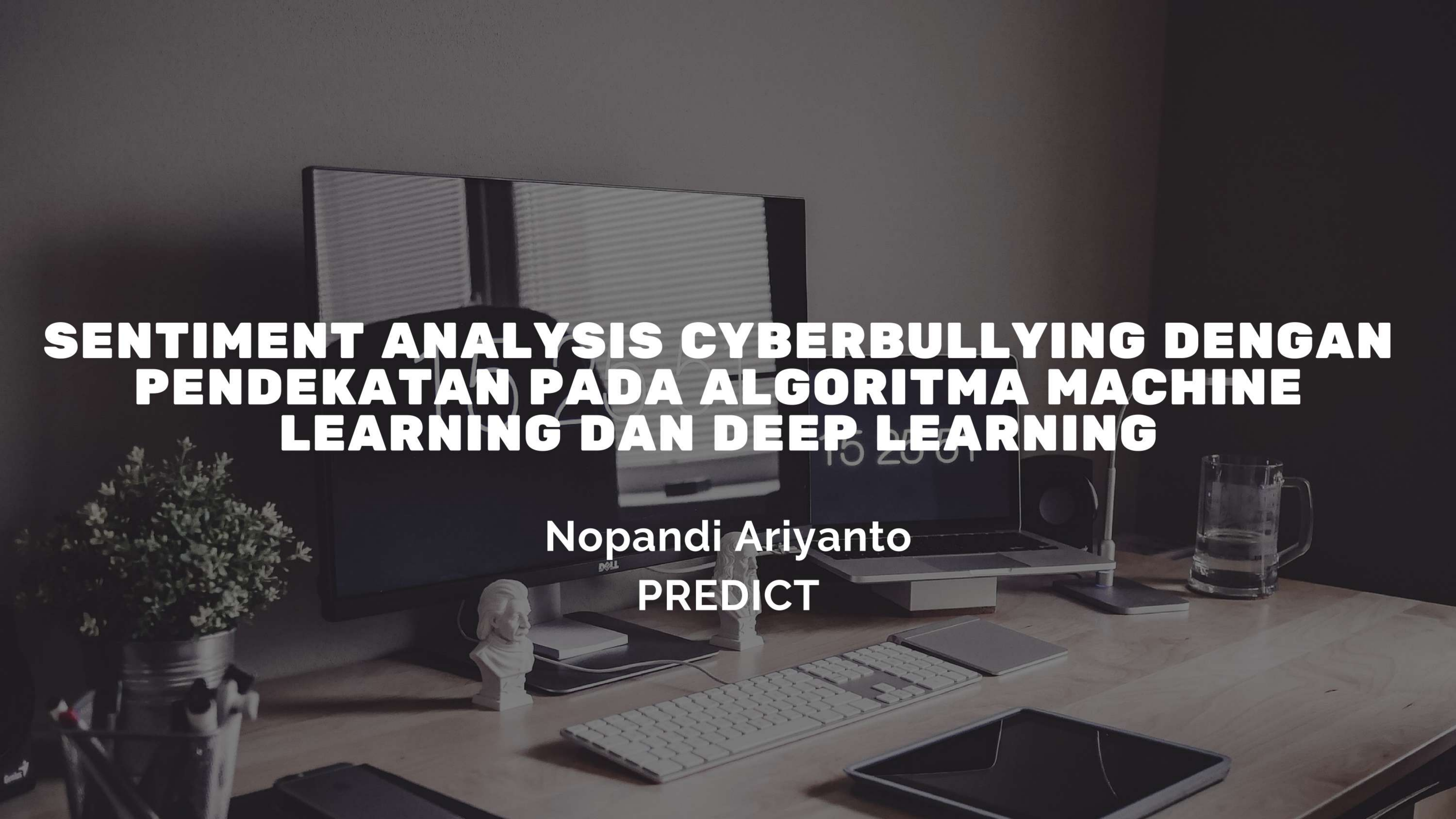
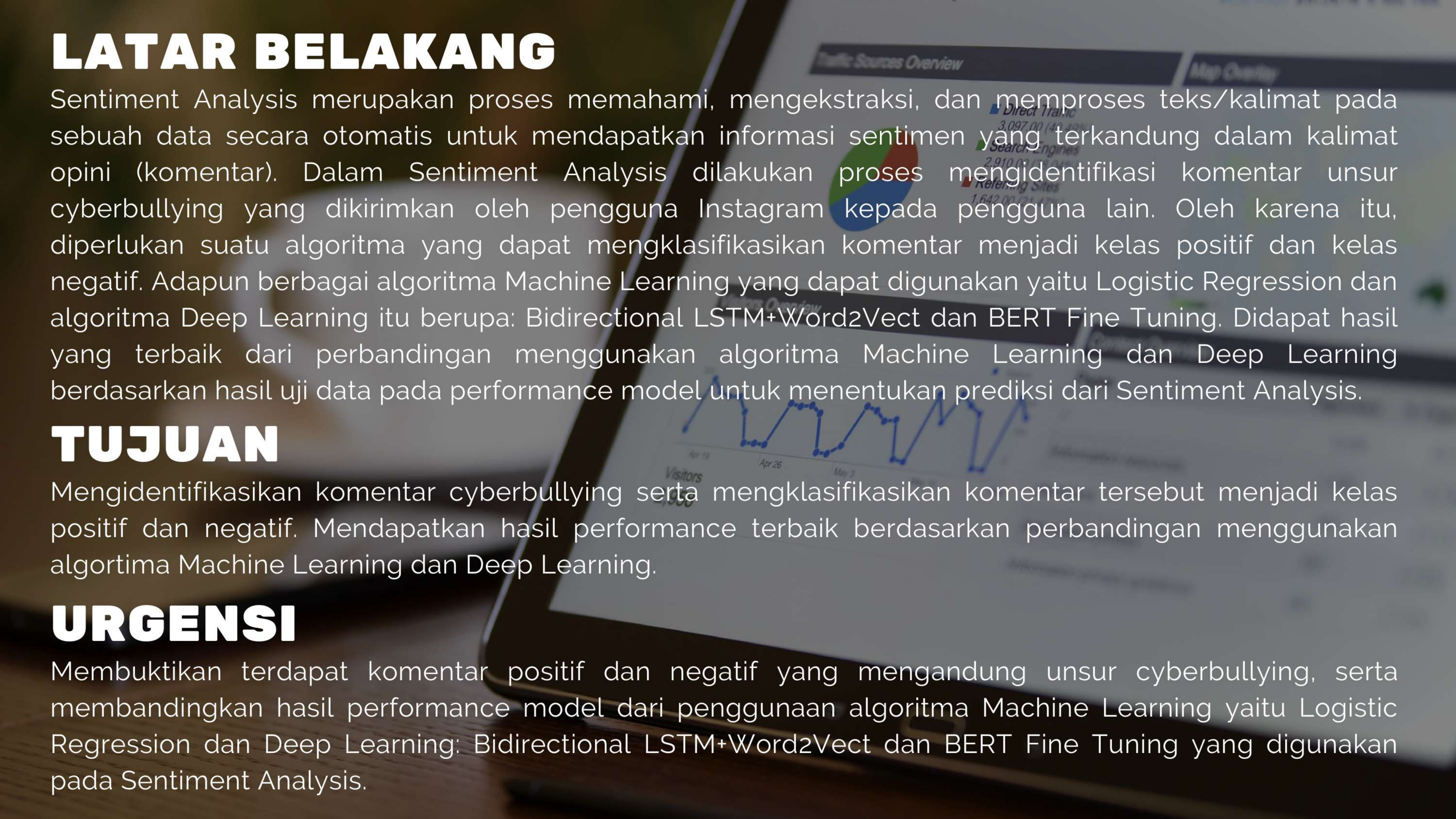


SENTIMENT ANALYSIS CYBERBULLYING DENGAN PENDEKATAN PADA ALGORITMA MACHINE LEARNING DAN DEEP LEARNING

Nopandi Ariyanto
PREDICT



LATAR BELAKANG

Sentiment Analysis merupakan proses memahami, mengekstraksi, dan memproses teks/kalimat pada sebuah data secara otomatis untuk mendapatkan informasi sentimen yang terkandung dalam kalimat opini (komentar). Dalam Sentiment Analysis dilakukan proses mengidentifikasi komentar unsur cyberbullying yang dikirimkan oleh pengguna Instagram kepada pengguna lain. Oleh karena itu, diperlukan suatu algoritma yang dapat mengklasifikasikan komentar menjadi kelas positif dan kelas negatif. Adapun berbagai algoritma Machine Learning yang dapat digunakan yaitu Logistic Regression dan algoritma Deep Learning itu berupa: Bidirectional LSTM+Word2Vect dan BERT Fine Tuning. Didapat hasil yang terbaik dari perbandingan menggunakan algoritma Machine Learning dan Deep Learning berdasarkan hasil uji data pada performance model untuk menentukan prediksi dari Sentiment Analysis.

TUJUAN

Mengidentifikasikan komentar cyberbullying serta mengklasifikasikan komentar tersebut menjadi kelas positif dan negatif. Mendapatkan hasil performance terbaik berdasarkan perbandingan menggunakan algortima Machine Learning dan Deep Learning.

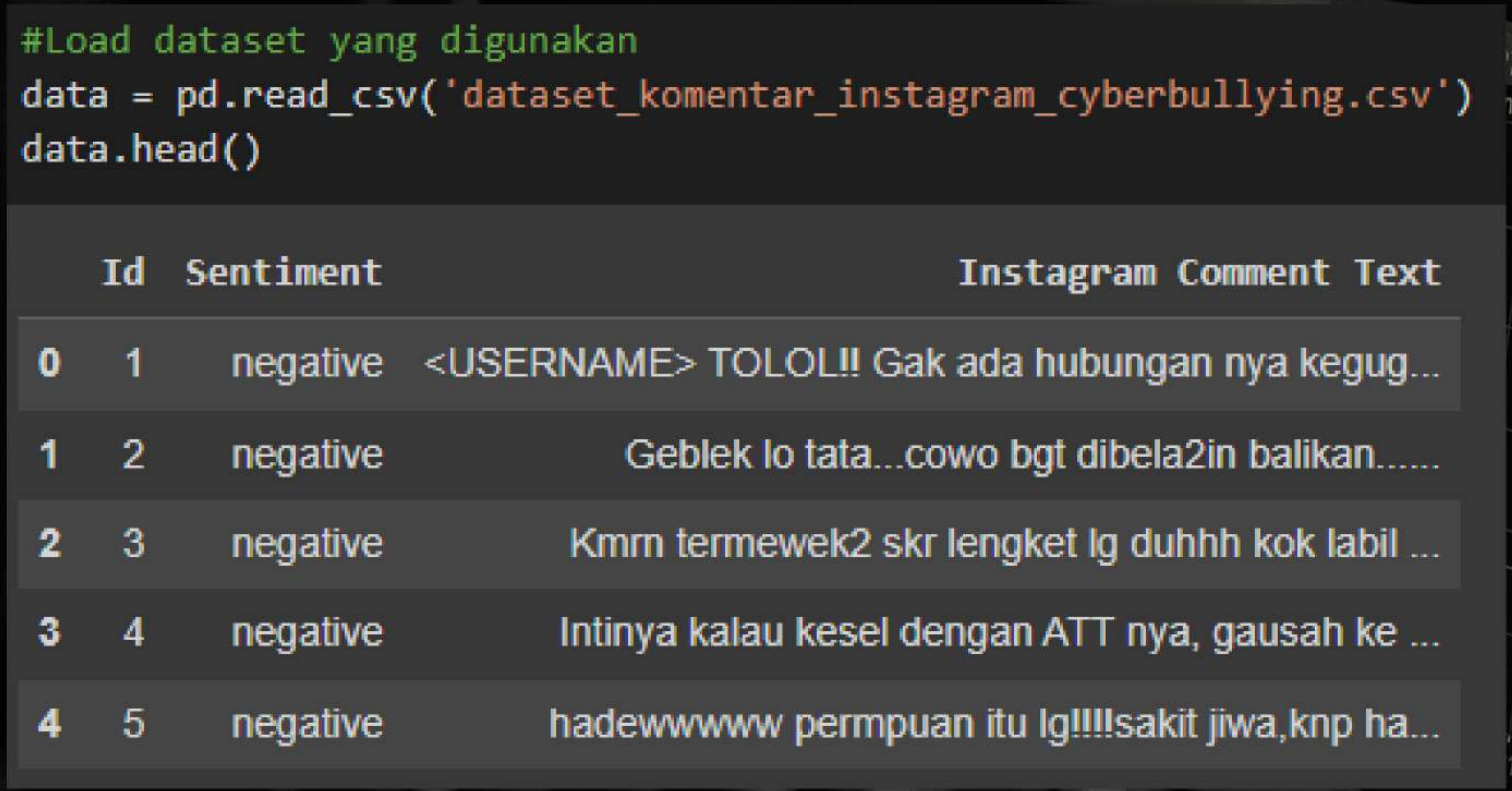
URGENSI

Membuktikan terdapat komentar positif dan negatif yang mengandung unsur cyberbullying, serta membandingkan hasil performance model dari penggunaan algoritma Machine Learning yaitu Logistic Regression dan Deep Learning: Bidirectional LSTM+Word2Vect dan BERT Fine Tuning yang digunakan pada Sentiment Analysis.

