(len(acc_scores))
x2 = np.arange(len(f1_scores))

ax1.bar(x1, acc_scores)
ax2.bar(x2, f1_scores, color='#559ebf')

# Place values on top of bars
for i, v in enumerate(list(zip(acc_scores, f1_scores))):
    ax1.text(i - 0.25, v[0] + 2, str(v[0]) + '%')
    ax2.text(i - 0.25, v[1] + 2, str(v[1]))

ax1.set_ylabel('Accuracy (%)')
ax1.set_title('Naive Bayes')
ax1.set_ylim([0, 100])

ax2.set_ylabel('F1 Score')
ax2.set_xlabel('Runs')
ax2.set_ylim([0, 100])

sns.despine(bottom=True, left=True)  # Remove the ticks on axes for cleaner presentation

plt.show()
```

Gambar 4.7 Pengujian k-fold

Setelah melakukan pengujian, maka sistem akan menampilkan grafik dari pengujian k-fold tersebut. Adapun source code proses naive bayes classifier seperti pada gambar 4.7 berikut :

```
#Libary NBC
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()
nb.fit(X_train_res, y_train_res)
nb.score(X_train_res, y_train_res)
```

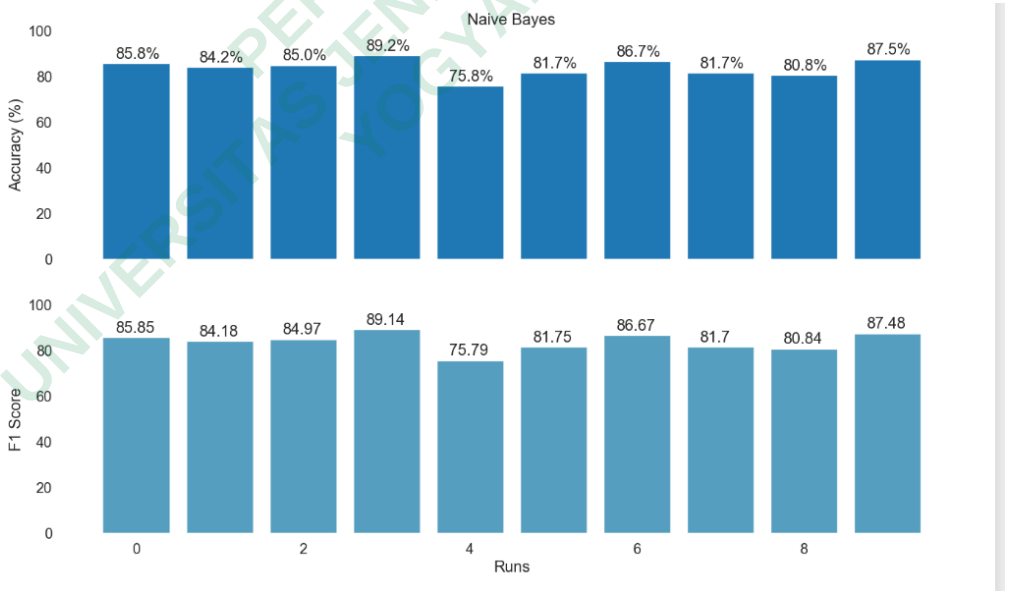
```
0.9173553719008265
```

```
# variable untuk prediksi

X_test_vect = vect.transform(X_test)

y_pred = nb.predict(X_test_vect)

y_pred
```

```
array([1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 0], dtype=int64)
```

Gambar 4.8 proses naive bayes classifier

Berikut adalah proses naive bayes classifier yang menggunakan library sklearn dan variabel untuk prediksi.



Gambar 4.9 hasil percobaan k-fold

Dari grafik tersebut, dapat dilihat bahwa nilai pengujian akurasi dan f1-score tertinggi pada percobaan pengujian ke-4, dengan nilai akurasi 89,2% dan f1-score 89,14%.

Setelah melakukan pengujian, maka selanjutnya membuat model pickle untuk proses data testing selanjutnya. Proses pembautan data model pickle seperti pada gambar 4.9 berikut :

```
#Membuat model klasifikasi, dimasukkan ke dalam file model_classifier_nbc.pickle
import os
import pickle
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer


X = df.tweet
y = df.kelas

txt_classifier = Pipeline([('vect', TfidfVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('classifier', MultinomialNB(alpha=1.0)),
                    ])
X_train = np.asarray(X)
txt_classifier = txt_classifier.fit(X_train, np.asarray(y))
```

Gambar 4.10 Membuat data pickle

4.1.4 Data Testing

Setelah melakukan pembuatan model pickle, maka selanjutnya proses pengujian model pickle terhadap data testing. Tahap pertama untuk melakukan pengujian pada data testing yaitu pemanggilan data. Source code untuk melakukan pemanggilan data pada Jupyter ditunjukkan pada gambar 4.10

```
: #library pandas

import pandas as pd
df = pd.read_excel('testing_bursa.xlsx')
df = pd.DataFrame(df)
df = df.dropna()
df
```

Gambar 4.11 Membaca data

Setelah data berhasil dibaca, maka proses selanjutnya adalah memberikan keterangan berupa TP, TN, FP dan FN pada data testing. Hal ini berfungsi untuk menghitung tingkat akurasi, precisson, recall dan f1-score pada data testing. Adapun kode untuk memberi keterangan pada data testing ditampilkan pada gambar 4.11 berikut :

```
]: #menghitung jumlah TP, FP, TN dan FN

df.loc[(df['actual'] == 1) & (df['predicted'] == 1), 'keterangan'] = 'TP'
df.loc[(df['actual'] == 1) & (df['predicted'] == 0), 'keterangan'] = 'FP'
df.loc[(df['actual'] == 0) & (df['predicted'] == 0), 'keterangan'] = 'TN'
df.loc[(df['actual'] == 0) & (df['predicted'] == 1), 'keterangan'] = 'FN'
```

Gambar 4.12 Data label

Setelah data dilkukan pelabelan berupa TP, FP, TN dan FN. Maka selanjutnya proses menghitung akurasi, precisson, recall dan f1-score. Untuk menghitung nilai akurasi, precisson, recall dan f1-score data testing kode ditampilkan seperti pada gambar 4.12- 4.15berikut :

```
7]: #menghitung akurasi data testing
    accuracy = (TP+TN)/(TP+TN+FP+FN)
    print('Accuracy =', accuracy)

    Accuracy = 0.64
```

Gambar 4.13 Menghitung Akurasi Data Testing

```
: #menghitung precission data testing
precission = (TP) / (TP+FP)
print('Precission =', precission)

Precission = 0.46
```

Gambar 4.14 Menghitung Precission Data Testing

```
]: #menghitung recall data testing
recall = (TP) / (TP + FN)
print('Recall =', recall)

Recall = 0.71875
```

Gambar 4.15 Menghitung Recall Data Testing

```
: #menghitung f1-score data testing
F1_Score = 2 * (recall*precission) / (recall + precission)
print('F1Score =', F1_Score)

F1Score = 0.5609756097560976
```

Gambar 4.16 Menghitung F1Score Data Testing

4.1.5 Klasifikasi

Setelah melakukan perhitungan pada data testing, maka selanjutnya proses menggunakan model pickle terhadap keseluruhan data untuk klasifikasi data. Proses pertama pada tahap klasifikasi adalah membuat membuka model pickle. Source code untuk membuka model pickle ditunjukkan pada gambar 4.16 berikut

