LAMPIRAN

Lampiran 1. Proses Mempersiapkan Dataset

```
#import dan menampilkan dataset
import string
from sklearn.pipeline import Pipeline
import pandas as pd
import numpy as np
data = pd.read_csv('/content/drive/MyDrive/DATA/dataset_tweet_vaksin_covid19.csv')
data

#delete kolom yang tidak
data.drop(data.columns[[0,1]], axis = 1, inplace = True)
data[:10]

#Visualisasi dataset sebelum proses text preprocessing
from wordcloud import WordCloud
import matplotlib.pyplot as plt

#Polarity == 0 negative
train_s0 = data[data["Label"] == 0]
all_text_s0 = ' '.join(word for word in train_s0["Text"])
wordcloud = WordCloud(colormap='Reds', width=1000, height=1000, mode='RGBA', bac
kground_color='white').generate(all_text_s0)
plt.figure(figsize=(20,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
```

```
plt.show()


#Polarity == 1 positive

train_s1 = data[data["Label"] == 1]

all_text_s1 = ' '.join(word for word in train_s1["Text"])

wordcloud = WordCloud(width=1000, height=1000, colormap='Blues', background_color
='white', mode='RGBA').generate(all_text_s1)

plt.figure( figsize=(20,10))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.margins(x=0, y=0)
```

```
plt.show()

#Menampilkan jumlah data pada kelas positif dan negatif

print("Hasil Sentimen :")

print("Tweet Negatif :", len(train_s0), "({}%)".format(100*len(train_s0)/len(data_clean)))

print("Tweet Positif :", len(train_s1), "({}%)".format(100*len(train_s1)/len(data_clean)))
```

Lampiran 2. Proses *Text Preprocessing*

```
#proses case folding

import re

def casefolding(Text):

    Text = Text.lower()

    Text = Text.strip(" ")

    Text = re.sub("https*\S+", " ", Text)

    Text = ' '.join(re.sub("(@\S+[A-Za-z0-9])|([^0-9A-Za-
z \t])|(\w+:\/\/\S+\d)|(eee)"," ",Text).split())

    Text = re.sub(r"\d+", "", Text)

    return Text

data['Text'] = data['Text'].apply(casefolding)

data[:10]


#proses tokenizing

def token(Text):

    str = Text.split(' ')

    data = []

    a = -1

    for u in str:
```

```
        a = a + 1
    if u == '':
        data.append(a)
    p = 0
    b = 0
    for q in data:
        b = q - p
        del str[b]
        p = p + 1
    return str
data['Text'] = data['Text'].apply(token)
data[:10]
```

```
#proses filtering
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

def stopword_removal(Text):
    filtering = stopwords.words('indonesian','english')
    x = []
    data = []
    def myFunc(x):
```

```
        if x in filtering:
            return False
        else:
            return True
    fit = filter(myFunc, Text)
    for x in fit:
        data.append(x)
    return data
data['Text'] = data['Text'].apply(stopword_removal)
data[:10]

#proses stemming
!pip install Sastrawi
from sklearn.pipeline import Pipeline
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def stemming(Text):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    do = []
    for x in Text:
        stm = stemmer.stem(x)
        do.append(st)
    clean = []
    clean = " ".join(do)
```

```
    print(clean)
    return clean
data['Text'] = data['Text'].apply(stemming)

data.to_csv('data_tweet_clean.csv', index=False)
data_clean = pd.read_csv('data_tweet_clean.csv', encoding='latin1')
data_clean.head()
```

Lampiran 3. Proses Pembobotan Kata

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

tf = TfidfVectorizer()
text_tf = tf.fit_transform(data_clean['Text'])
text_tf
```

Lampiran 4. Proses Klasifikasi

```
#Splitting data
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(text_tf, data_clean['Label'], test_size=0.2, r
andom_state=20)

#Performa Algoritma Support Vector Machine
from sklearn import svm
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

clfsvm = svm.SVC(kernel="linear")
clfsvm.fit(X_train,y_train)
predicted = clfsvm.predict(X_test)
print("Accuracy :" , accuracy_score(y_test, predicted))
print("Precision :" , precision_score(y_test, predicted))
print("Recall :" , recall_score(y_test, predicted))
print("f1_score :" , f1_score(y_test, predicted))

print(f'confusion matrix: \n {confusion_matrix(y_test, predicted)}')
print('-----------------------------------------------\n')
print(classification_report(y_test, predicted, zero_division=0))

#Performa Algoritma Naive Bayes Clasifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

clf = MultinomialNB().fit(X_train, y_train)
predicted = clf.predict(X_test)
print("Accuracy :" , accuracy_score(y_test, predicted))
```

```
print("Precision :" , precision_score(y_test, predicted))
print("Recall :" , recall_score(y_test, predicted))
print("f1_score :" , f1_score(y_test, predicted))

print(f'confusion matrix: \n {confusion_matrix(y_test, predicted)}')
print('-----------------------------------------------\n')
print(classification_report(y_test, predicted, zero_division=0))
```