DATA

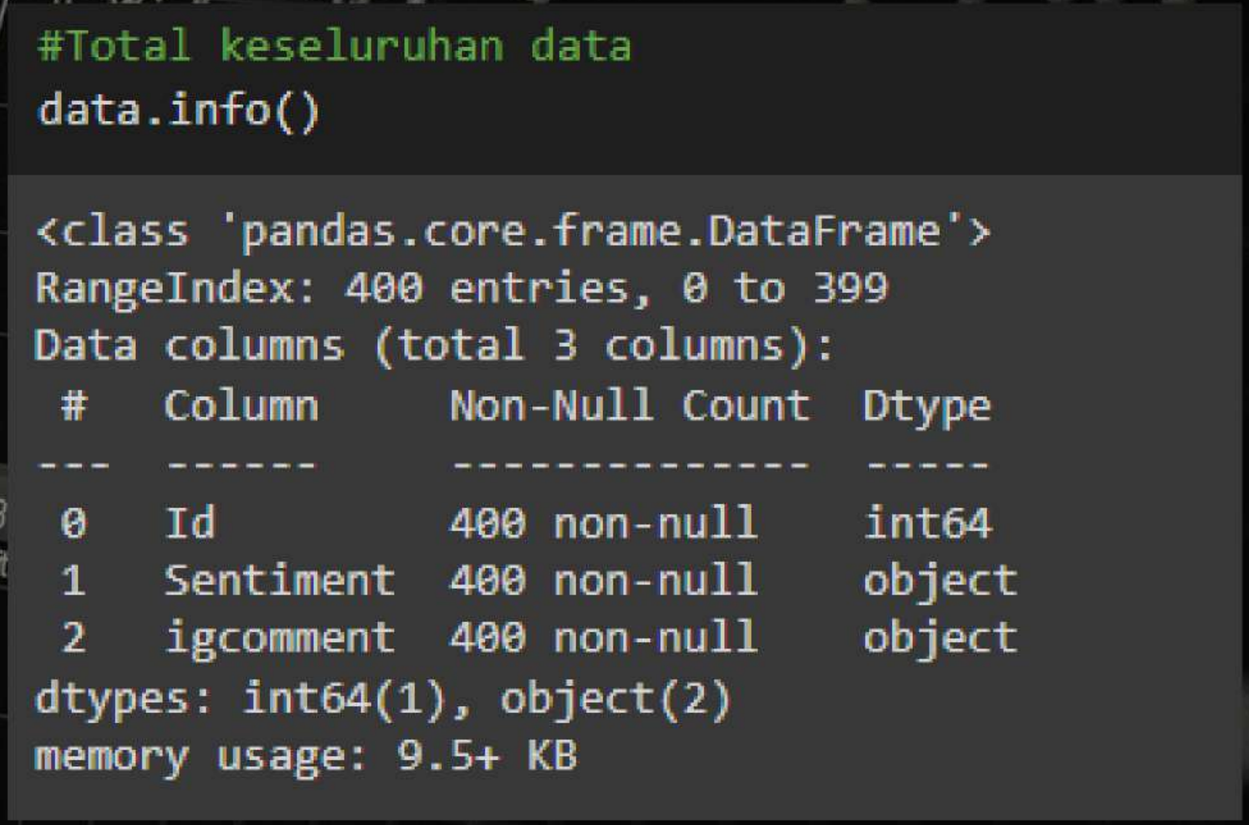
Dataset: **dataset_komentar_instagram_cyberbullying.csv**
Sumber: https://raw.githubusercontent.com/rizalespe/DatasetSentimen-Analisis-Bahasa-Indonesia/master/dataset_komentar_instagram_cyberbullying.csv

VARIABLE



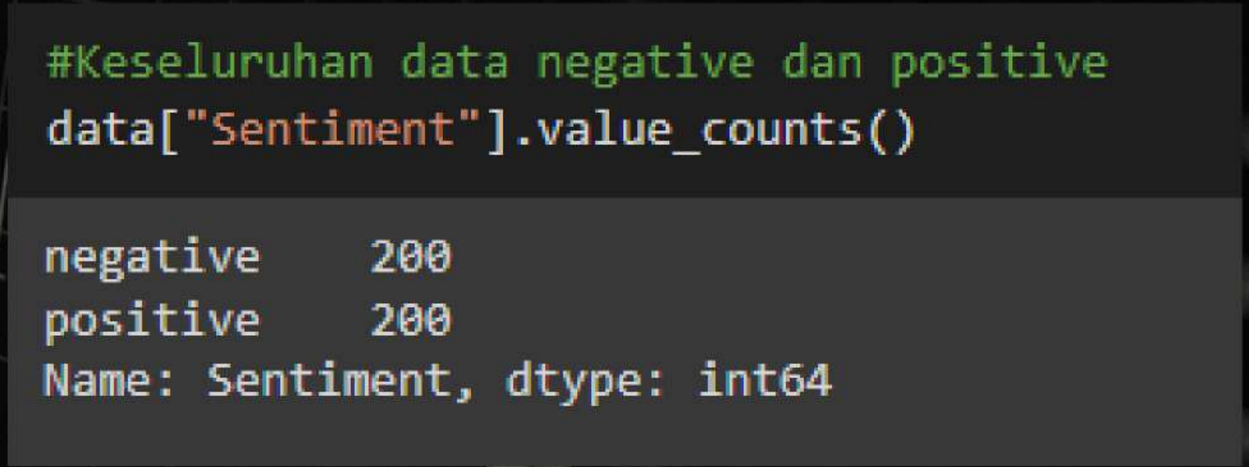
```
#Load dataset yang digunakan
data = pd.read_csv('dataset_komentar_instagram_cyberbullying.csv')
data.head()
```

	Id	Sentiment	Instagram Comment Text
0	1	negative	<USERNAME> TOLOL!! Gak ada hubungan nya kegug...
1	2	negative	Geblek lo tata...cowo bgt dibela2in balikan......
2	3	negative	Kmrn termewek2 skr lengket lg duhhh kok labil ...
3	4	negative	Intinya kalau kesel dengan ATT nya, gausah ke ...
4	5	negative	hadewwwww permpuan itu lg!!!!sakit jiwa,knp ha...



```
#Total keseluruhan data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Id         400 non-null    int64
 1   Sentiment  400 non-null    object
 2   igcomment  400 non-null    object
dtypes: int64(1), object(2)
memory usage: 9.5+ KB
```

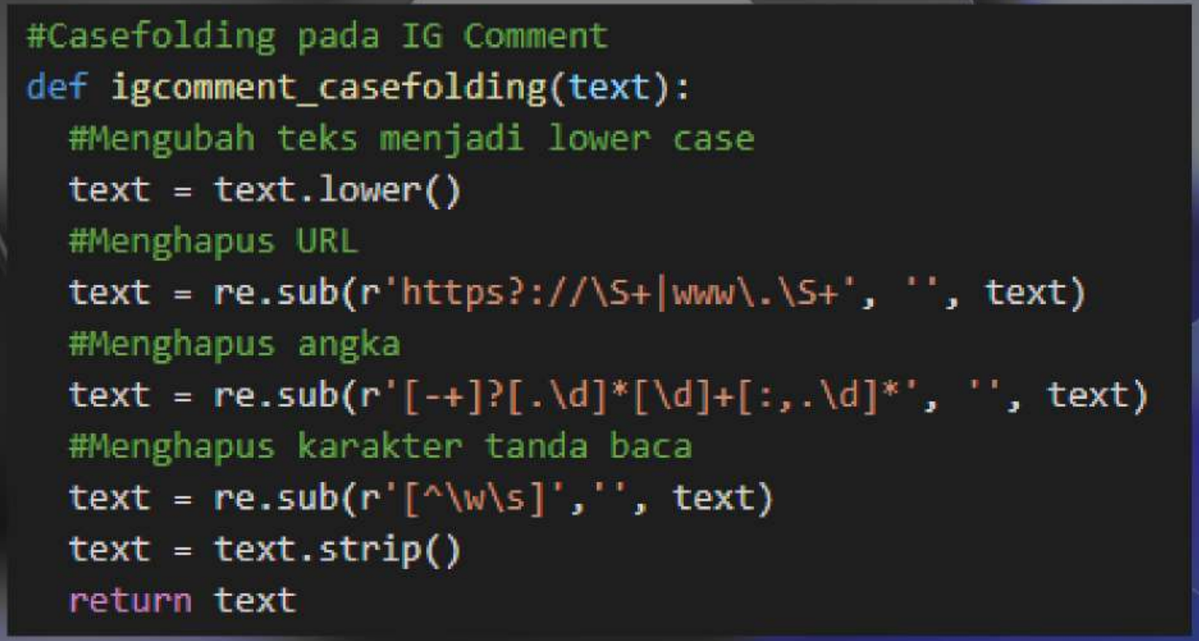


```
#Keseluruhan data negative dan positive
data["Sentiment"].value_counts()
```

```
negative    200
positive    200
Name: Sentiment, dtype: int64
```

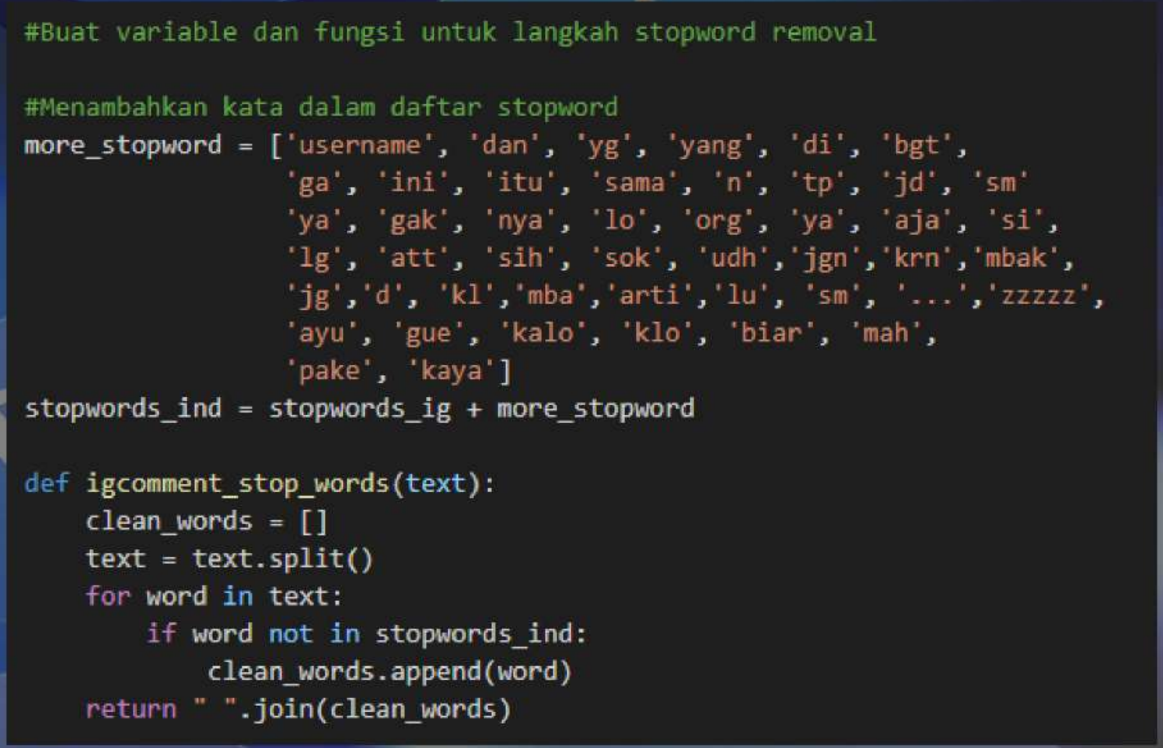
PREPROCESSING DATA

CASEFOLDING



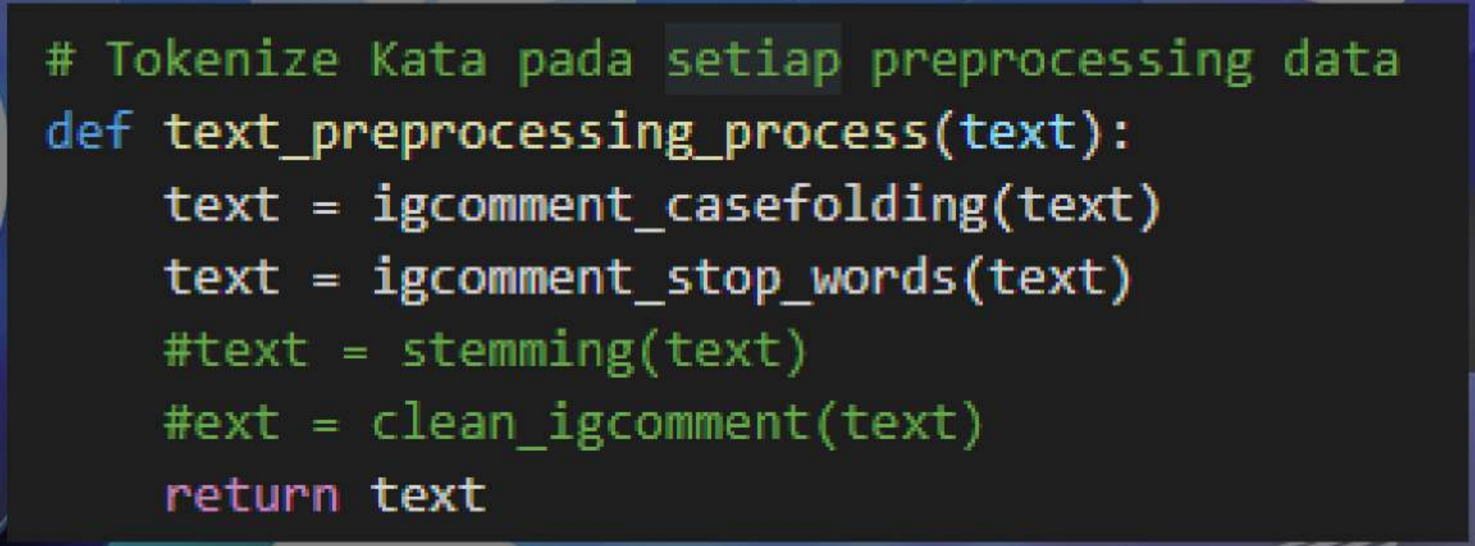
```
#Casefolding pada IG Comment
def igcomment_casefolding(text):
  #Mengubah teks menjadi lower case
  text = text.lower()
  #Menghapus URL
  text = re.sub(r'https?://\S+|www\.\S+', '', text)
  #Menghapus angka
  text = re.sub(r'[-+]?[.\d]*[\d]+[:,.\d]*', '', text)
  #Menghapus karakter tanda baca
  text = re.sub(r'[^\w\s]','', text)
  text = text.strip()
  return text
```