```
#membuka model pickle untuk data klasifikasi

model = open('model_nbc_pse.pickle', 'rb')
nbc_classifier = pickle.load(model)
print(nbc_classifier)

Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf', TfidfTransformer()),
                ('classifier', MultinomialNB())])
```

Gambar 4.17 Membuka Model Pickle

Selanjutnya membuka file keseluruhan data, proses pemanggilan data ditunjukkan pada gambar 4.18 berikut :

```
#menampilkan file excel klasifikasi data

df_tweet = pd.read_excel('master_data_adit.xlsx')
df_tweet = pd.DataFrame(df_tweet)
df_tweet = df_tweet.fillna(' ')
df_tweet.head()
```

Gambar 4.18 Pemanggilan Data

```
pd.DataFrame(data_tweet)
```

	tweet
0	mgoalcomidid spesial transfer main lima main p...
1	kaka liverpool mgoalcomidid spesial transfer m...
2	dlvr spesial transfer main bintang la liga spa...
3	infobarca spesial transfer main bintang la lig...
4	fcbinews spesial transfer main bintang la liga...
...	...
11278	real madrid transfer buat tahun w aurelien tch...
11279	isco umum diri resmi tinggal real madrid statu...
11280	pilih transfer real madrid ganti mbappe bintan...
11281	mungkin serius mo salah tinggal liverpool stat...
11282	depan chelsea segera segel transfer dembele ba...

11283 rows × 1 columns

Gambar 4.19 Jumlah Data

Pada gambar 4.19 menujukkan jumlah data yang akan di klasifikasi oleh model. Jumlah data yang terbaca oleh Jupyter sebanyak 11.283 Data. Setelah data terbaca, maka selanjutnya proses prediksi keseluruhan data. Kode untuk melakukan prediksi data ditampilkan pada gambar 4.20 :

```
#melakukan prediksi pada data klasfikasi

predicted = nbc_classifier.predict(np.asarray(data_tweet))
```

```
predicted
```

```
array([1, 1, 0, ..., 1, 1, 0], dtype=int64)
```

Gambar 4.20 Prediksi Data

Maka hasil prediksi akhir dari keseluruhan data 11.283 ditampilkan pada gambar 4.21

```
]: #menampilkan hasil dari data klasifikasi

result_tweet=[]
for i in range(len(predicted)):
    if(predicted[i]==1):
        sentiment_result='Positif'
    elif(predicted[i]==0):
        sentiment_result='Negatif'
    result_tweet.append({'tweet':data_tweet[i], 'class':predicted[i]})
    #     result_tweet.append({'tweet':data_tweet[i], 'class':predicted[i], 'result_nbc':sentiment_result})
```

```
]: data=pd.DataFrame(result_tweet)
   data
```

]:

	tweet	class
0	mgoalcomidid spesial transfer main lima main p...	1
1	kaka liverpool mgoalcomidid spesial transfer m...	1
2	dlvr spesial transfer main bintang la liga spa...	0
3	infobarca spesial transfer main bintang la lig...	0
4	fcbinews spesial transfer main bintang la liga...	0
...	...	...
11278	real madrid transfer buat tahun w aurelien tch...	0
11279	isco umum diri resmi tinggal real madrid statu...	1
11280	pilih transfer real madrid ganti mbappe bintan...	1
11281	mungkin serius mo salah tinggal liverpool stat...	1
11282	depan chelsea segera segel transfer dembele ba...	0

11283 rows × 2 columns

Gambar 4.21 Menampilkan hasil dari data klasifikasi

Pada gambar tersebut merupakan hasil prediksi yang sudah di masukan ke dalam data frame.

4.2 RINGKASAN HASIL PENELITIAN

Hasil penelitian ini merupakan penelitian analisis sentimen *positif* dan sentimen negative. Penggunaan metode *Naïve Bayes Classifer* (NBC) dalam penelitian ini dipilih dikarenakan pada algoritma NBC dapat melakukan proses pengolahan data diskrit dan data kuantitatif dengan menggunakan sampel yang relative sedikit dan juga perhitungan pada algoritma NBC lebih cepat.

Pengambilan data berupa topik mengena keyword “Transfer La Liga”, “Transfer Real Madrid”, “Transfer Barcelona”, “Transfer Liga Spanyol” dan “Transfer Copa Del Ray”. Data tweet di ambil dari periode 1 Januari 2020 sampai dengan 31 Mei 2022, dengan jumlah data total 11.282.

Pada tahap training data yang digunakan sejumlah 600 *tweet* dengan masing-masing 300 positif dan 300 negatif yang sudah diberi label secara manual. Sedangkan untuk data testing jumlah data yang digunakan adalah 200 data yang sudah diberi label positif dan negatif. Hasil dari penelitian ini pada data testing mendapatkan nilai akurasi sebesar 85%. Sedangkan, pada hasil klasifikasi total dari "Bursa Transfer Pemain La Liga Spanyol" mendapatkan nilai 3631 ulasan Positif dan 7652 Ulasan negatif.

```
In [16]: clean={"Username":data['Username'], "Cleaned_Text":data_preprocess}

         df=pd.DataFrame(clean)
         df.to_excel("preprocessing_latihan.xlsx", index=False, encoding='utf8')
```

DATA TRAINING

```
In [17]: # memasang library pandas
         import pandas as pd

In [18]: # pip install openpyxl

In [19]: #membaca data

         df = pd.read_excel (r'pelabelan_manual.xlsx', sheet_name='Sheet1')
         df=pd.DataFrame(df)
         df
```

Out[19]:

	no	tanggal	username	tweet	label	kelas
0	1	2012-03-07 17:32:43+00:00	agentaruhanbola	account transfer main bola situs taruh bola li...	Positif	1
1	2	2010-06-20 09:15:32+00:00	baskaramp	aduh aduh kangen liga inggris spanyol itali ni...	Positif	1
2	3	2021-04-15 16:30:00+00:00	Bolanet	akhir spekulasi barcelona segera umum transfer...	Negatif	0
3	4	2021-04-15 10:28:14+00:00	M88Indo	akhir spekulasi barcelona segera umum transfer...	Negatif	0
4	5	2020-08-30 15:51:35+00:00	_fireshare	aku masuk opini kalau macam insentif bisnis sp...	Negatif	0
...	...	...	...	...	...	...
595	596	2021-04-19 02:10:14+00:00	anggaimaginer	wabilkhusus barcelona bikin hutang sebenernya ...	Negatif	0
596	597	2021-08-25 11:23:58+00:00	minggussenin	waduuuhh bahaya nih transfer main atlet musim ...	Negatif	0
597	598	2012-09-01 14:35:00+00:00	febriantopw	weseet edan tenan psg i moso keluar transfer m...	Negatif	0
598	599	2011-06-30 16:04:19+00:00	maudvannisa	veve tidak masuk d soesial transfer main binta...	Negatif	0

Gambar 4.22 memasang library pandas dan membaca data

Pada gambar tersebut menggunakan pandas untuk membuka file data tweet yang sudah di lakukan pelabelan manual

```
In [20]: #jumlah data yang sudaah di label
         df.label.value_counts()

Out[20]: Positif    300
         Negatif    300
         Name: label, dtype: int64

