LAMPIRAN

Lampiran 1. *Source Code Program*

```
!pip3 install
git+https://github.com/JustAnotherArchivist/snscrape.git

import snscrape.modules.twitter as sntwitter

import pandas as pd
```

L1. Kode untuk melakukan import library snscrapper

```
keyword = "stress mahasiswa"
data = []
limits = 2000
# scrape twitter data
for i, tweet in enumerate(sntwitter.TwitterSearchScraper(keyword
+ ' lang:id').get_items()):
 if i > limits:
 break
 data.append([tweet.date,tweet.user.username, tweet.content])
# create dataframe
df = pd.DataFrame(data, columns=['Date', 'User', 'Tweet'])
print(df)
# make csv
df.to_csv('tweet_{}.csv'.format(keyword),index=False,
encoding='utf-8')
```

L2. Kode Untuk Melakukan Crawling

```
import string
import re
#seluruh fungsi untuk cleaning text
#remove mention
def remove_mention(text):
 text = re.sub(r'@\w+', '', text)
 return text
def cleaningText(text):
 text = re.sub(r'#[A-Za-z0-9]+', '', text) # hapus hashtag
 text = re.sub(r'RT[\s]', '', text) # hapus RT
 text = re.sub(r"http\S+", '', text) # hapus link
 text = re.sub(r'[0-9]+', '', text) # hapus angka
 text = re.sub(r'(.)\1+', r'\1', text) #hapus huruf berlebih
di akhir kata
 text = text.replace('\n', ' ') # mengganti baris baru ke dalam
space
 text = text.translate(str.maketrans('', '',
string.punctuation)) # hapus semua tanda baca
 text = text.strip(' ') # hapus spasi karakter dari teks kiri
dan kanan
 emoj = re.compile("["
```

```
 u"\U0001F600-\U0001F64F" # emoticon
 u"\U0001F300-\U0001F5FF" # simbol & piktograf
 u"\U0001F680-\U0001F6FF" # transport & simbol peta
 u"\U0001F1E0-\U0001F1FF" # flags (iOS)
 u"\U00002500-\U00002BEF" # Karakter china
 u"\U00002702-\U000027B0"
 u"\U00002702-\U000027B0"
 u"\U000024C2-\U0001F251"
 u"\U0001f926-\U0001f937"
 u"\U00010000-\U0010ffff"
 u"\u2640-\u2642"
 u"\u2600-\u2B55"
 u"\u200d"
 u"\u23cf"
 u"\u23e9"
 u"\u231a"
 u"\ufe0f" # dingbats
 u"\u3030"
 "]+", re.UNICODE)
 text = re.sub(emoj, '', text)
 return text
#remove ekspresi
def remove_expresion(text):
text = re.sub(r'\b(wk)+\b', '', text)
 # haha
 text = re.sub(r'\b(ha)+\b', '', text)
 # hehe
 text = re.sub(r'\b(he)+\b', '', text)
 # hihi
 text = re.sub(r'\b(hi)+\b', '', text)
 # hoho
 text = re.sub(r'\b(ho)+\b', '', text)
 # kwkw
 text = re.sub(r'\b(kw)+\b', '', text)
 # xixi
 text = re.sub(r'\b(xi)+\b', '', text)
 # xd
 text = re.sub(r'\b(xd)+\b', '', text)
 return text
```

L3. Kode Untuk Melakukan *Cleaning*

```
def casefoldingText(text):
 # Mengubah semua karakter dalam teks menjadi huruf kecil
 text = text.lower()
 return text
```

L4. Kode Untuk Melakukan *Case Folding*

```
def stemmingText(text): # digunakan untuk menghilangkan
imbuhan dari sebuah kata
 factory = StemmerFactory()
 stemmer = factory.create_stemmer()
 text = [stemmer.stem(word) for word in text]
 return text
```

L5. Kode Untuk Melakukan Stemming

```
# Mengubah label menjadi angka
# labeling data: 0 = Ringan 1 = Sedang 2 = Berat
dataset['Label'] = dataset['Label'].replace(['Ringan'], 0)
dataset['Label'] = dataset['Label'].replace(['Sedang'], 1)
dataset['Label'] = dataset['Label'].replace(['Berat'], 2)
dataset
```

L6. Kode Untuk Mengubah Label Menjadi Angka

```
# Pisahkan data (dengan data train 80%, data test 20%)
from sklearn.model_selection import train_test_split
X = dataset['Tweet']
y = dataset['Label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

L7. Kode Untuk Melakukan Pembagian Data

```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
```

L8. Kode Untuk Import *Tokenizer*

```
#model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense,
Dropout
embed_dim = 32
model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length =
maxlen_fix))
model.add(LSTM(256))
model.add(Dropout(0.2))
model.add(Dense(3, activation = 'softmax'))
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
```

```
model.compile(loss = 'sparse_categorical_crossentropy',
optimizer = 'adam', metrics = ['accuracy'])
print(model.summary())
```

L9. Kode Untuk Membuat Model

```
model_prediction = model.fit(X_train, y_train, validation_split
= 0.1, epochs = 10, batch_size=32, callbacks=[cb])
print(cb.logs)
print(sum(cb.logs))
```

L10. Kode Untuk Melakukan Pelatihan

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
accuracy = accuracy_score(y_test, y_pred)
#jika ingin mengetes menggunakan y_train
# accuracy = accuracy_score(y_train, y_pred)
print('Model Accuracy on Test Data:', accuracy)
confusion_matrix(y_test, y_pred)
#jika ingin mengetes menggunakan y_train
# confusion_matrix(y_train, y_pred)
fig, ax = plt.subplots(figsize = (8,6))
sns.heatmap(confusion_matrix(y_true = y_test, y_pred = y_pred),
fmt = 'g', annot = True)
# confusion_matrix(y_test, y_pred)
# sns.heatmap(confusion_matrix(y_true = y_train, y_pred =
y_pred), fmt = 'g', annot = True)
ax.xaxis.set_label_position('top')
ax.xaxis.set_ticks_position('top')
ax.set_xlabel('Prediction', fontsize = 14)
ax.set_xticklabels(['Ringan (0)', 'Sedang (1)', 'Berat (2)'])
ax.set_ylabel('Actual', fontsize = 14)
ax.set_yticklabels(['Ringan (0)', 'Sedang (1)', 'Berat (2)'])
plt.show()
```

L11. Kode Untuk Menampilkan *Confusion Matrix*

Lampiran 2. Identitas Pakar

A. Pakar Psikologi

Nama	Prof. Dr. Kadek Suranata, S.Pd., M.Pd., Kons.
Profesi	Guru Besar dalam bidang Bimbingan dan Konseling di Fakultas Ilmu Pendidikan, Universitas Pendidikan Ganesha
Email	kadek.suranata@undiksha.ac.id

B. Pakar Bahasa Indonesia

Nama	Dr. Kadek Wirahyuni, S.Pd., M.Pd.
Profesi	Dosen di Program Studi Pendidikan Bahasa dan Sastra Indonesia, Universitas Pendidikan Ganesha
Email	kadek.wirahyuni@undiksha.ac.id