STOPWORD REMOVAL



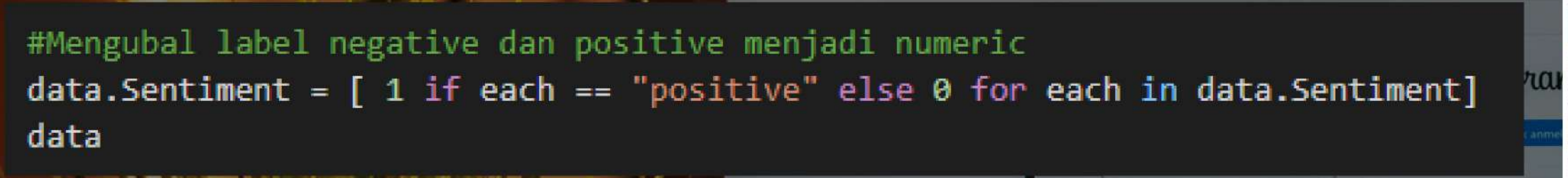
```
#Buat variable dan fungsi untuk langkah stopword removal

#Menambahkan kata dalam daftar stopword
more_stopword = ['username', 'dan', 'yg', 'yang', 'di', 'bgt',
                 'ga', 'ini', 'itu', 'sama', 'n', 'tp', 'jd', 'sm'
                 'ya', 'gak', 'nya', 'lo', 'org', 'ya', 'aja', 'si',
                 'lg', 'att', 'sih', 'sok', 'udh','jgn','krn','mbak',
                 'jg','d', 'kl','mba','arti','lu', 'sm', '...','zzzzz',
                 'ayu', 'gue', 'kalo', 'klo', 'biar', 'mah',
                 'pake', 'kaya']
stopwords_ind = stopwords_ig + more_stopword

def igcomment_stop_words(text):
    clean_words = []
    text = text.split()
    for word in text:
        if word not in stopwords_ind:
            clean_words.append(word)
    return " ".join(clean_words)
```

TEXT PREPROCESSING

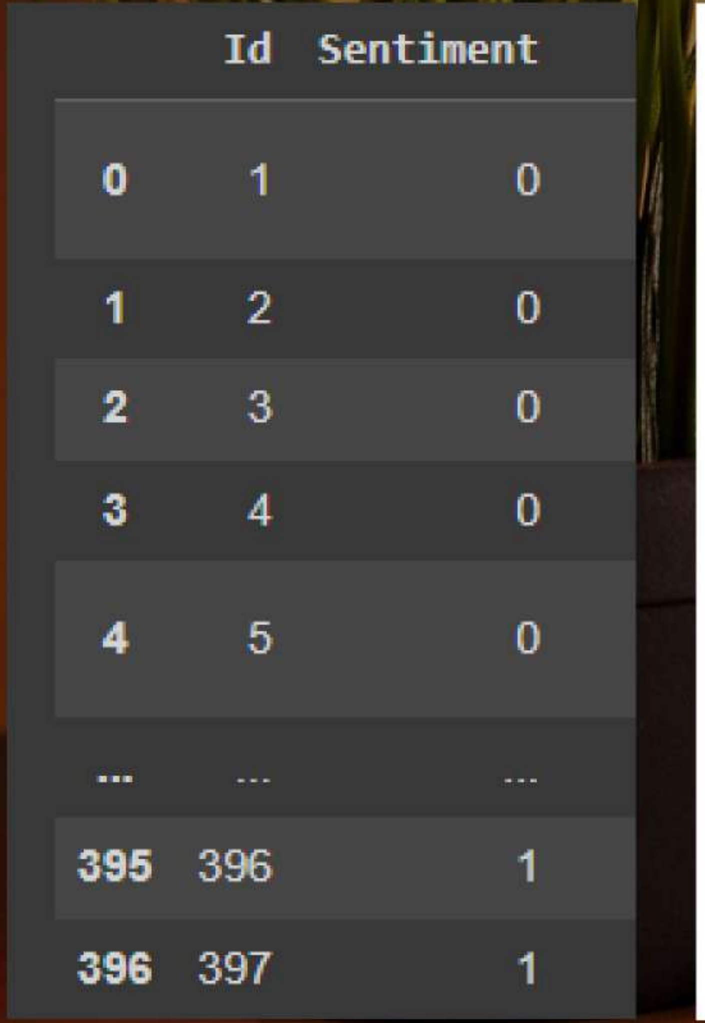


```
# Tokenize Kata pada setiap preprocessing data
def text_preprocessing_process(text):
    text = igcomment_casefolding(text)
    text = igcomment_stop_words(text)
    #text = stemming(text)
    #ext = clean_igcomment(text)
    return text
```

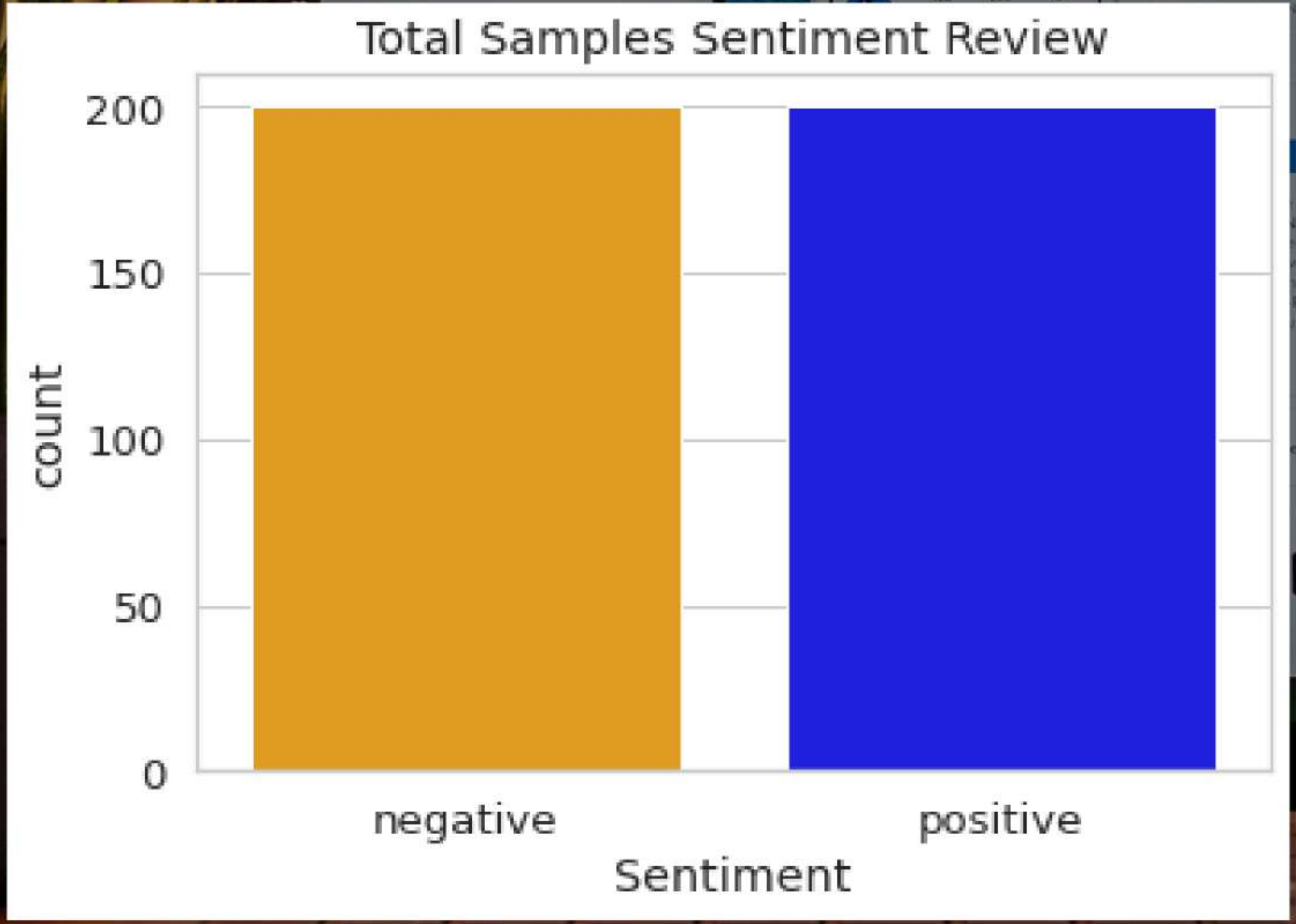
KLASIFIKASI DATA



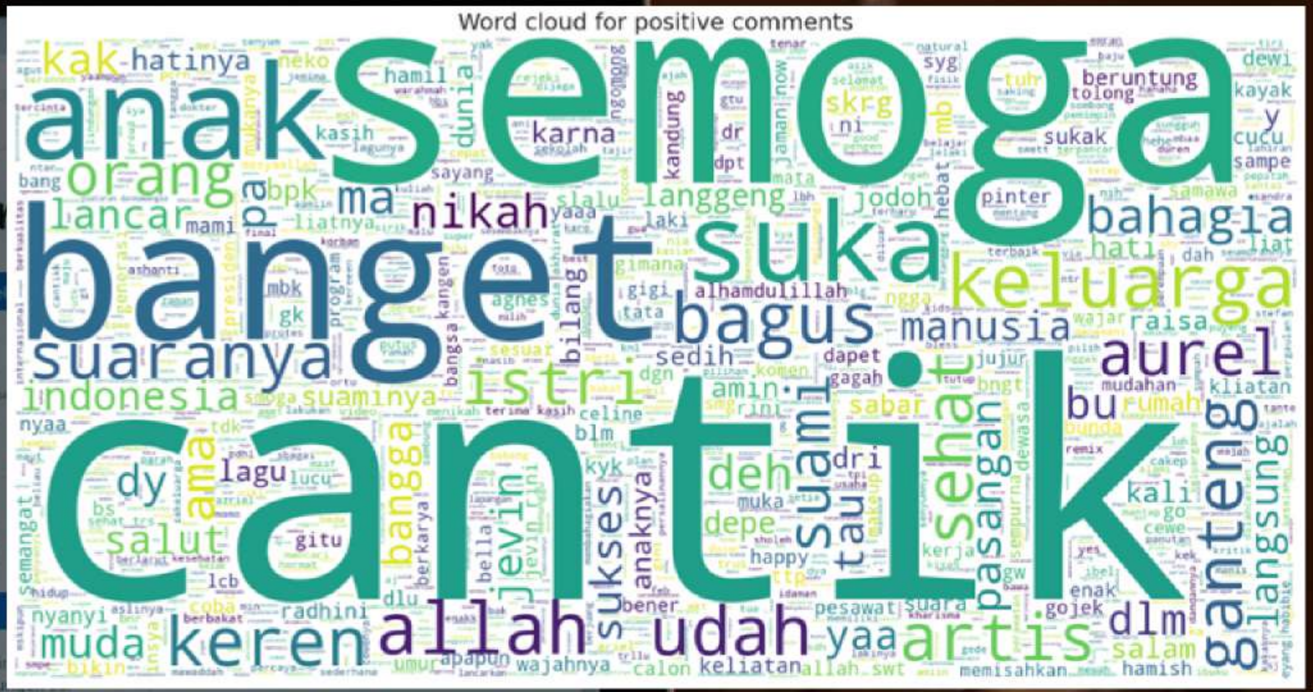
```
#Mengubal label negative dan positive menjadi numeric
data.Sentiment = [ 1 if each == "positive" else 0 for each in data.Sentiment]
data
```