In [21]: # pip install scikit-learn

In [22]: from sklearn.feature_extraction.text import TfidfVectorizer

         d1 = "real madrid lineup ganti raphael varane tengah minat transfer manchester united"
         d2 = "real madrid tumbal vinicius junior transfer kylian mbappe"
         d3 = "update transfer main sergio aguero sepakat gabung barcelona kontrak tahun depan"
         d4 = "rumor transfer sergio aguero sepakat gabung barcelona musim depan"

         vect = TfidfVectorizer()
         X = vect.fit_transform([d1, d2, d3, d4])

         X.toarray()

Out[22]: array([[0.        , 0.        , 0.        , 0.        , 0.32417842,
                 0.        , 0.        , 0.        , 0.32417842, 0.25558599,
                 0.        , 0.32417842, 0.        , 0.32417842, 0.        ,
                 0.32417842, 0.25558599, 0.        , 0.        , 0.        ,
                 0.        , 0.32417842, 0.16916975, 0.        , 0.32417842,
                 0.        , 0.32417842, 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.39176533, 0.        , 0.39176533, 0.        , 0.30887228,
                 0.        , 0.        , 0.39176533, 0.        , 0.        ,
                 0.        , 0.30887228, 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.2044394 , 0.39176533, 0.        ,
                 0.        , 0.        , 0.39176533],
                [0.2787129 , 0.2787129 , 0.2787129 , 0.2787129 , 0.        ,
                 0.        , 0.35351198, 0.        , 0.        , 0.        ,
                 0.35351198, 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.2787129 , 0.2787129 ,
```

Gambar 4.23 jumlah label dan contoh perhitungan TF IDF

Pada gambar tersebut setelah dilakukan di labeling manuak di dapatkan nilai sentimen positif 300 dan negatif 300, dan contoh perhitungan TF IDF dengan menggunakan contoh dokumen

```
In [23]: list(zip(X.toarray()[0], vect.get_feature_names()))

C:\Users\Asus\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Functio
n get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_na
mes_out instead.
  warnings.warn(msg, category=FutureWarning)

Out[23]: [(0.0, 'aguero'),
 (0.0, 'barcelona'),
 (0.0, 'depan'),
 (0.0, 'gabung'),
 (0.32417842259348545, 'ganti'),
 (0.0, 'junior'),
 (0.0, 'kontrak'),
 (0.0, 'kylian'),
 (0.32417842259348545, 'lineup'),
 (0.255585991926846, 'madrid'),
 (0.0, 'main'),
 (0.32417842259348545, 'manchester'),
 (0.0, 'mbappe'),
 (0.32417842259348545, 'minat'),
 (0.0, 'musim'),
 (0.32417842259348545, 'raphael'),
 (0.255585991926846, 'real'),
 (0.0, 'rumor'),
 (0.0, 'sepakat'),
 (0.0, 'sergio'),
 (0.0, 'tahun'),
 (0.32417842259348545, 'tengah'),
 (0.16916974924594824, 'transfer'),
 (0.0, 'tumbal'),
 (0.32417842259348545, 'united'),
 (0.0, 'update'),
 (0.32417842259348545, 'varane'),
 (0.0, 'vinicius')]
```

Gambar 4.24 Nilai TF IDF

Pada gambar tersebut untuk menampilkan hasil TF IDF.

```
In [24]: list(zip(X.toarray()[1], vect.get_feature_names()))

Out[24]: [(0.0, 'aguero'),
 (0.0, 'barcelona'),
 (0.0, 'depan'),
 (0.0, 'gabung'),
 (0.0, 'ganti'),
 (0.39176532687938803, 'junior'),
 (0.0, 'kontrak'),
 (0.39176532687938803, 'kylian'),
 (0.0, 'lineup'),
 (0.3088722835775363, 'madrid'),
 (0.0, 'main'),
 (0.0, 'manchester'),
 (0.39176532687938803, 'mbappe'),
 (0.0, 'minat'),
 (0.0, 'musim'),
 (0.0, 'raphael'),
 (0.3088722835775363, 'real'),
 (0.0, 'rumor'),
 (0.0, 'sepakat'),
 (0.0, 'sergio'),
 (0.0, 'tahun'),
 (0.0, 'tengah'),
 (0.20443939970227634, 'transfer'),
 (0.39176532687938803, 'tumbal'),
 (0.0, 'united'),
 (0.0, 'update'),
 (0.0, 'varane'),
 (0.39176532687938803, 'vinicius')]

In [25]: vect = TfidfVectorizer(max_features=600, binary=True)
X = vect.fit_transform(df.tweet)

X.toarray()
```

Gambar 4.25 menampilkan index 1

Pada gambar tersebut untuk menampilkan perhitungan TF IDF pada dokumen 2

```
Out[25]: array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.5054859 , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                ...,
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.51594285,
                 0.58317616]])
```

```
In [26]: from sklearn.model_selection import train_test_split

         X = df.tweet
         y = df.kelas

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
In [27]: # TF-IDF
         from sklearn.feature_extraction.text import TfidfVectorizer

         vect = TfidfVectorizer(max_features=600, binary=True)
         X_train_vect = vect.fit_transform(X_train)
```

Gambar 4.26 mendefinisikan nilai x dan y

Pada gambar tersebut mendefinisikan nilai x sebagai data tweet dan nilai y sebagai sentimen nya

```
In [28]: print(X_train_vect)
```

```
  (0, 41)       0.39987443158930697
  (0, 372)      0.3245085236092023
  (0, 510)      0.15146427712335966
  (0, 310)      0.14884177177178856
  (0, 594)      0.3934475370310798
  (0, 180)      0.387464241341971
  (0, 281)      0.39987443158930697
  (0, 219)      0.36696380519165195
  (0, 54)       0.17846576208167836
  (0, 330)      0.14278470033188853
  (0, 554)      0.086903875612982
  (0, 104)      0.17440778721857347
  (1, 183)      0.4192044680787169
  (1, 161)      0.4192044680787169
  (1, 566)      0.4192044680787169
  (1, 483)      0.4192044680787169
  (1, 511)      0.4192044680787169
  (1, 7)        0.30638419686594215
  (1, 54)       0.14900403911744486
  (1, 554)      0.07255749411121197
  (2, 30)       0.483264231457467
  (2, 418)      0.5143553079374337
  (2, 535)      0.5143553079374337
  (2, 499)      0.358526823044597
  (2, 328)      0.17823108375117305
  :     :
  (478, 510)    0.1947039667655193
  (478, 310)    0.19133279433785289
  (478, 330)    0.18354656342764183
  (478, 554)    0.11171300342564598
  (479, 198)    0.26211787116325336
  (479, 38)     0.2751424264436363
  (479, 341)    0.25201524121908114
  (479, 550)    0.25201524121908114
```

Gambar 4.27 perhitungan tf idf

Pada gambar tersebut merupakan perhitungan TF IDF pada data testing

```
In [29]: # pip install imblearn

In [30]: #Libary prediksi

         from imblearn.over_sampling import SMOTE

         sm = SMOTE()

         X_train_res, y_train_res = sm.fit_resample(X_train_vect, y_train)

In [31]: import numpy as np

In [32]: unique, counts = np.unique(y_train_res, return_counts=True)
         print(list(zip(unique, counts)))

         [(0, 242), (1, 242)]
```

Naive Bayes Clasifier

```
In [33]: #Libary NBC
         from sklearn.naive_bayes import MultinomialNB

         nb = MultinomialNB()
         nb.fit(X_train_res, y_train_res)
         nb.score(X_train_res, y_train_res)

Out[33]: 0.9173553719008265
```

Gambar 4.28 melakukan perhitungan smote dan naive bayes

Pada gambar tersebut melakukan perhitungan smote dan naive bayes

```
In [34]: # variable untuk prediksi

         X_test_vect = vect.transform(X_test)

         y_pred = nb.predict(X_test_vect)

         y_pred

Out[34]: array([1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
                1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
                1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
                0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
                1, 1, 1, 0, 0, 0, 1, 0, 1, 0], dtype=int64)

In [35]: # mengukur akurasi dan f1-score

         from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

         print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred) * 100))
         print("\nF1 Score: {:.2f}".format(f1_score(y_test, y_pred, average='weighted') * 100))
         print("\nCOnfusion Matrix:\n", confusion_matrix(y_test, y_pred))

         Accuracy: 85.00%

         F1 Score: 85.00

         COnfusion Matrix:
          [[53  9]
          [ 9 49]]

In [36]: # pip install seaborn
```

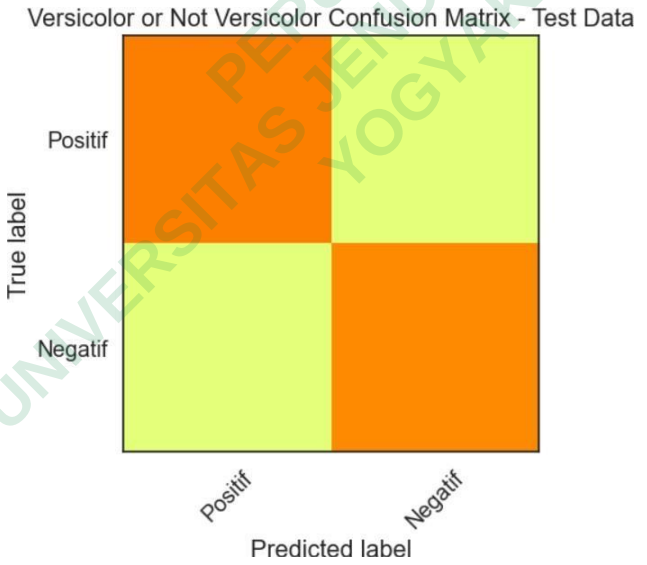
Gambar 4.29 menghitung variabel untuk prediksi dan mengukur akurasi dan f1-score

Pada gambar tersebut melakukan perhitungan variabel untuk prediksi dan mengukur akurasi dan f1-score

```
In [37]: #pembuatan matriks TP, TN. FP, FN

import math
import random
from collections import defaultdict
from pprint import pprint

# Prevent future/deprecation warnings from showing in output
import warnings
warnings.filterwarnings(action='ignore')

import seaborn as sns
import matplotlib.pyplot as plt

# Set global styles for plots
sns.set_style(style='white')
sns.set_context(context='notebook', font_scale=1.3, rc={'figure.figsize': (16,9)})

cm=confusion_matrix(y_test, y_pred)
plt.clf()
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Positif','Negatif']
plt.title('Versicolor or Not Versicolor Confusion Matrix - Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
# s = [['TN','FP'], ['FN', 'TP']]
# for i in range(3):
#     for j in range(3):
#         plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
plt.show()
```

Gambar 4.30 melakukan pembuatan matriks TP, TN, FP, FN

Pada gambar tersebut untuk membuat program yang menampilkan confusion matrix



Gambar 4.31 menampilkan confusion matrix

Pada gambar tersebut tabel confusion matrix yang menggambarkan prediksi dan aktual positif negatif daa tweet

Cross Validation

In [38]:
```
# Library untuk membagi data training
from sklearn.model_selection import ShuffleSplit

X = df.tweet
y = df.kelas

ss = ShuffleSplit(n_splits=10, test_size=0.2)
sm = SMOTE()

accs = []
f1s = []
cms = []

for train_index, test_index in ss.split(X):

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit vectorizer and transform X train, then transform X test
    X_train_vect = vect.fit_transform(X_train)
    X_test_vect = vect.transform(X_test)

    # Oversample
    X_train_res, y_train_res = sm.fit_resample(X_train_vect, y_train)

    # Fit Naive Bayes on the vectorized X with y train labels,
    # then predict new y labels using X test
    nb.fit(X_train_res, y_train_res)
    y_pred = nb.predict(X_test_vect)
```

Gambar 4.32 library untuk membagi data training

Pada gambar tersebut melakukan klasifikasi terhadap nilai x dan y data testing dengan menggunakan cross validation

```
    # Oversample
    X_train_res, y_train_res = sm.fit_resample(X_train_vect, y_train)

    # Fit Naive Bayes on the vectorized X with y train labels,
    # then predict new y labels using X test
    nb.fit(X_train_res, y_train_res)
    y_pred = nb.predict(X_test_vect)

    # Determine test set accuracy and f1 score on this fold using the true y labels and predicted y labels
    accs.append(accuracy_score(y_test, y_pred))
    f1s.append(f1_score(y_test, y_pred, average='weighted'))
    cms.append(confusion_matrix(y_test, y_pred))

print("\nAverage accuracy across folds: {:.2f}%".format(sum(accs) / len(accs) * 100))
print("\nAverage F1 score across folds: {:.2f}%".format(sum(f1s) / len(f1s) * 100))
print("\nAverage Confusion Matrix across folds: \n {}".format(sum(cms) / len(cms)))
```

```
Average accuracy across folds: 83.83%

Average F1 score across folds: 83.84%

Average Confusion Matrix across folds:
 [[52.1  9.3]
 [10.1 48.5]]
```

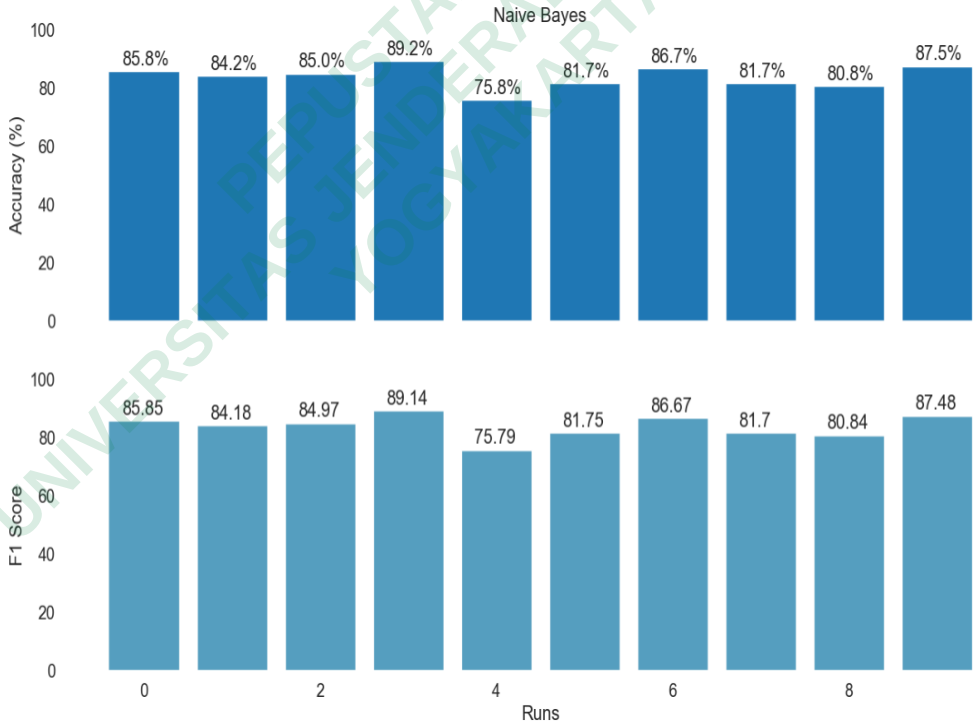
Gambar 4.33 hasil dari cross validation

Pada gambar tersebut merupakan perhitungan akurasi f1 score dan confusion matrix