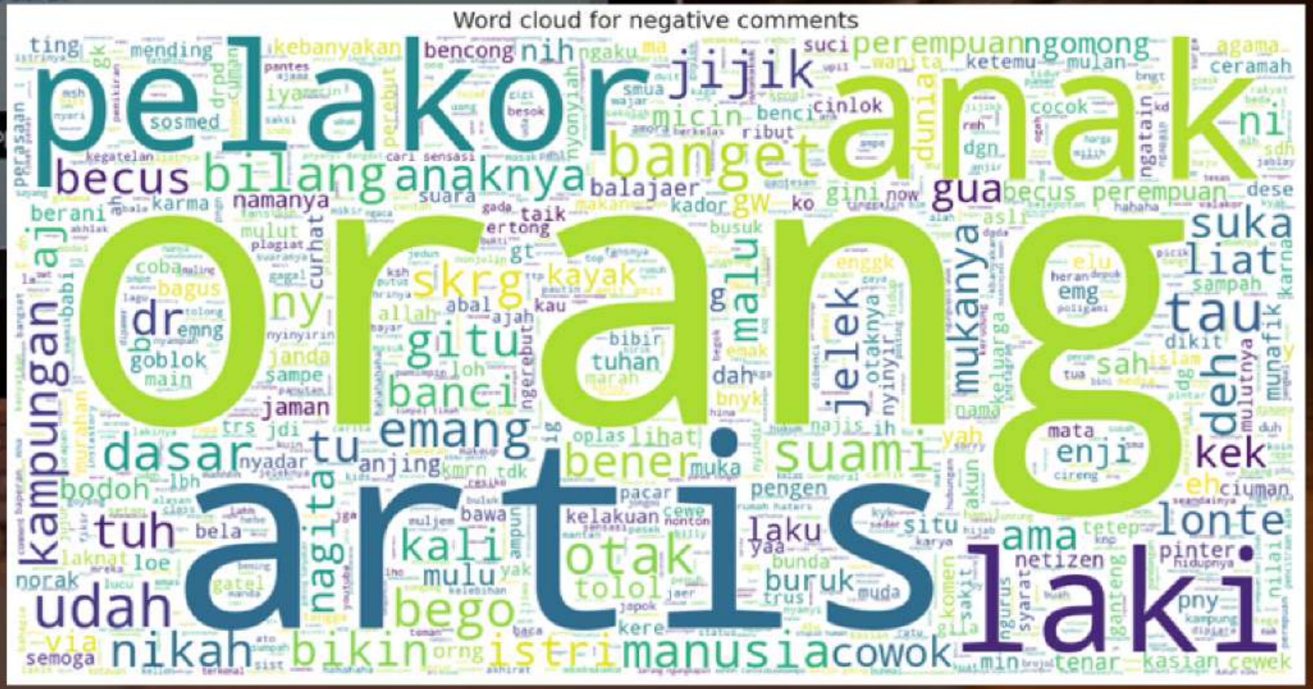
	Id	Sentiment
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0
...	...	...
395	396	1
396	397	1



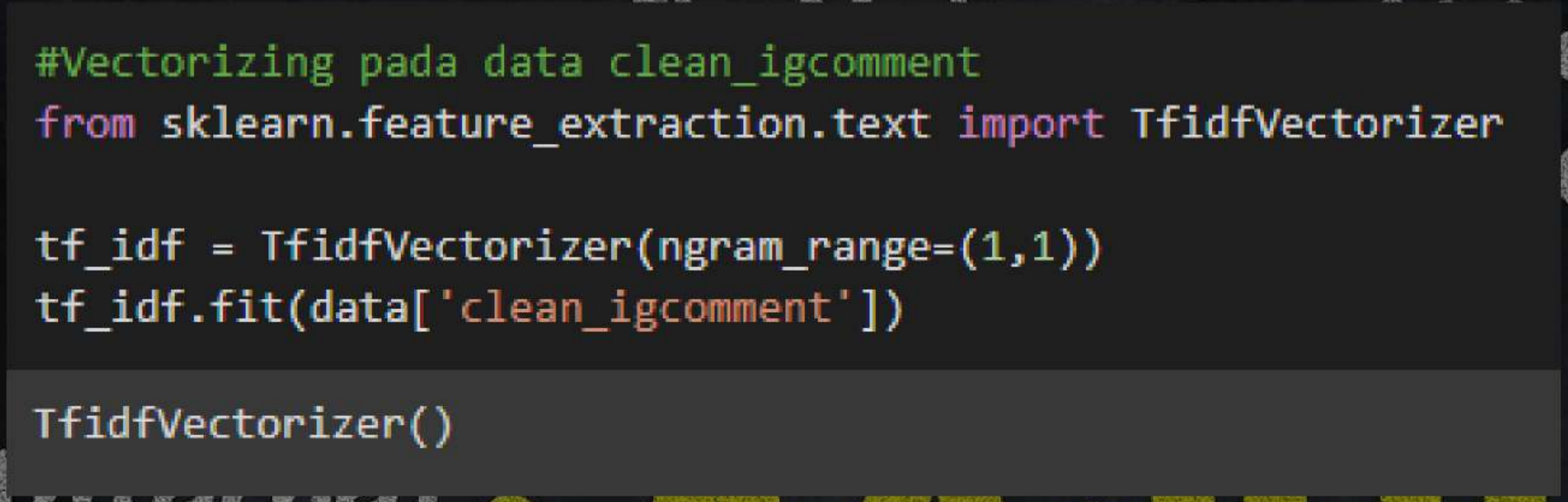
POSITIVE



NEGATIVE

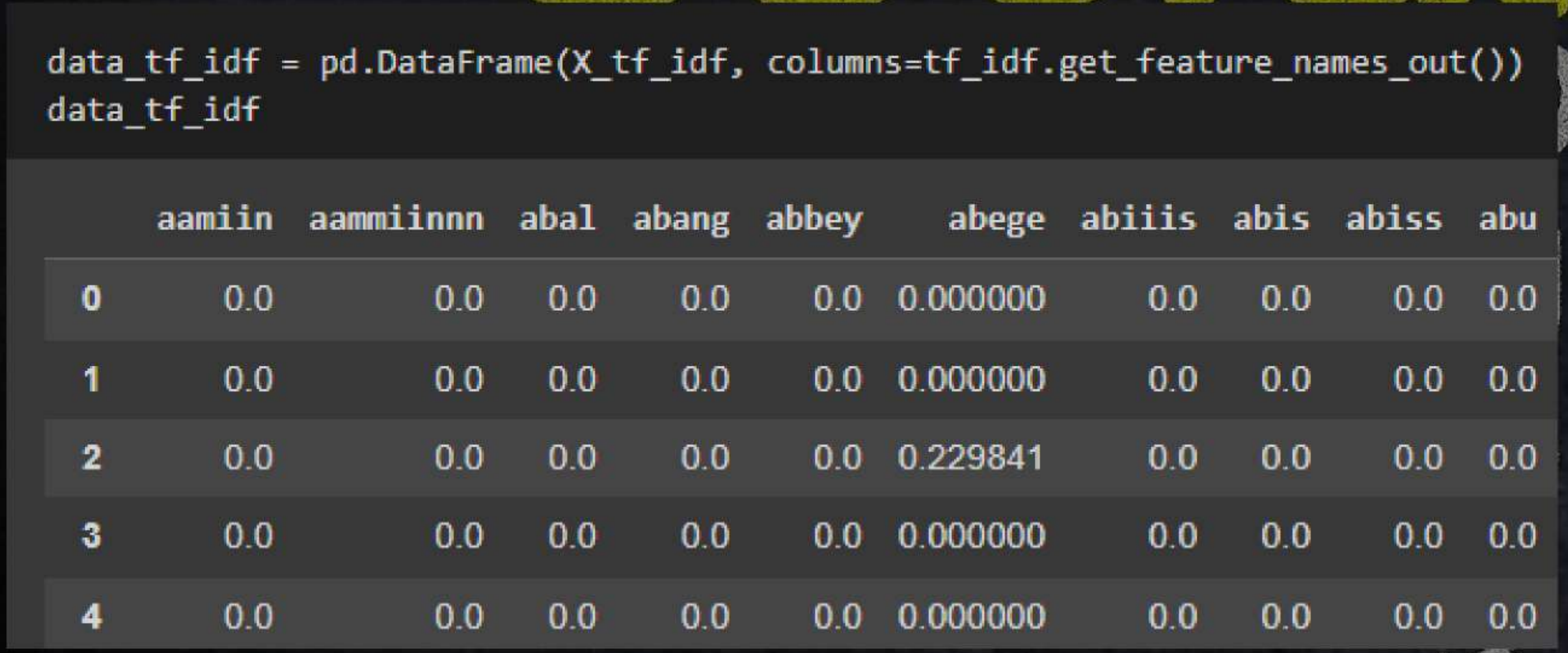


FEATURE EXTRACTION



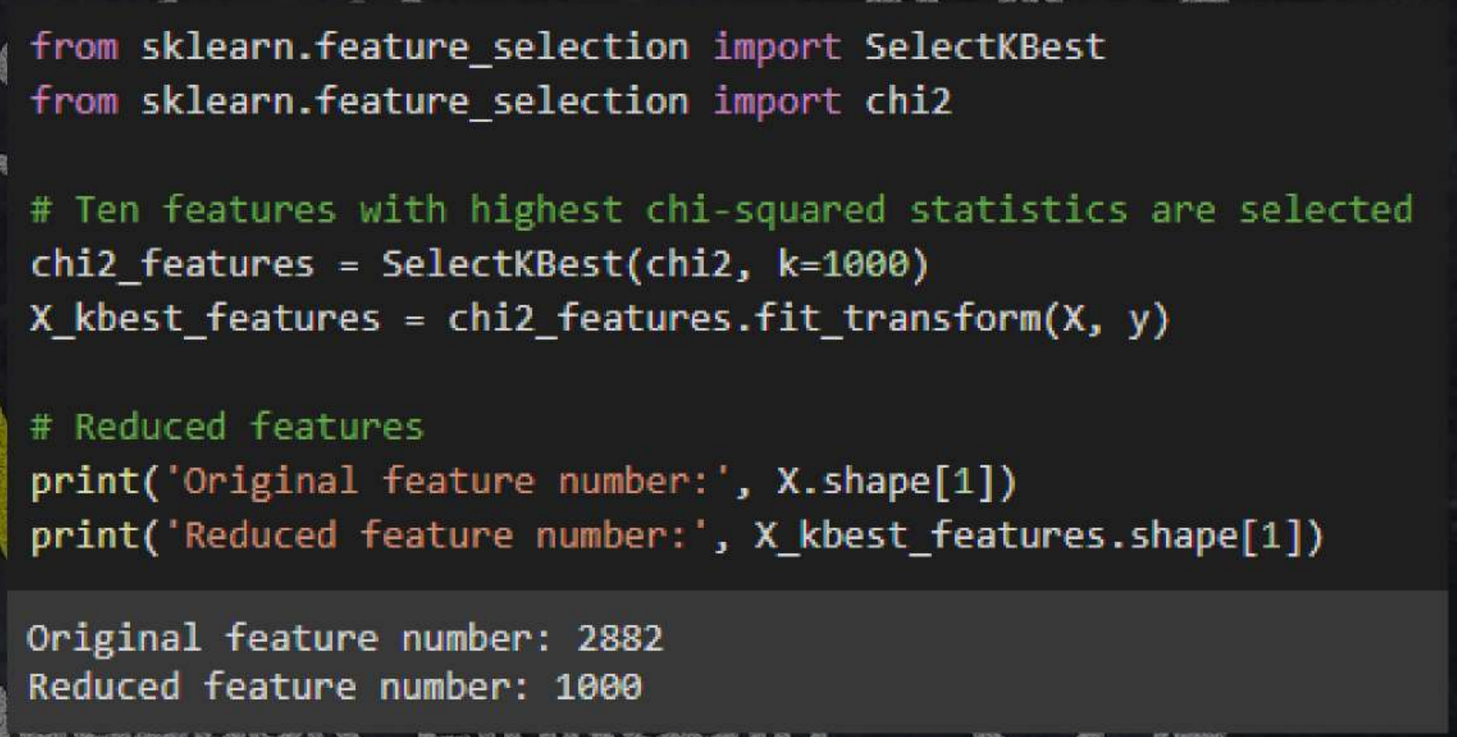
```
#Vectorizing pada data clean_igcomment
from sklearn.feature_extraction.text import TfidfVectorizer

tf_idf = TfidfVectorizer(ngram_range=(1,1))
tf_idf.fit(data['clean_igcomment'])
```

```
TfidfVectorizer()
```