```
In [39]: #pengujian k fold sebanyak 10 kali

fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(16,9))

acc_scores = [round(a * 100, 1) for a in accs]
f1_scores = [round(f * 100, 2) for f in f1s]

x1 = np.arange(len(acc_scores))
x2 = np.arange(len(f1_scores))

ax1.bar(x1, acc_scores)
ax2.bar(x2, f1_scores, color='#559ebf')

# Place values on top of bars
for i, v in enumerate(list(zip(acc_scores, f1_scores))):
    ax1.text(i - 0.25, v[0] + 2, str(v[0]) + '%')
    ax2.text(i - 0.25, v[1] + 2, str(v[1]))

ax1.set_ylabel('Accuracy (%)')
ax1.set_title('Naive Bayes')
ax1.set_ylim([0, 100])

ax2.set_ylabel('F1 Score')
ax2.set_xlabel('Runs')
ax2.set_ylim([0, 100])

sns.despine(bottom=True, left=True)  # Remove the ticks on axes for cleaner presentation

plt.show()
```

Gambar 4.34 melakukan pengujian K Fold

Pada gambar tersebut merupakan program untuk menampilkan grafik K Fold



Gambar 4.35 hasil cross validation

Pada gambar tersebut menampilkan hasil cross validation yang dilakukan sebanyak 10 kali

```
In [40]: #Membuat model klasifikasi, dimasukkan ke dalam file model_classifier_nbc.pickle
```

```
In [41]: #Membuat model klasifikasi, dimasukkan ke dalam file model_classifier_nbc.pickle
import os
import pickle
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer


X = df.tweet
y = df.kelas

txt_classifier = Pipeline([('vect', TfidfVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('classifier', MultinomialNB(alpha=1.0)),
                    ])
X_train = np.asarray(X)
txt_classifier = txt_classifier.fit(X_train, np.asarray(y))
```

```
In [42]: #simpan ke PICKLE
files = open('model_nbc.pickle', 'wb')
pickle.dump(txt_classifier, files)
files.close()

print('Proses Training Naive Bayes Selesai!')
```

```
Proses Training Naive Bayes Selesai!
```

Gambar 4.36 membuat model klasifikasi

Pada gambar tersebut untuk membuat model klasifikasi dan di simpan ke dalam bentuk file pickle

```
In [43]: #simpan ke PICKLE
model = open('model_nbc.pickle', 'rb')
nbc_classifier = pickle.load(model)
print(nbc_classifier)
```

```
Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf', TfidfTransformer()),
                ('classifier', MultinomialNB())])
```

```
In [44]:
df_tweet = pd.read_excel (r'data_testing.xlsx')
df_tweet=pd.DataFrame(df_tweet)
df_tweet=df_tweet.dropna()
df_tweet.head()
```

Out[44]:

	no	tanggal	username	tweet	label	kelas
0	1	2020-09-08 05:38:02+00:00	OctaHannnnn	afk transfer deal done everton resmi dapat jam...	Negatif	0
1	2	2020-09-08 00:01:51+00:00	barcastuff_idn	agen lautaro martinez datang italia barcelona ...	Positif	1
2	3	2020-09-04 17:18:38+00:00	neragrana	akhir saga transfer lionel messi bintang asal ...	Negatif	0
3	4	2021-08-29 01:22:04+00:00	adit_putra08	aku manajemen chelsea jdi lebih bagus skrng so...	Positif	1
4	5	2020-09-09 21:30:00+00:00	indosportdotcom	anchester united pasti dapat untung real madri...	Positif	1

```
In [45]: data_tweet = df_tweet.tweet
```

Gambar 4.37 membuka ke file pickle

Pada gambar tersebut untuk membuka file pickleyang sudah di buat

```
In [46]: pd.DataFrame(data_tweet)
```

Out[46]:

	tweet
0	afk transfer deal done everton resmi dapat jam...
1	agen lautaro martinez datang italia barcelona ...
2	akhir saga transfer lionel messi bintang asal ...
3	aku manajemen chelsea jdi lebih bagus skrng so...
4	anchester united pasti dapat untung real madri...
...	...
195	video bursa transfer chelsea pinjam bintang ba...
196	wujud ingin ronald koeman belanja main petingg...
197	wujud ingin ronald koeman belanja main petingg...
198	yang takut kalo saing macam real madridjuventu...
199	yaudah lah ikhlasin aja banyak ujung tombak gu...

200 rows × 1 columns

```
In [47]: #melakukan prediksi
         predicted = nbc_classifier.predict(np.asarray(data_tweet))
```

Gambar 4.38 menampilkan data frame pada data training

Pada gambar tersebut menampilkan data training

```
In [48]: predicted
```

```
Out[48]: array([1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
                1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
                0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0], dtype=int64)
```

```
In [49]: #menampilkan hasil positif dan negatif
         result_tweet=[]
         for i in range(len(predicted)):
             if(predicted[i]==1):
                 sentiment_result='Positif'
             elif(predicted[i]==0):
                 sentiment_result='Negatif'
         #     result_tweet.append({'class':prediction_linear[i], 'result_nbc':sentiment_result})
             result_tweet.append({'Cleaned_Text':data_tweet[i], 'class':predicted[i] })
```

Gambar 4.39 menampilkan hasil positif dan negatif

Pada gambar tersebut menampilkan hasil prediksi pada data training

```
In [50]: data=pd.DataFrame(result_tweet)
         data.head()
```

Out[50]:

	Cleaned_Text	class
0	afk transfer deal done everton resmi dapat jam...	1
1	agen lautaro martinez datang italia barcelona ...	0
2	akhir saga transfer lionel messi bintang asal ...	0
3	aku manajemen chelsea jdi lebih bagus skrng so...	0
4	anchester united pasti dapat untung real madri...	1

```
In [51]: data.to_excel('testing_bursa2.xlsx')
```

Gambar 4.40 menyimpan hasil data testing

Pada gambar tersebut menyimpan hasil prediksi data testing dengan nama bursa2

DATA TESTING

```
In [52]: #library pandas

         import pandas as pd
         df = pd.read_excel('testing_bursa.xlsx')
         df = pd.DataFrame(df)
         df = df.dropna()
         df
```

Out[52]:

	Unnamed: 0	Cleaned_Text	actual	predicted
0	1	afk transfer deal done everton resmi dapat jam...	0	1
1	2	agen lautaro martinez datang italia barcelona ...	1	0
2	3	akhir saga transfer lionel messi bintang asal ...	0	0
3	4	aku manajemen chelsea jdi lebih bagus skrng so...	1	0
4	5	anchester united pasti dapat untung real madri...	1	1
...	...	...	...	...
195	196	video bursa transfer chelsea pinjam bintang ba...	1	0
196	197	wujud ingin ronald koeman belanja main petingg...	1	0
197	198	wujud ingin ronald koeman belanja main petingg...	1	0
198	199	yang takut kalo saing macam real madridjuventu...	0	0
199	200	yaudah lah ikhlasin aja banyak ujung tombak gu...	0	0

200 rows × 4 columns

Gambar 4.41 membuka file label manual

Pada gambar tersebut sebelum melakukan testing data membuka file data tweet yang sudah di lakukan pelabelan manual