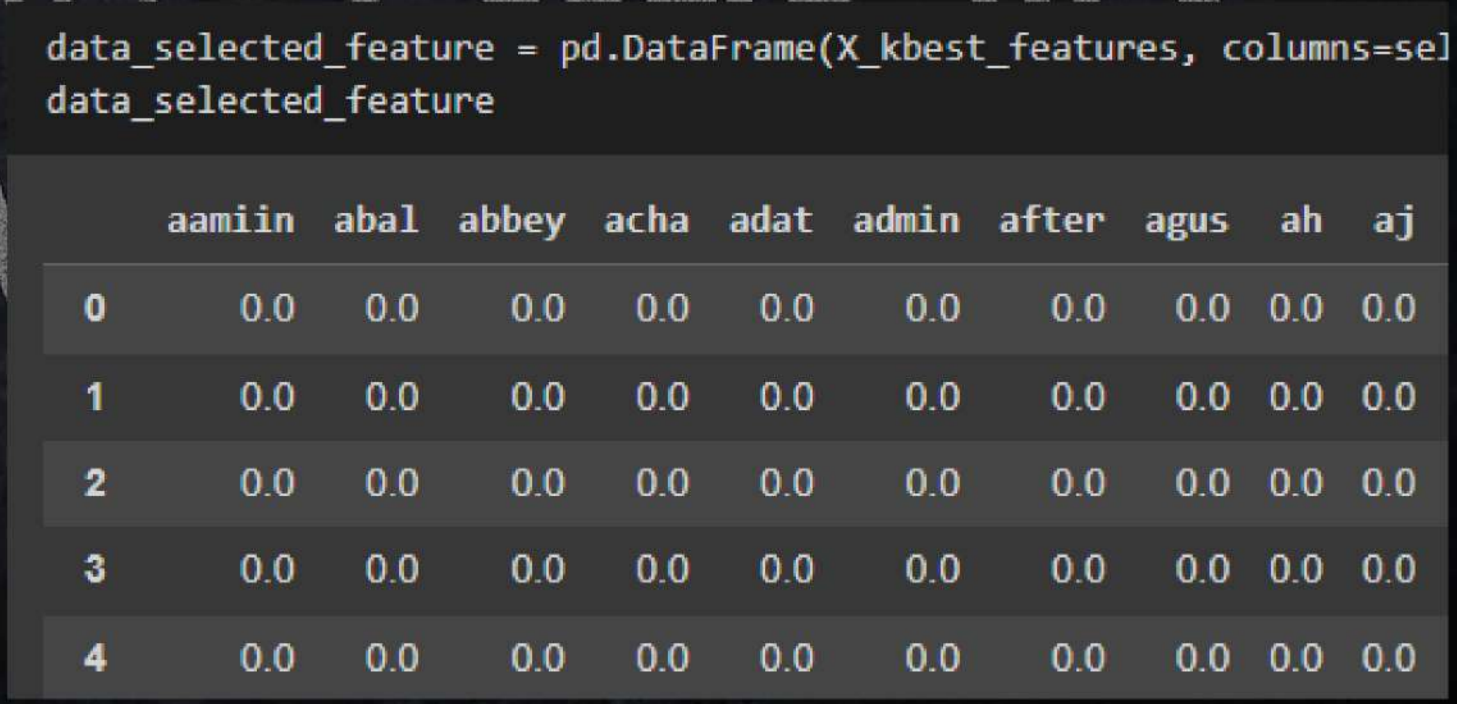
```
data_tf_idf = pd.DataFrame(X_tf_idf, columns=tf_idf.get_feature_names_out())
data_tf_idf
```

	aamiin	aammiinnn	abal	abang	abbey	abege	abiiis	abis	abiss	abu
0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.229841	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0



```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# Ten features with highest chi-squared statistics are selected
chi2_features = SelectKBest(chi2, k=1000)
X_kbest_features = chi2_features.fit_transform(X, y)

# Reduced features
print('Original feature number:', X.shape[1])
print('Reduced feature number:', X_kbest_features.shape[1])
```

```
Original feature number: 2882
Reduced feature number: 1000
```

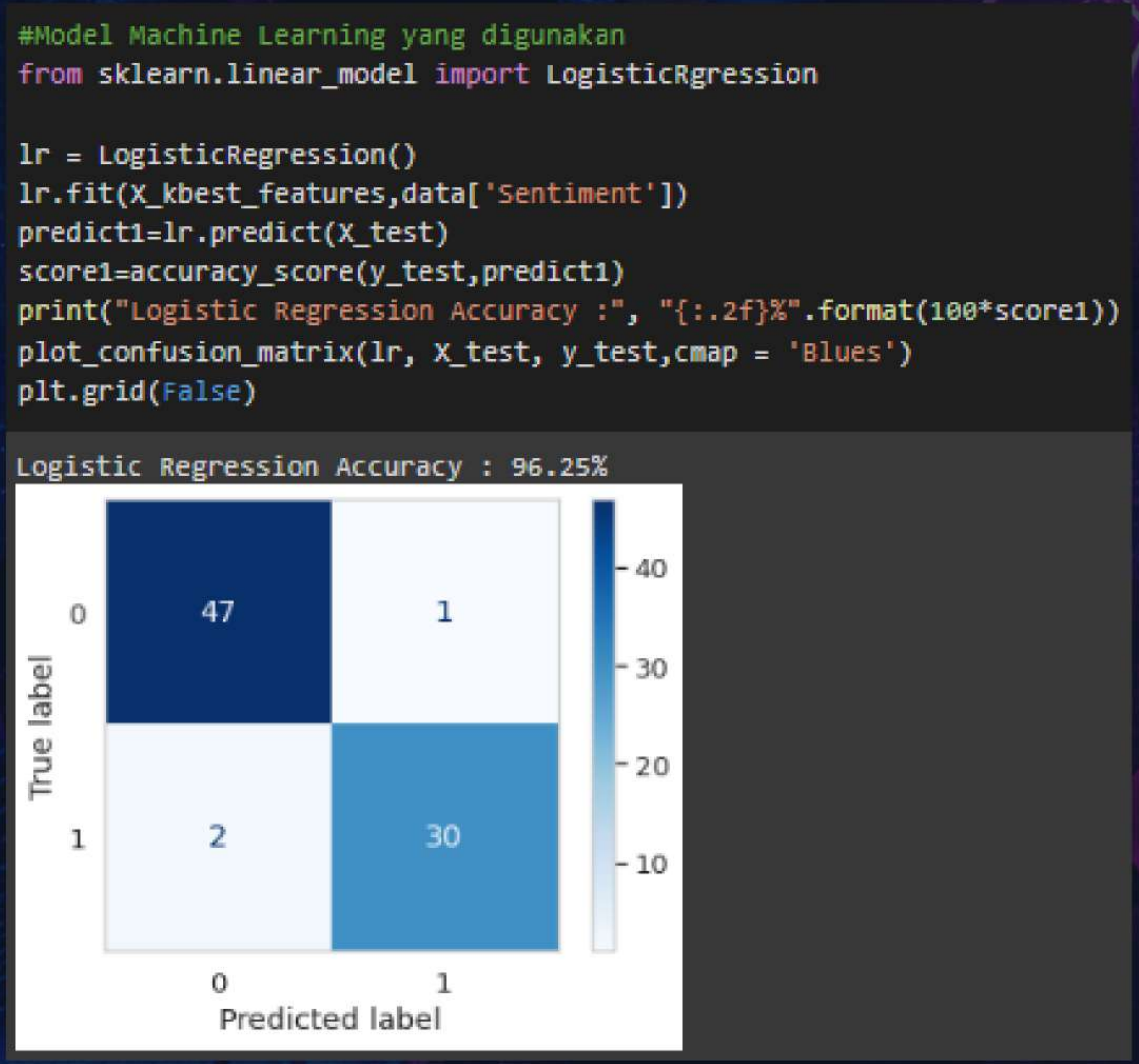


```
data_selected_feature = pd.DataFrame(X_kbest_features, columns=sel
data_selected_feature
```

	aamiin	abal	abbey	acha	adat	admin	after	agus	ah	aj
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

MODEL

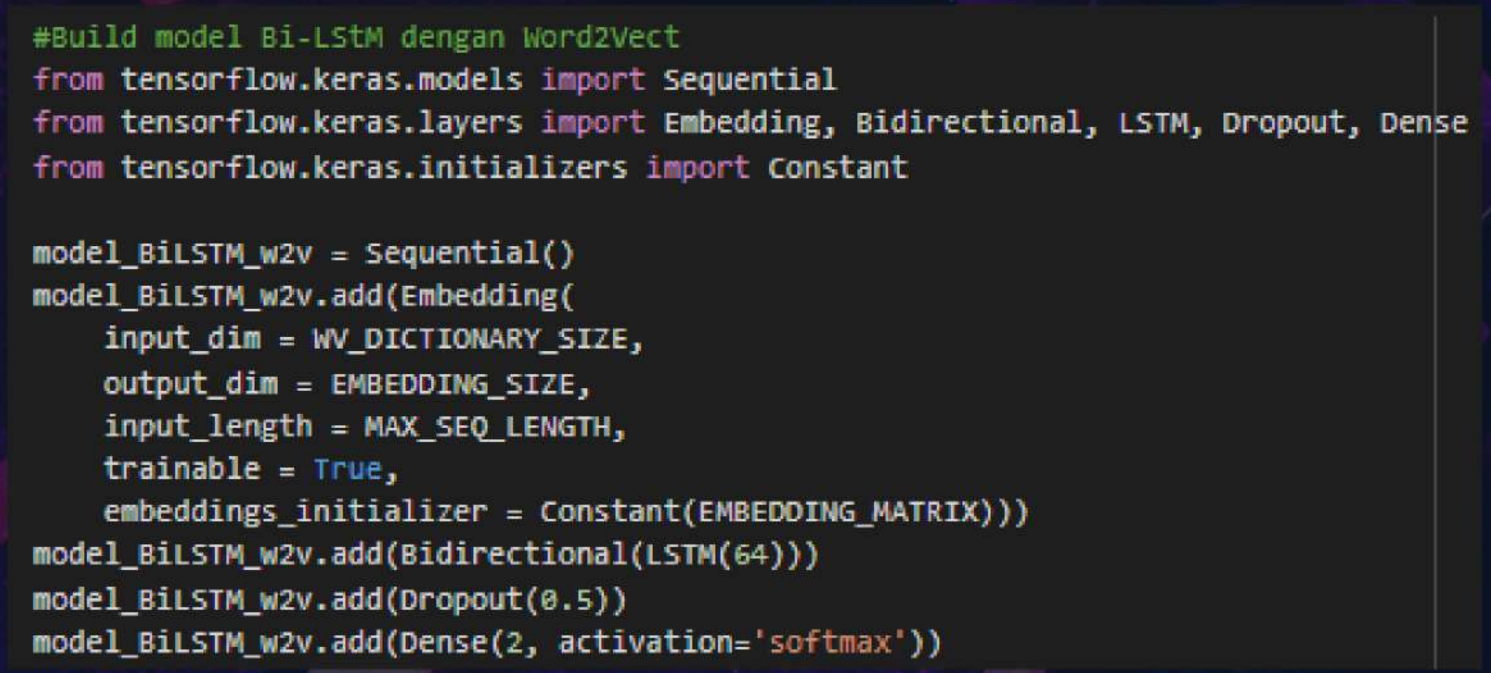
LOGISTIC REGRESSION



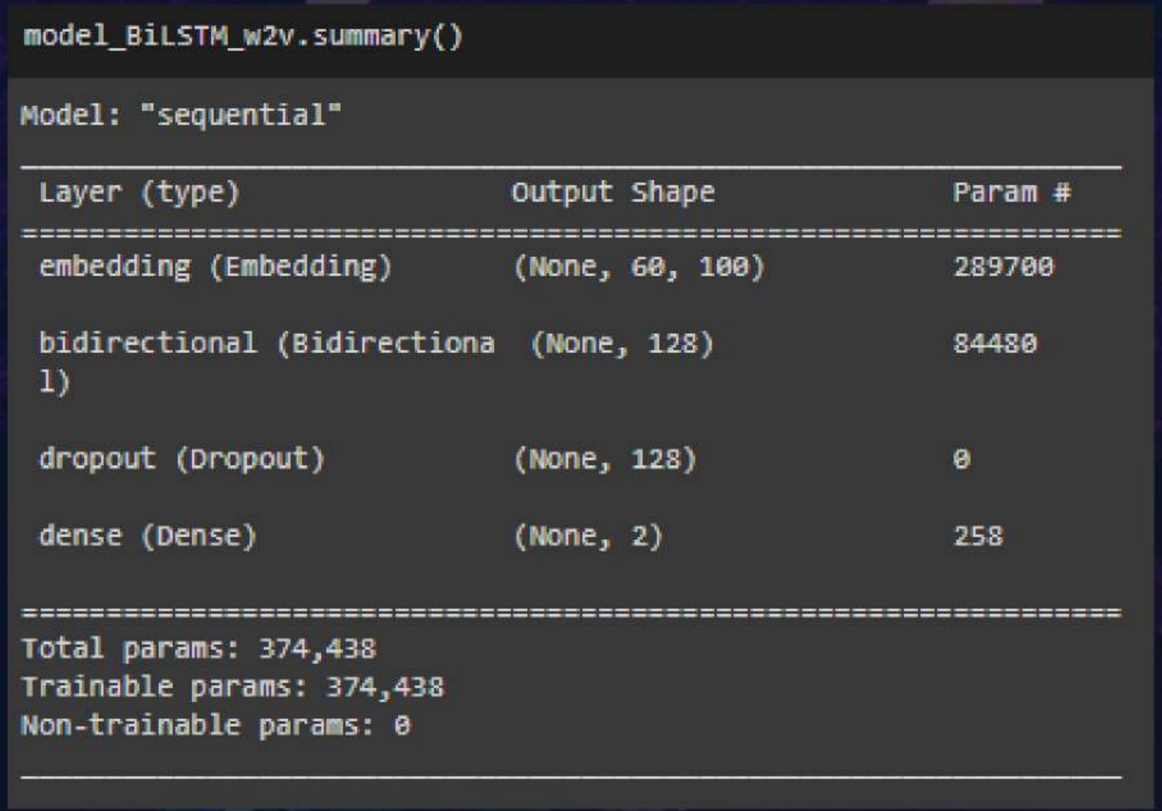
```
#Model Machine Learning yang digunakan
from sklearn.linear_model import LogisticRgression

lr = LogisticRegression()
lr.fit(X_kbest_features,data['Sentiment'])
predict1=lr.predict(X_test)
score1=accuracy_score(y_test,predict1)
print("Logistic Regression Accuracy :", "{:.2f}%".format(100*score1))
plot_confusion_matrix(lr, X_test, y_test,cmap = 'Blues')
plt.grid(False)
```

```
Logistic Regression Accuracy : 96.25%
```