```
In [53]: #menghitung jumlah TP, FP, TN dan FN

         df.loc[(df['actual'] == 1) & (df['predicted'] == 1), 'keterangan'] = 'TP'
         df.loc[(df['actual'] == 1) & (df['predicted'] == 0), 'keterangan'] = 'FP'
         df.loc[(df['actual'] == 0) & (df['predicted'] == 0), 'keterangan'] = 'TN'
         df.loc[(df['actual'] == 0) & (df['predicted'] == 1), 'keterangan'] = 'FN'
```

```
In [54]: df
```

Out[54]:

	Unnamed: 0	Cleaned_Text	actual	predicted	keterangan
0	1	afk transfer deal done everton resmi dapat jam...	0	1	FN
1	2	agen lautaro martinez datang italia barcelona ...	1	0	FP
2	3	akhir saga transfer lionel messi bintang asal ...	0	0	TN
3	4	aku manajemen chelsea jdi lebih bagus skrng so...	1	0	FP
4	5	anchester united pasti dapat untung real madri...	1	1	TP
...	...	...	...	...	...
195	196	video bursa transfer chelsea pinjam bintang ba...	1	0	FP
196	197	wujud ingin ronald koeman belanja main petingg...	1	0	FP
197	198	wujud ingin ronald koeman belanja main petingg...	1	0	FP
198	199	yang takut kalo saing macam real madridjuventu...	0	0	TN
199	200	yaudah lah ikhlasin aja banyak ujung tombak gu...	0	0	TN

200 rows × 5 columns

Gambar 4.42 Menghitung jumlah TP, FP, TN, dan FN

Pada gambar tersebut untuk menghitung jumlah pada TP, FP, TN dan, FN

```
In [55]: df.groupby(by='keterangan').agg('count')
```

Out[55]:

keterangan	Unnamed: 0	Cleaned_Text	actual	predicted
FN	18	18	18	18
FP	54	54	54	54
TN	82	82	82	82
TP	46	46	46	46

```
In [56]: TP=df['keterangan'].value_counts()['TP']
         TN=df['keterangan'].value_counts()['TN']
         FP=df['keterangan'].value_counts()['FP']
         FN=df['keterangan'].value_counts()['FN']
```

```
In [57]: #menghitung akurasi data testing
         accuracy = (TP+TN)/(TP+TN+FP+FN)
         print('Accuracy =', accuracy)
```

```
Accuracy = 0.64
```

```
In [58]: #menghitung precission data testing
         precission = (TP) / (TP+FP)
         print('Precission =', precission)
```

```
Precission = 0.46
```

```
In [59]: #menghitung recall data testing
         recall = (TP) / (TP + FN)
         print('Recall =', recall)
```

```
Recall = 0.71875
```

Gambar 4.43 Menghitung akurasi data, precission data, recall data testing

Pada gambar tersebut menjelaskan tentang menghitung akurasi data testing, precission data testing, dan recall data testing

```
In [60]: #menghitung f1-score data testing
         F1_Score = 2 * (recall*precission) / (recall + precission)
         print('F1Score =', F1_Score)

         F1Score = 0.5609756097560976
```

Gambar 4.44 Menghitung F-1 score data testing

Pada gambar tersebut untuk menghitung nilai f1 score

KLASIFIKASI

```
In [61]: #membuka model pickle untuk data klasifikasi

         model = open('model_nbc_pse.pickle', 'rb')
         nbc_classifier = pickle.load(model)
         print(nbc_classifier)

         Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf', TfidfTransformer()),
                         ('classifier', MultinomialNB())])
```

```
In [62]: #menampilkan file excel klasifikasi data

         df_tweet = pd.read_excel('master_data_adit.xlsx')
         df_tweet = pd.DataFrame(df_tweet)
         df_tweet = df_tweet.fillna(' ')
         df_tweet.head()
```

Out[62]:

	no	tanggal	username	tweet
0	1	2011-05-10 11:24:26+00:00	irvanfadhilla	mgoalcomidid spesial transfer main lima main p...
1	2	2011-05-10 11:41:00+00:00	agri_to	kaka liverpool mgoalcomidid spesial transfer m...
2	3	2011-06-09 16:50:16+00:00	neomaverix	dlvr spesial transfer main bintang la liga spa...
3	4	2011-06-09 17:12:20+00:00	AlbicelesteID	infobarca spesial transfer main bintang la lig...
4	5	2011-06-09 17:14:13+00:00	FCBI_Semarang	fcbinews spesial transfer main bintang la liga...

```
In [63]: data_tweet = df_tweet.tweet
```

Gambar 4.45 Membuka model pickle dan menampilkan file excel pada klasifikasi data

Pada gambar tersebut untuk membuat model klasifikasi

```
In [64]: pd.DataFrame(data_tweet)
```

Out[64]:

	tweet
0	mgoalcomidid spesial transfer main lima main p...
1	kaka liverpool mgoalcomidid spesial transfer m...
2	dlvr spesial transfer main bintang la liga spa...
3	infobarca spesial transfer main bintang la lig...
4	fcbinews spesial transfer main bintang la liga...
...	...
11278	real madrid transfer buat tahun w aurelien tch...
11279	isco umum diri resmi tinggal real madrid statu...
11280	pilih transfer real madrid ganti mbappe bintan...
11281	mungkin serius mo salah tinggal liverpool stat...
11282	depan chelsea segera segel transfer dembele ba...

11283 rows × 1 columns

```
In [65]: #melakukan prediksi pada data klasfikasi

         predicted = nbc_classifier.predict(np.asarray(data_tweet))

In [66]: predicted
Out[66]: array([1, 1, 0, ..., 1, 1, 0], dtype=int64)
```

Gambar 4.46 Melakukan prediksi pada data klasifikasi

Pada gambar tersebut untuk melakukan klasifikasi terhadap data uji coba sebanyak 11.282 data tweet

```
In [67]: #menampilkan hasil dari data klasifikasi

result_tweet=[]
for i in range(len(predicted)):
    if(predicted[i]==1):
        sentiment_result='Positif'
    elif(predicted[i]==0):
        sentiment_result='Negatif'
    result_tweet.append({'tweet':data_tweet[i], 'class':predicted[i]})
    #     result_tweet.append({'tweet':data_tweet[i], 'class':predicted[i], 'result_nbc':sentiment_result})
```