BI-LSTM + WORD2VECT



```
#Build model Bi-LStM dengan Word2Vect
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dropout, Dense
from tensorflow.keras.initializers import Constant

model_BiLSTM_w2v = Sequential()
model_BiLSTM_w2v.add(Embedding(
    input_dim = WV_DICTIONARY_SIZE,
    output_dim = EMBEDDING_SIZE,
    input_length = MAX_SEQ_LENGTH,
    trainable = True,
    embeddings_initializer = Constant(EMBEDDING_MATRIX)))
model_BiLSTM_w2v.add(Bidirectional(LSTM(64)))
model_BiLSTM_w2v.add(Dropout(0.5))
model_BiLSTM_w2v.add(Dense(2, activation='softmax'))
```

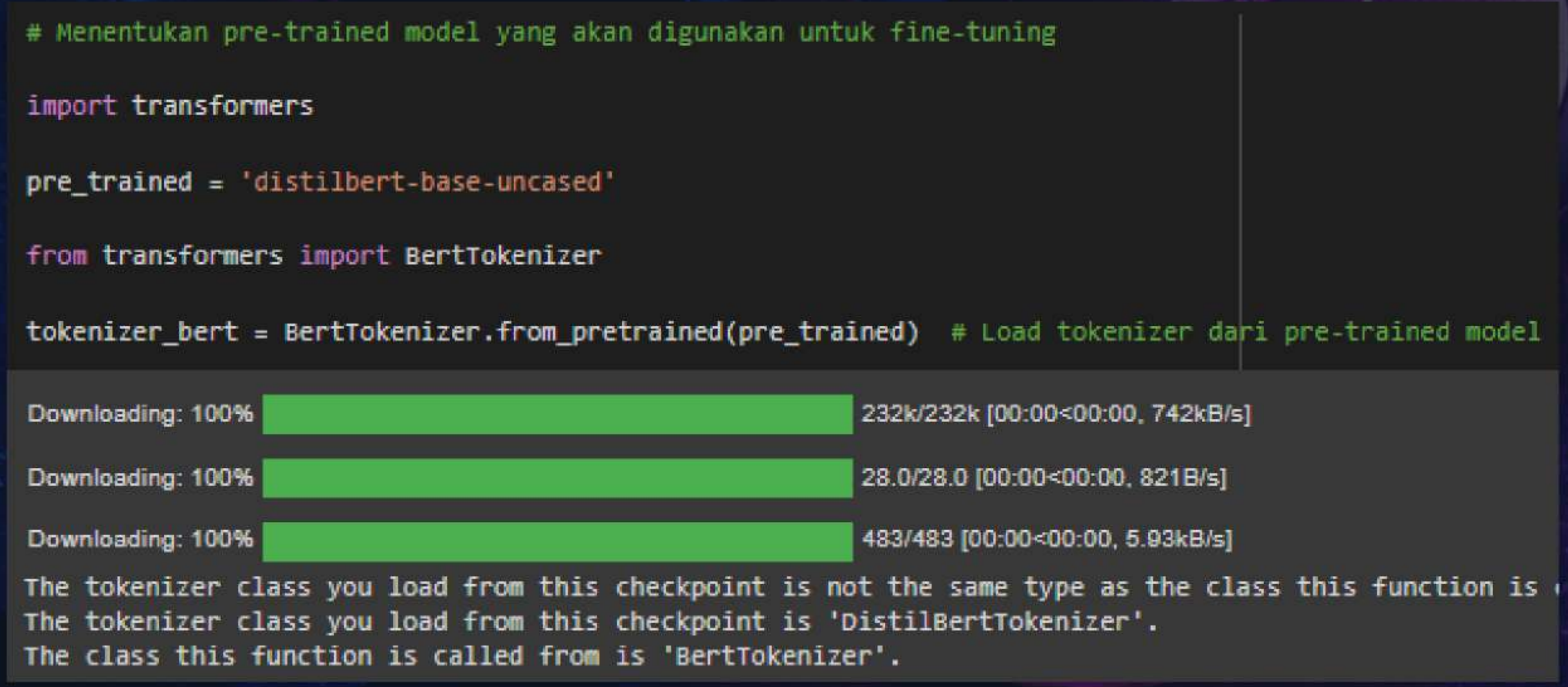


```
model_BiLSTM_w2v.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 60, 100)           289700

 bidirectional (Bidirectiona  (None, 128)              84480
 l)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 2)                 258

=================================================================
Total params: 374,438
Trainable params: 374,438
Non-trainable params: 0
_________________________________________________________________
```

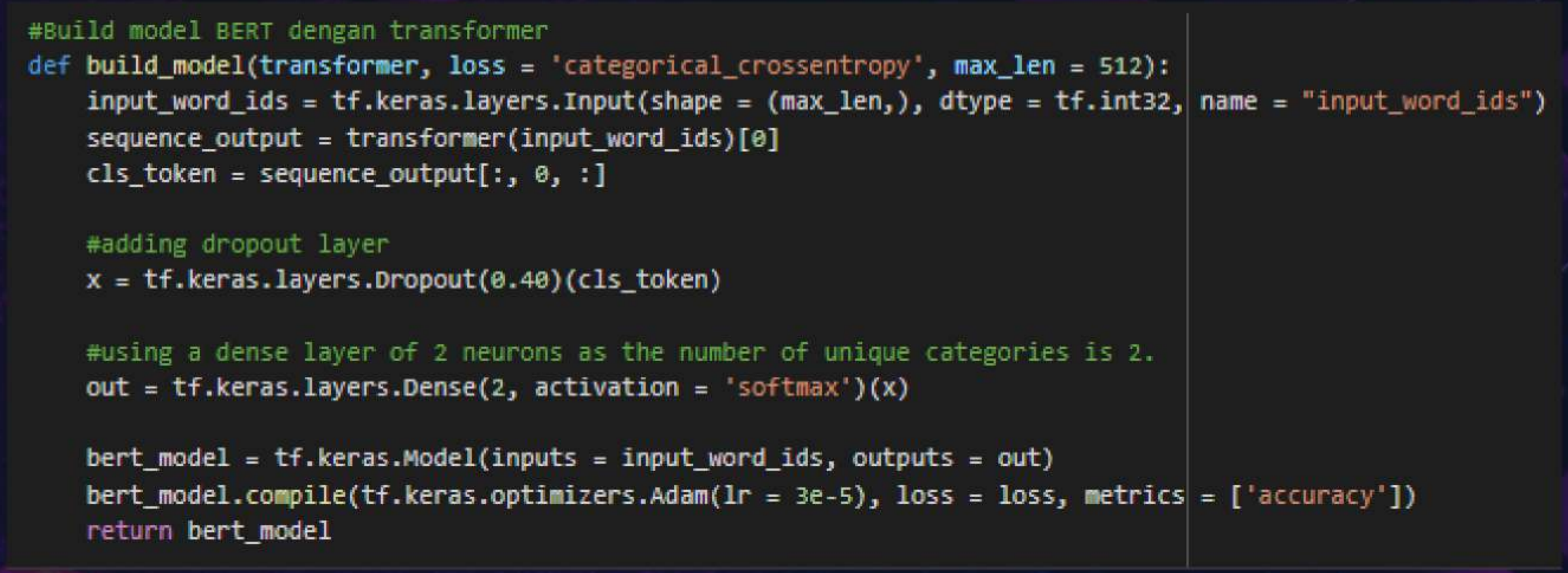
MODEL

BERT FINE TUNING

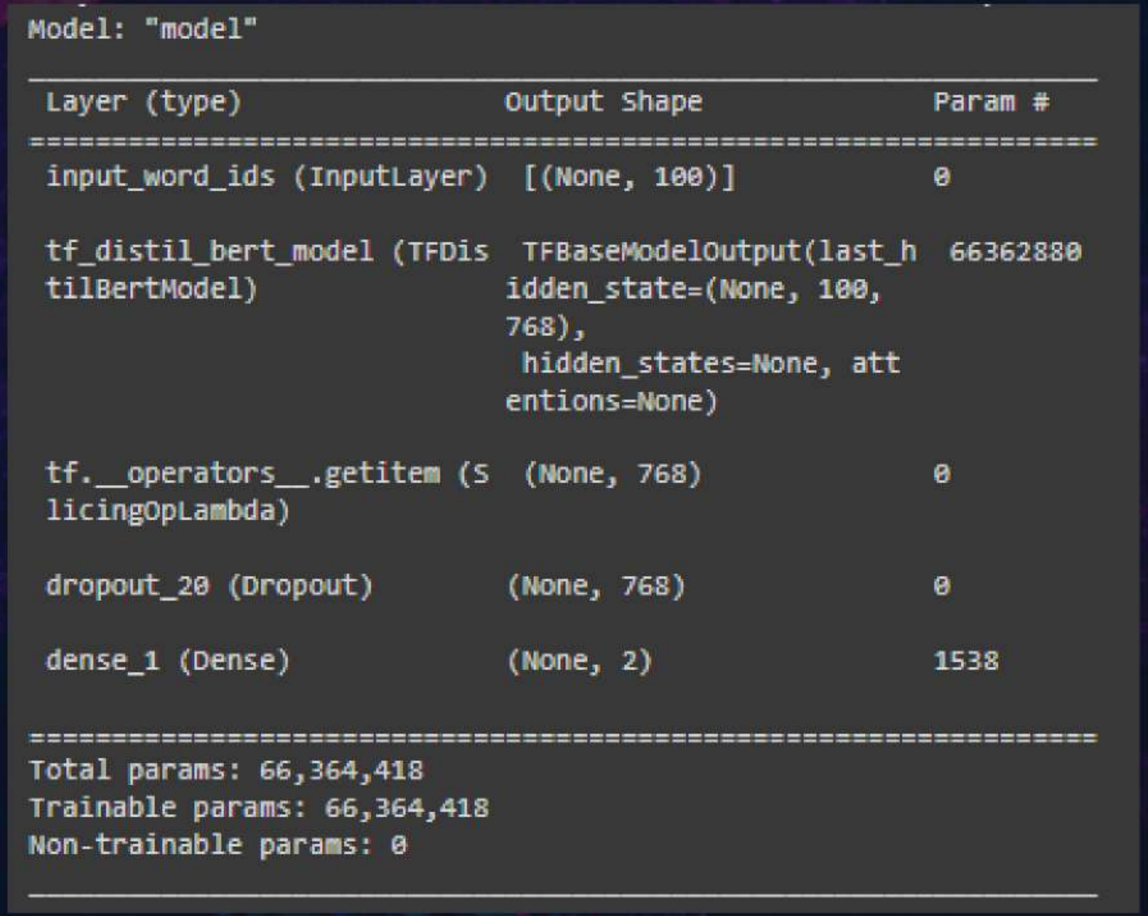


```
# Menentukan pre-trained model yang akan digunakan untuk fine-tuning

import transformers

pre_trained = 'distilbert-base-uncased'

from transformers import BertTokenizer

tokenizer_bert = BertTokenizer.from_pretrained(pre_trained)  # Load tokenizer dari pre-trained model
```

```
Downloading: 100%  232k/232k [00:00<00:00, 742kB/s]
Downloading: 100%  28.0/28.0 [00:00<00:00, 821B/s]
Downloading: 100%  483/483 [00:00<00:00, 5.93kB/s]
The tokenizer class you load from this checkpoint is not the same type as the class this function is
The tokenizer class you load from this checkpoint is 'DistilBertTokenizer'.
The class this function is called from is 'BertTokenizer'.
```



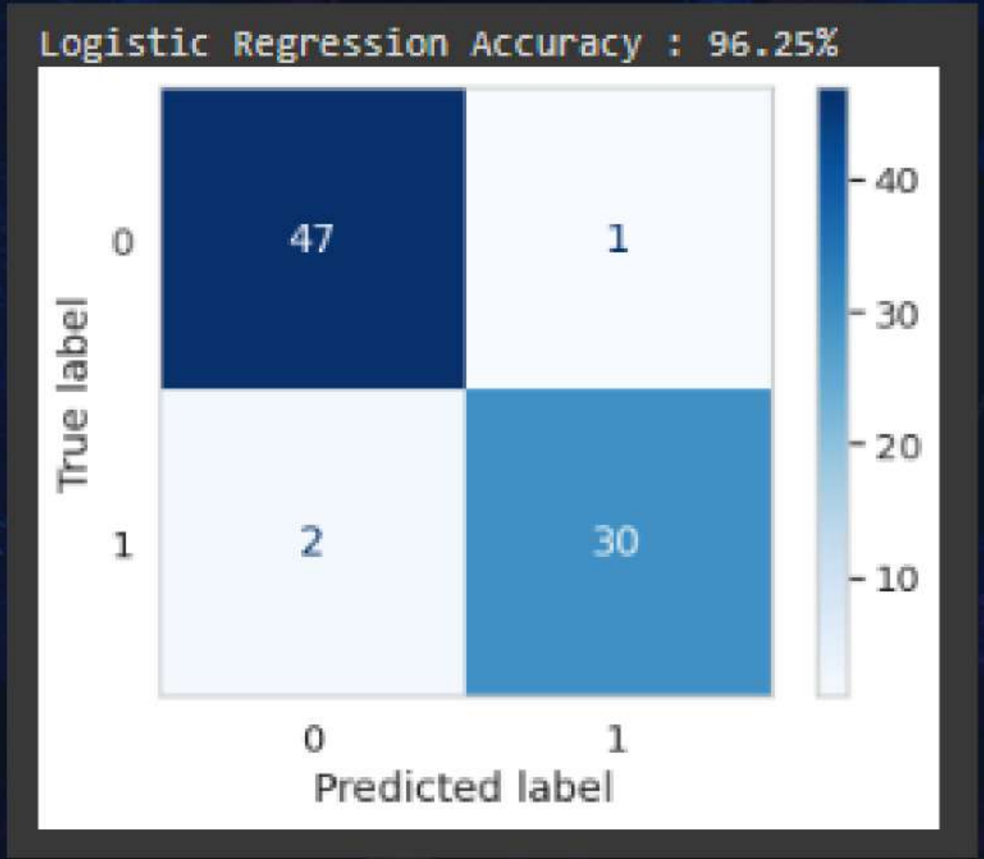
```
#Build model BERT dengan transformer
def build_model(transformer, loss = 'categorical_crossentropy', max_len = 512):
    input_word_ids = tf.keras.layers.Input(shape = (max_len,), dtype = tf.int32, name = "input_word_ids")
    sequence_output = transformer(input_word_ids)[0]
    cls_token = sequence_output[:, 0, :]

    #adding dropout layer
    x = tf.keras.layers.Dropout(0.40)(cls_token)

    #using a dense layer of 2 neurons as the number of unique categories is 2.
    out = tf.keras.layers.Dense(2, activation = 'softmax')(x)

    bert_model = tf.keras.Model(inputs = input_word_ids, outputs = out)
    bert_model.compile(tf.keras.optimizers.Adam(lr = 3e-5), loss = loss, metrics = ['accuracy'])
    return bert_model
```



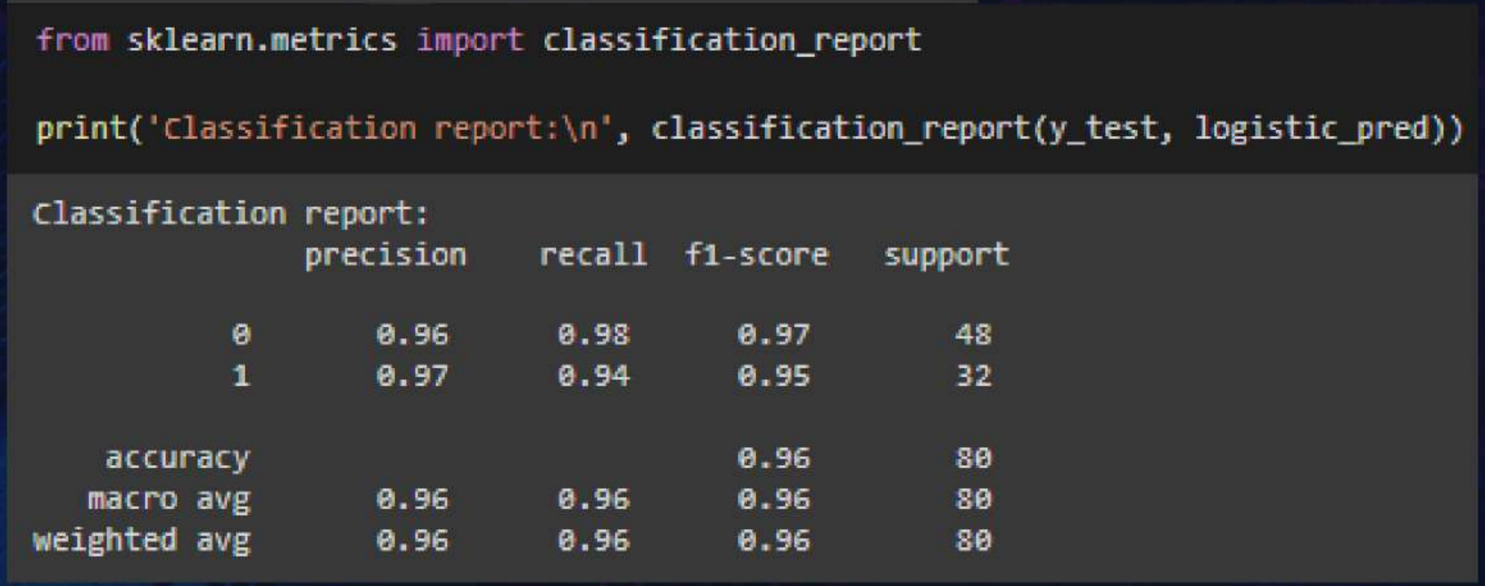
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_word_ids (InputLayer)  [(None, 100)]            0

 tf_distil_bert_model (TFDis  TFBaseModelOutput(last_h  66362880
 tilBertModel)               idden_state=(None, 100,
                             768),
                              hidden_states=None, att
                             entions=None)

 tf.__operators__.getitem (S  (None, 768)              0
 licingOpLambda)

 dropout_20 (Dropout)        (None, 768)               0

 dense_1 (Dense)             (None, 2)                 1538

=================================================================
Total params: 66,364,418
Trainable params: 66,364,418
Non-trainable params: 0
_________________________________________________________________
```

PERFORMANCE MODEL

LOGISTIC REGRESSION



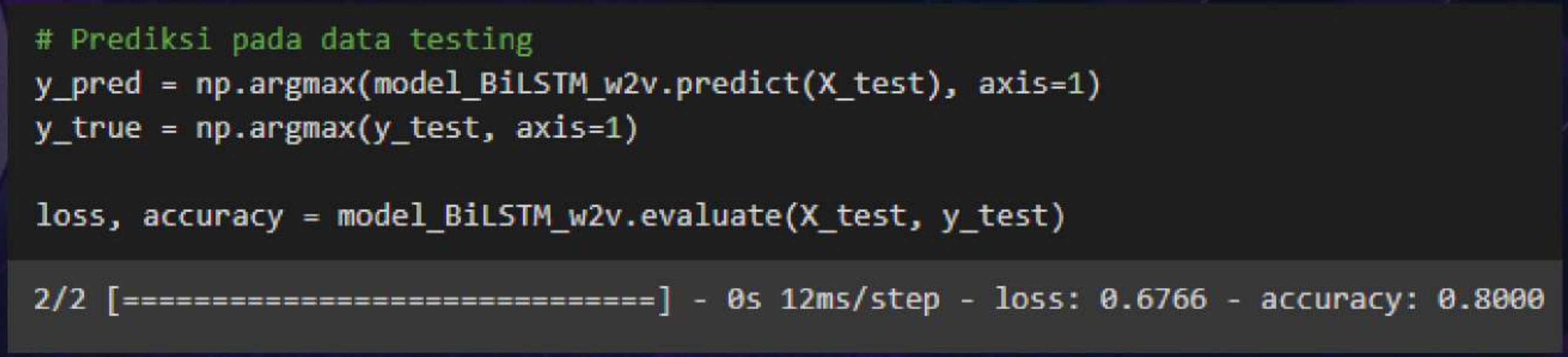
```
Logistic Regression Accuracy : 96.25%
```



```
from sklearn.metrics import classification_report

print('Classification report:\n', classification_report(y_test, logistic_pred))
```

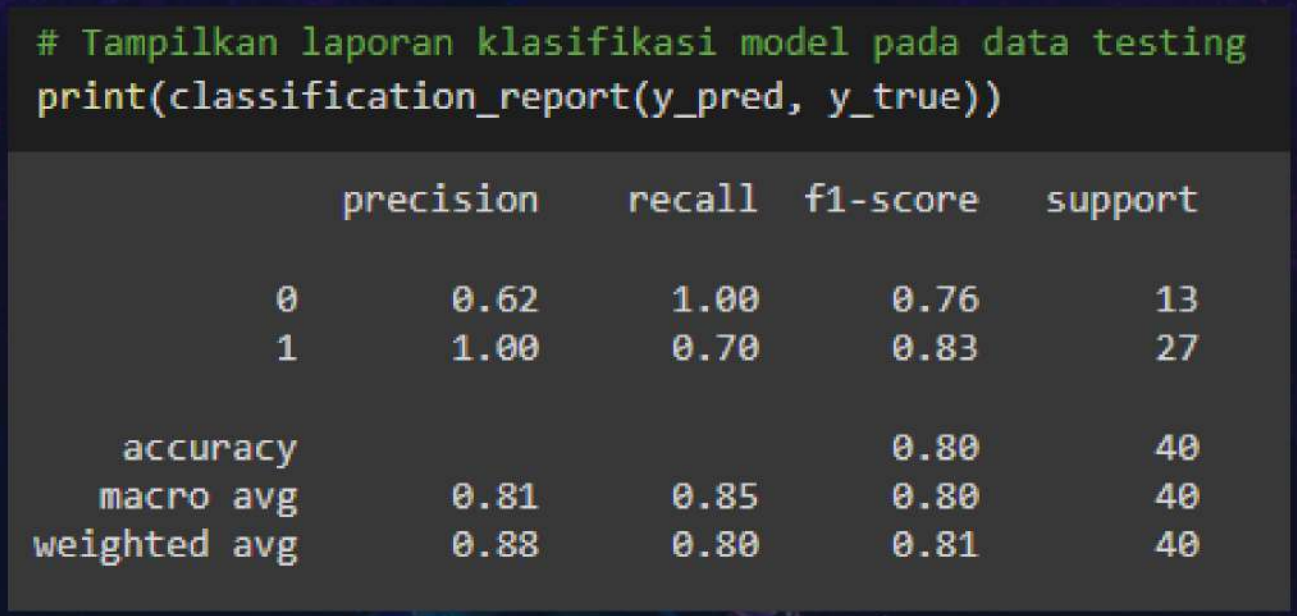
```
Classification report:
               precision    recall  f1-score   support

           0       0.96      0.98      0.97        48
           1       0.97      0.94      0.95        32

    accuracy                           0.96        80
   macro avg       0.96      0.96      0.96        80
weighted avg       0.96      0.96      0.96        80
```

BI-LSTM + WORD2VECT



```
# Prediksi pada data testing
y_pred = np.argmax(model_BiLSTM_w2v.predict(X_test), axis=1)
y_true = np.argmax(y_test, axis=1)

loss, accuracy = model_BiLSTM_w2v.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 12ms/step - loss: 0.6766 - accuracy: 0.8000
```

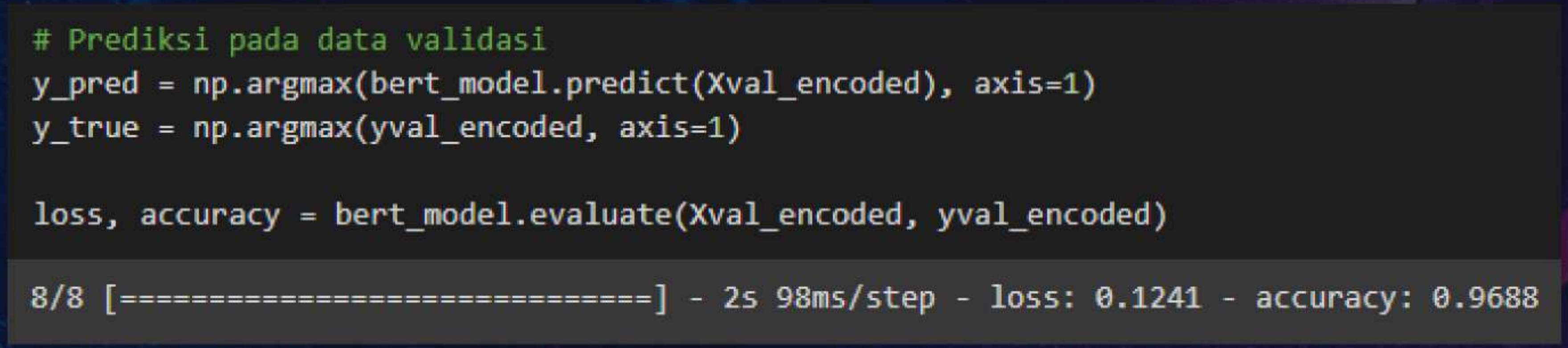


```
# Tampilkan laporan klasifikasi model pada data testing
print(classification_report(y_pred, y_true))
```