```
In [68]: data=pd.DataFrame(result_tweet)
data
```

Out[68]:

	tweet	class
0	mgoalcomidid spesial transfer main lima main p...	1
1	kaka liverpool mgoalcomidid spesial transfer m...	1
2	dlvr spesial transfer main bintang la liga spa...	0
3	infobarca spesial transfer main bintang la lig...	0
4	fcbinews spesial transfer main bintang la liga...	0
...	...	...
11278	real madrid transfer buat tahun w aurelien tch...	0
11279	isco umum diri resmi tinggal real madrid statu...	1
11280	pilih transfer real madrid ganti mbappe bintan...	1
11281	mungkin serius mo salah tinggal liverpool stat...	1
11282	depan chelsea segera segel transfer dembele ba...	0

11283 rows × 2 columns

Gambar 4.47 menampilkan hasil dari data klasifikasi

Pada gambar tersebut merupakan hasil klasifikasi yang sudah di masukan ke dalam data frame

```
In [69]: #menghitung jumlah sentimen

data.groupby(by='class').agg('count')
```

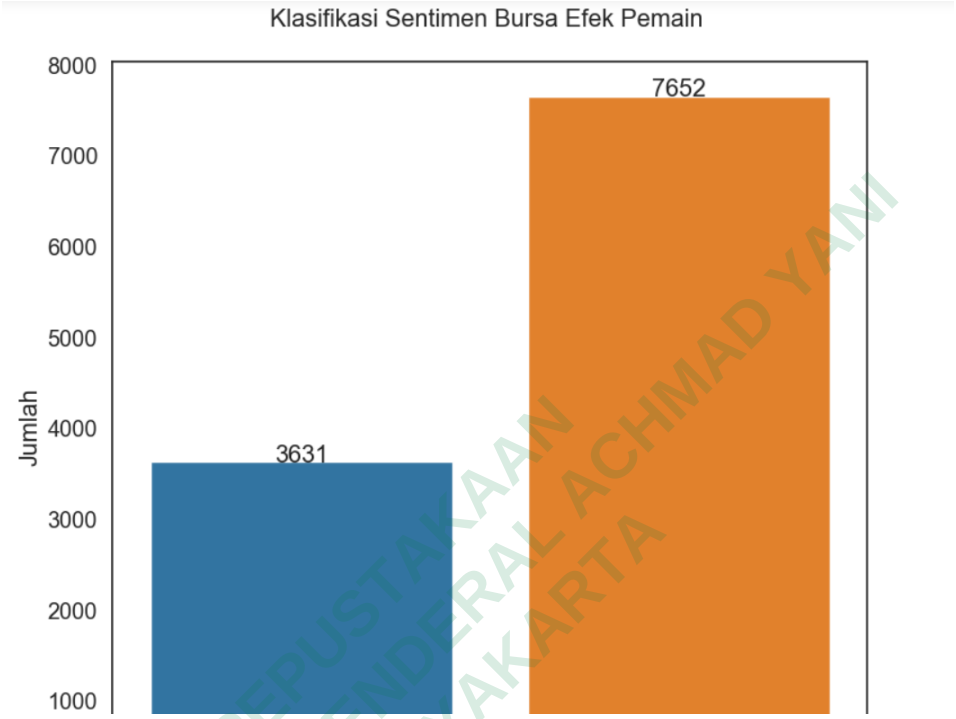
Out[69]:

class	tweet
0	7652
1	3631

```
In [70]: #Visualisasi sentimen negatif dan positif

import matplotlib.pyplot as plt
import seaborn as sns

plt.rcParams["figure.figsize"] = [8,8]
plt.rcParams["figure.autolayout"] = True
x = ['Positif', 'Negatif',]
y = [3631,7652]
percentage = [3631,7652]
ax = sns.barplot(x=x, y=y)
patches = ax.patches
for i in range(len(patches)):
    x = patches[i].get_x() + patches[i].get_width()/2
    y = patches[i].get_height()+.5
    ax.annotate('{:}'.format(percentage[i]), (x, y), ha='center')

plt.title('Klasifikasi Sentimen Bursa Efek Pemain \n')
plt.xlabel('Jenis Klasifikasi')
plt.ylabel('Jumlah')
plt.show()
```

Gambar 4.48 Visualisasi sentimen negatif dan positif

Pada gambar tersebut merupakan kode program untuk menampilkan histogram



Gambar 4.49 Menampilkan Hasil Klasifikasi Data

Pada hasil klasifikasi di dapatkan 3631 data data berlabel positif dan 7652 berlabel negatif

4.3 HASIL EVALUASI DAN MODEL KLASIFIKASI

Evaluasi model klasifikasi menggunakan 1.000 data *tweet* untuk melakukan data training yang dimana masing-masing data 500 positif dan 500 negatif yang sudah dilabel secara manual. Untuk memeriksa keakuratan aplikasi yang dibuat, kita perlu menguji *confusion matrix* untuk melihat nilai data aktual dan klasifikasi. Tabel 4.1 menunjukkan hasil dari perhitungan *confusion matrix*

Tabel 4.1 Confusion Matrix Data Training

Kelas Klasifikasi	Kelas Aktual	
	Positif	Negatif
Positif	48	13
Negatif	9	50

Hasil confussion matrix pada tabel 4.1 mendapatkan hasil dengan *True Positive* (TP) = 48, *True Negative* (TN) = 50, *False Positive* FP = 13 dan *False Negative* (FN) = 9. Setelah mengetahui masing-masing nilainya, maka dapat menghitung nilai *accuracy* dengan rumus persamaan dan menghitung nilai *f-measure* dengan rumus persamaan. Berikut merupakan masing-masing nilai pada *accuracy* dan *f-measure* dapat dilihat pada Tabel 4.2

Tabel 4.2 Akurasi Data Training

Akurasi Data Training	Hasil
Accuracy	81,67 %
F-measure	81, 66%

Dengan penghitungan *confusion matrix* pada cross validation untuk menguji keakuratan dengan melihat nilai data aktual dan klasifikasi. Berikut hasil perhitungan cross validation dapat dilihat pada Tabel 4.3

Tabel 4.3 Confusion Matrix Cross Validation

Kelas Klasifikasi	**Kelas Aktual**	
	Positif	**Negatif**
Positif	48	11
Negatif	9	53

Hasil confusion matrix pada Tabel 4.3 mendapatkan hasil dengan TP = 48, TN = 53, FP = 11 dan FN = 9. Setelah mendapatkan nilai *confusion matrix* dapat dilanjutkan untuk penghitungan akurasi model dari cross validation. Menguji *k-fold* cross validation dilakukan agar mengetahui angka yang sesuai dari perulangan yang sebanyak 10 kali. Dan perhitungan tiap fold yang terdapat pada *k-fold cross validation* sudah mendapatkan nilainya sendiri-sendiri dan hasil yang

berbeda pada tiap-tiap foldnya. Berikut hasil perhitungan 10 fold cross validation pada *accuracy* dan *f-measure* terdapat pada Tabel 4.4

Tabel 4.4 Nilai K-fold Accuracy dan F-measure

Fold	Accuracy	F-measure
Fold 1	85 %	84,99 %
Fold 2	86,7 %	86,75 %
Fold 3	81,7 %	81,68 %
Fold 4	82,5 %	82,56 %
Fold 5	80 %	80 %
Fold 6	80,8 %	80,84 %
Fold 7	79,2 %	79,18 %
Fold 8	88,3 %	88,28 %
Fold 9	72,5 %	72,65 %
Fold 10	80,0 %	80,02 %

Dari hasil yang sudah didapatkan dari perhitungan *cross validation* yang sudah dilakukan sebanyak 10 kali maka didapatkan nilai rata-rata akurasi yang baik yaitu 83,67 % untuk *accuracy* dan 83,64% untuk *f-measure.*

Pada tahap selanjutnya yaitu data *testing* jumlah data *tweet* yang digunakan adalah 200 dan masing-masing telah diberi label 100 positif dan 100 negatif secara manual. Dan untuk mengetahui nilai akurasi pada tahap testing ini diperlukan penghitung *confusion matrix* agar mengetahui perbedaan nilai antara data *training* dan data *testing*. Berikut hasil perhitungan *confusion matrix* data *testing* dapat dilihat pada Tabel 4.5

Tabel 4.5 Confusion Matrix Data Testing

Kelas Klasifikasi	**Kelas Aktual**	
	Positif	**Negatif**
Positif	78	21
Negatif	9	92

Maka dapat menghitung nilai akurasi pada data testing yang dimana sudah diketahui nilai TP = 78, TN = 92, FP = 21 dan FN = 9 dengan menggunakan rumis persamaan untuk *accuracy* dan rumus persamaan untuk menghitung nilai f-measure. Berikut untuk mengetahui nilai *accuracy* dan *f-measure* pada data *testing* pada Tabel 4.6.

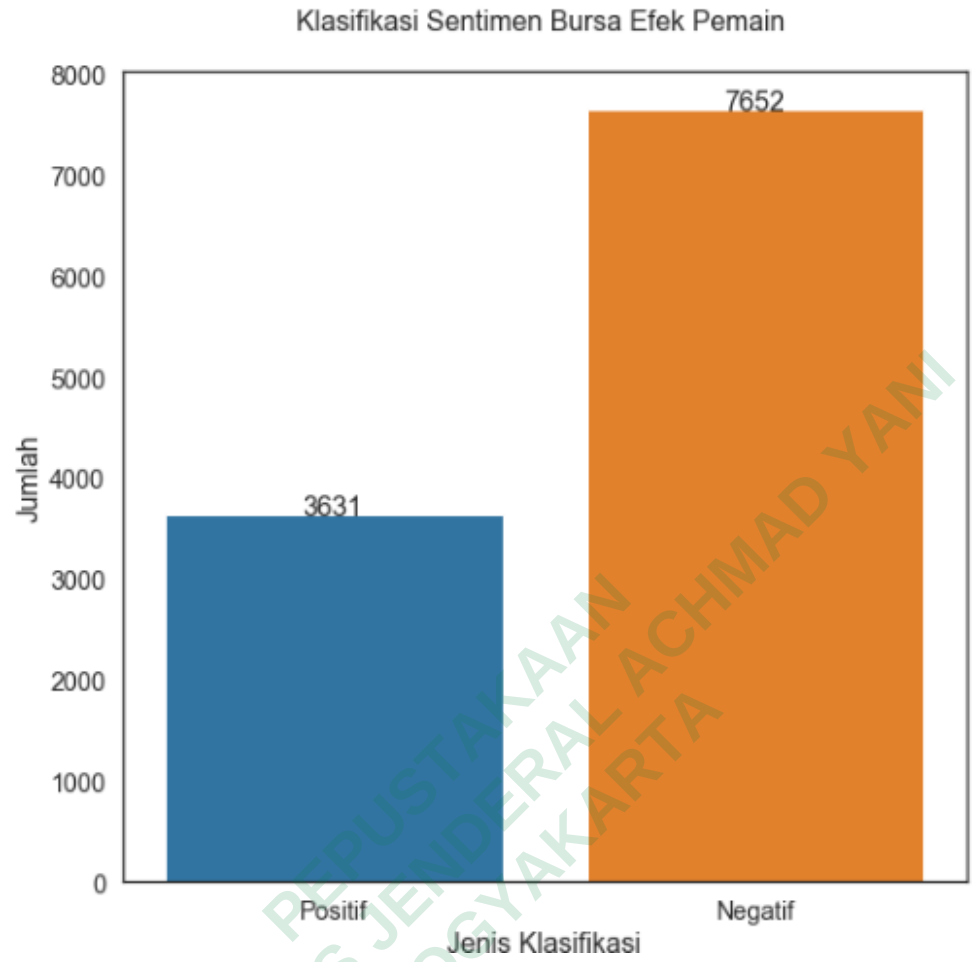
Tabel 4.6 Akurasi Data Testing

Akurasi Data Testing	Hasil
Accuracy	85 %
F-measure	78 %

Pada penjelasan Tabel 4.6 sudah diketahui bahwasanya nilai *accuracy* dan *f-measure* dari data testing dengan jumlah 200 data *tweet* yang sudah dilabeli secara manual dengan mendapatkan nilai 85 % dan 78%.

4.4 HASIL ANALISIS

Pada hasil analisis yaitu menghitung jumlah data keseluruhan data *tweet* yang sudah terambil dengan total 11.282 data, dengan rincian 600 data *tweet* yang telah dilabeli secara manual dengan masing-masing nilai yang sama dalam proses *training* kemudian dari data 10.682 diambil 200 data *tweet* untuk proses testing yang dimana telah didapatkannya nilai akurasi yang baik dari tahap training dan testing, hasil klasifikasi pada data tweet keseluruhan yang sudah diketahui hasil positif dan negatif dapat dilihat pada Gambar 4.1



Gambar 4.1 Histogram Data Positif dan Negatif

Pada gambar 4.1 telah didapatkan hasil histogram dengan jumlah 3631 Label Positif dan 7652 Label Negatif. Pada sentimen negatif pengguna Twitter dikarenakan banyak klub sepak bola sering mengeluarkan banyak uang tetapi tidak ada prestasi demi klub sepak bola itu sendiri. Kemudian klub sepak bola juga sering salah membeli pemain, sehingga tidak menghasilkan prestasi demi klub sepak bola di tim tersebut. Berikut merupakan contoh data tweet negatif pada Tabel 4.7

Tabel 4.7 Data Tweet Negatif

No	Data Tweet
1	jorjoran belanja bursa transfer musim panas juara laliga spanyol real madrid jual empat main tak guna tutup rugi finansial
2	luka modric transfer buruk liga spanyol main gelandang barca
3	gaji main transfer liga spanyol musim depan kurang akibat pandemi
4	iya bener barcelona rugi kalau kontrak messi habis tahun free transfer
5	kisruh soal lionel messi dampak transfer lautaro martinez lautaro kabar tolak barcelona

Sedangka sentimen Positif pada data tweet banyak yang membahas tentang peluang transfer pemain, tentang penempatan posisi pemain di dalam klub sepak bola dan nilai gaji dari pemain. Berikut Contoh beberapa data *tweet* positif terdapat pada Tabel 4.8

Tabel 4.8 Data Tweet Positif

No	Data Tweet
1	peluang arsenal dapat gelandang real madrid bursa transfer musim dingin buka lebar bursatransfer
2	roberto chen gabung malaga malaga prlm klub anggota primera liga spanyol malaga rampung transfer main bertahan
3	rejeki nomplok real madrid proses transfer achraf hakim homealone
4	mantan asisten latih real madrid jose morais nilai erling haaland cocok jadi terus melanjutkan kontrak
5	real madrid mulai tabung transfer kylian mbappe

Berdasarkan hasil klasifikasi sentimen pada data *tweet,* banyak yang diklasifikasikan ke negatif mengenai pandangan pengguna Twitter terhadap transfer pamain klub sepak bola yang tidak sesuai serta tidak adanya prestasi yang membanggakan terhadap klub tersebut.