```
              precision    recall  f1-score   support

           0       0.62      1.00      0.76        13
           1       1.00      0.70      0.83        27

    accuracy                           0.80        40
   macro avg       0.81      0.85      0.80        40
weighted avg       0.88      0.80      0.81        40
```

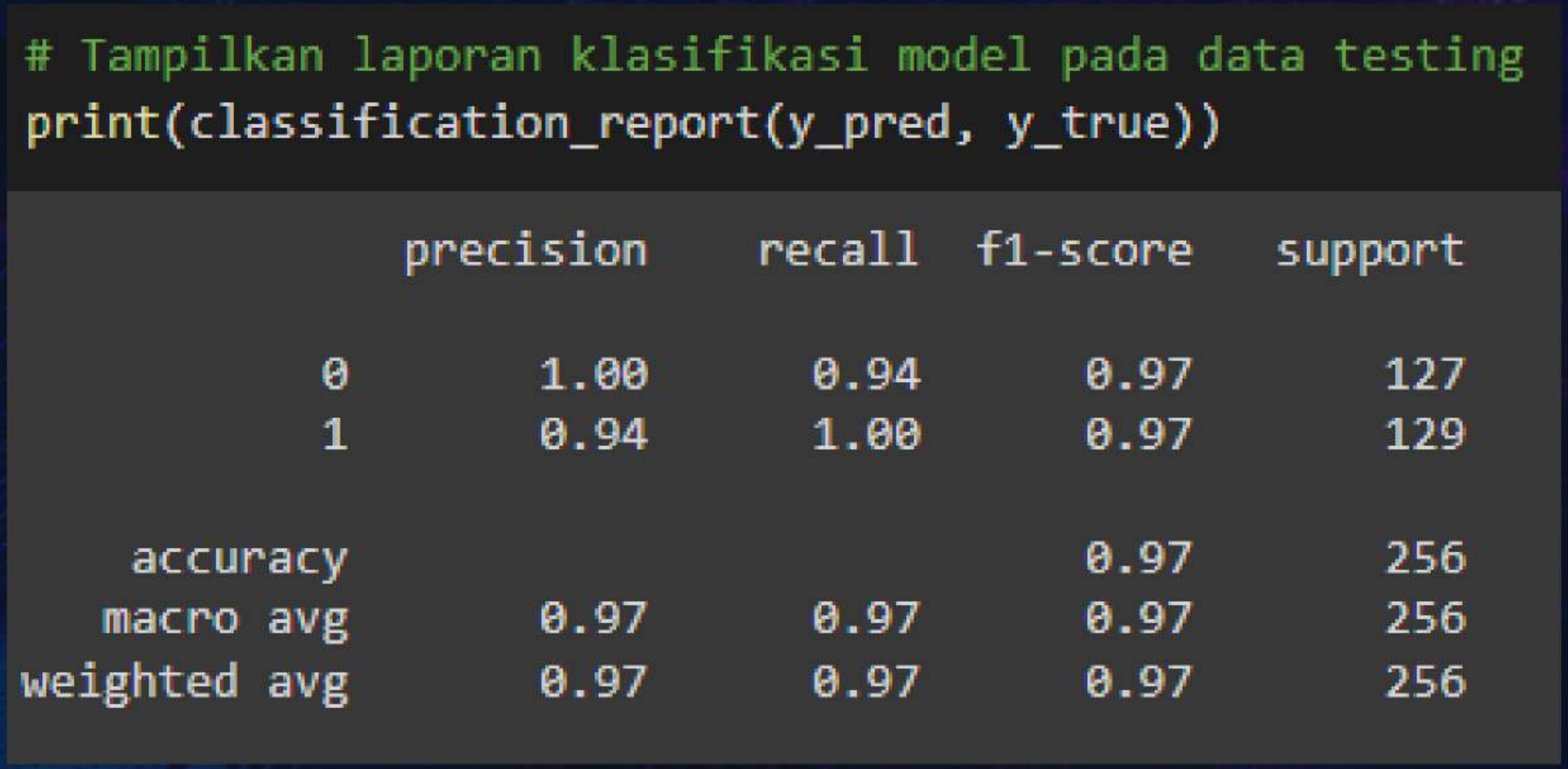
PERFORMANCE MODEL

BERT FINE TUNING



```
# Prediksi pada data validasi
y_pred = np.argmax(bert_model.predict(Xval_encoded), axis=1)
y_true = np.argmax(yval_encoded, axis=1)

loss, accuracy = bert_model.evaluate(Xval_encoded, yval_encoded)
```

```
8/8 [==============================] - 2s 98ms/step - loss: 0.1241 - accuracy: 0.9688
```

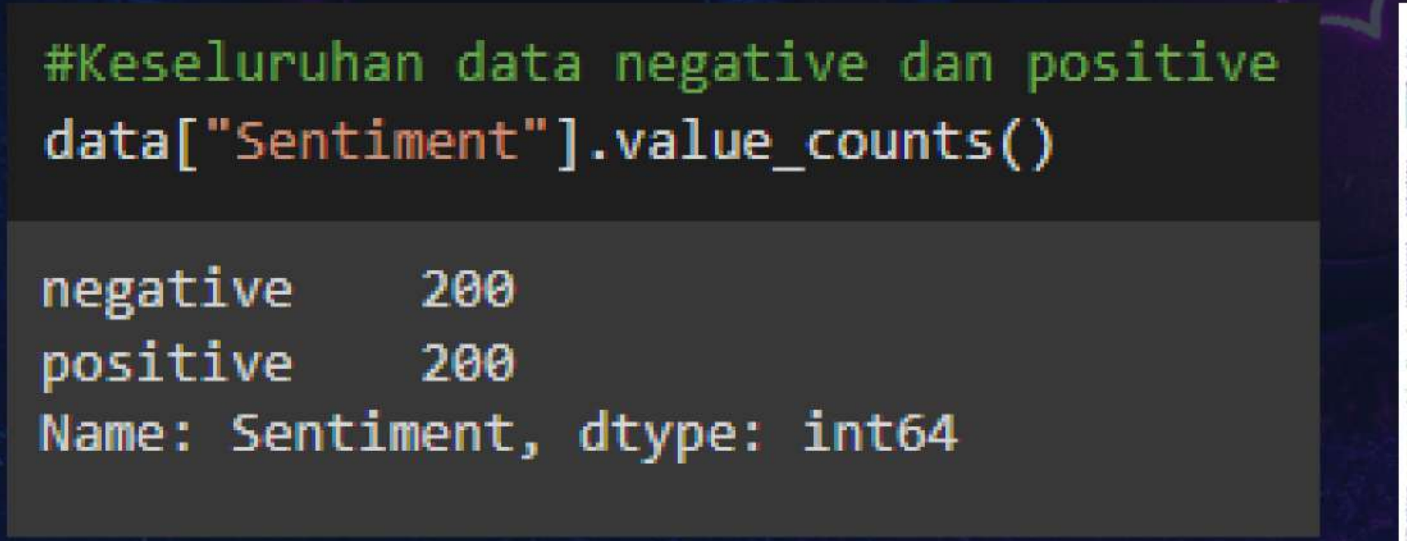


```
# Tampilkan laporan klasifikasi model pada data testing
print(classification_report(y_pred, y_true))
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       127
           1       0.94      1.00      0.97       129

    accuracy                           0.97       256
   macro avg       0.97      0.97      0.97       256
weighted avg       0.97      0.97      0.97       256
```

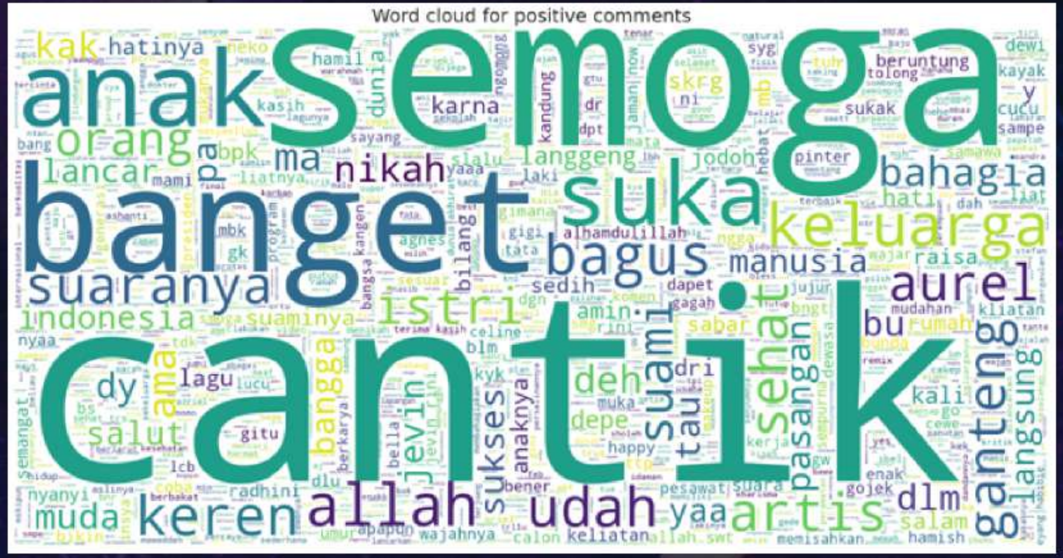
KESIMPULAN

Jumlah total komentar berdasarkan sentimen positive dan negative: 400 data.
Menghasilkan sentimen yang memiliki kecenderungan relatif sama dikarenakan jumlah positive dan negative yang sering muncul pada wordcloud.

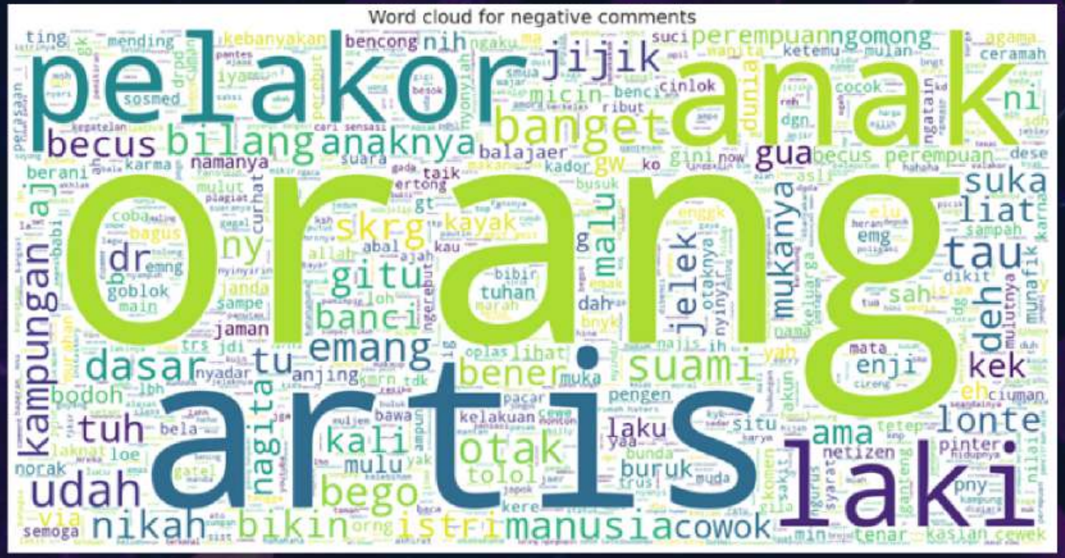


```
#Keseluruhan data negative dan positive
data["Sentiment"].value_counts()
```

```
negative    200
positive    200
Name: Sentiment, dtype: int64
```

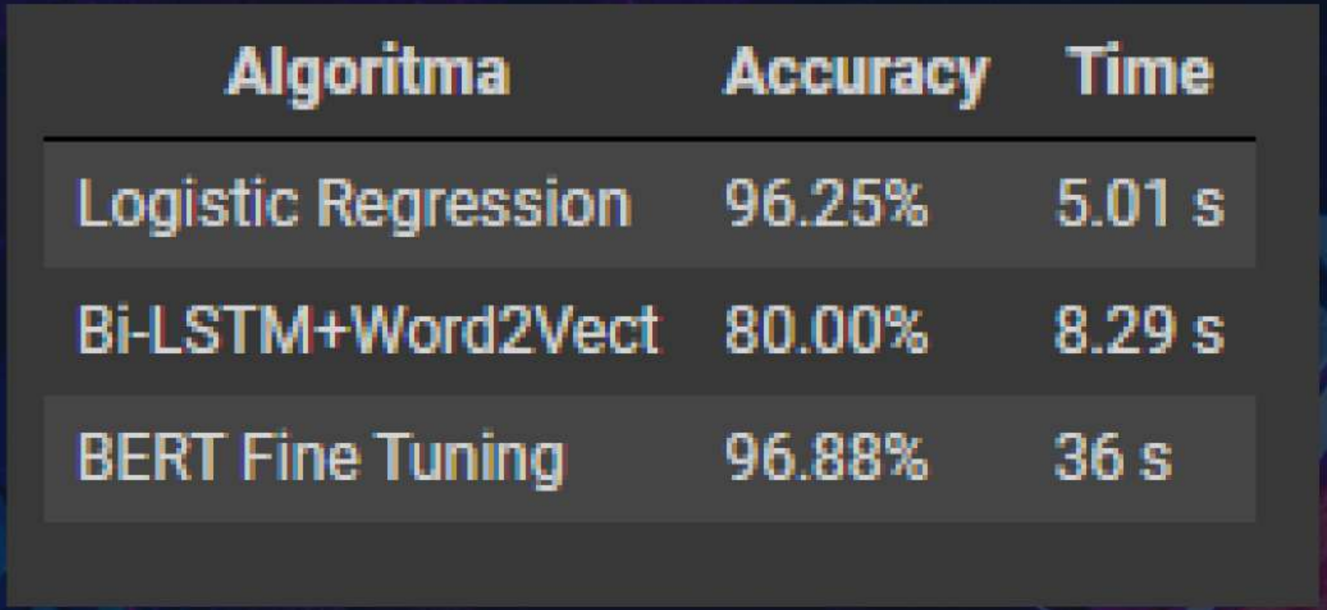


Word cloud for positive comments



Word cloud for negative comments

Berdasarkan hasil performance model algoritma BERT dengan Fine Tuning memberikan insight terbaik dibandingkan algoritma seperti Logistic Regression maupun Bi-LSTM dengan Word2Vect dari segi penilaian akurasi dan waktu yang dihasilkan pada Sentiment Analysis Cyberbullying pada Instagram Comment.



Algoritma	Accuracy	Time
Logistic Regression	96.25%	5.01 s
Bi-LSTM+Word2Vect	80.00%	8.29 s
BERT Fine Tuning	96.88%	36 s