APPENDIX

IMPORT MODULE USED FOR LSTM

```
1. import pandas as pd
2. import numpy as np
3. import re
4. import string
5. import nltk
6. import matplotlib.pyplot as plt
7. from nltk.tokenize import sent_tokenize, word_tokenize
8. from nltk.corpus import stopwords
9. from keras.preprocessing.text import Tokenizer
10. from keras.preprocessing.sequence import pad_sequences
11. from keras.layers import Dense , Input , LSTM , Embedding, Dropout ,
    Activation, GRU, Flatten
12. from keras.layers import Bidirectional, GlobalMaxPool1D
13. from keras.models import Model, Sequential
14. from keras.layers import Convolution1D
15. from keras import initializers, regularizers, constraints, optimizers,
    layers
16. from sklearn.model_selection import train_test_split
17. from sklearn.metrics import accuracy_score, f1_score,
    confusion_matrix, recall_score, precision_score, confusion_matrix
18. nltk.download('stopwords')
19. nltk.download('punkt')
```

IMPORT MODULE USED FOR SVM

```
1. import pandas as pd
2. import re
3. import nltk
4. import string
5. import numpy as np
6. import matplotlib.pyplot as plt
7. from nltk.tokenize import word_tokenize
8. from nltk.corpus import stopwords
9. from keras.preprocessing.text import Tokenizer
10. from sklearn.feature_extraction.text import CountVectorizer
11. from sklearn.model_selection import train_test_split
12. from sklearn.metrics import accuracy_score, f1_score, recall_score,
    precision_score, confusion_matrix
```

IMPORT DATASET

```
1. from google.colab import files
2. uploaded = files.upload()
3. import io
4. df = pd.read_csv(io.BytesIO(uploaded['dataset4.csv']))
5. df.tweet = df.tweet.astype(str)
6. df.head(10)
```

GIVING STATUS TO DATA (POSITIVE/NEGATIVE) BASED ON THE EXISTING POLARITY

```
def buatstatus(polarity):
  if polarity < 0:
    return 'negatif'
  else:
    return 'positif'
df['status'] = df['polarity'].apply(buatstatus)
df.head(10)
```

GIVE LABEL (0/1) ON DATA BASED ON STATUS

```
def buatlabel(status):
  if status == 'negatif':
    return 0
  else:
    return 1
df['label'] = df['status'].apply(buatlabel)
df.head(10)
```

CLEANSING PROCESS ON DATASET

```
stop_words = set(stopwords.words('english'))
def cleantext(text):
text = text.lower()
text = word_tokenize(text)
text = [item for item in text if item not in stop_words]
text = ' '.join(text) #menggabungkan kata
text = re.sub(r"\d+", "", text)
return text
df['tweet'] = df['tweet'].apply(cleantext)
df.head(10)
```

SPLIT TRAINING DATA AND TESTING DATA ON LSTM

```
x = df['tweet']
y = df['label']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

DOING TOKENIZATION PROCESS ON DATASET FOR LSTM

```
num_words = 2500
tokenizer = Tokenizer(num_words)
tokenizer.fit_on_texts(x_train)
list_tokenized_train = tokenizer.texts_to_sequences(x_train)
```

```
5. max_review_length = 100
6. x_train = pad_sequences(list_tokenized_train,Maxlen=max_review_length)
```

BUILD LSTM MODEL

```
1. model = Sequential()
2. model.add(Embedding(num_words+1, 100, input_length=max_review_length))
3. model.add(LSTM(600))
4. model.add(Dense(1, activation='sigmoid'))
5. model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['ac
   curacy'])
6. model.summary()
```

CREATING AND DISPLAYING ACCURACY AND VALIDATION RESULTS OF LSTM TRAINING DATA

```
1. model.fit(x_train,y_train,epochs=10,batch_size=32,validation_split=0.2
```

CREATE AND DISPLAY ACCURACY RESULTS OF LSTM TESTING DATA

```
1. list_tokenized_test = tokenizer.texts_to_sequences(x_test)
2. x_test = pad_sequences(list_tokenized_test, maxlen=max_review_length)
3. prediction = model.predict(x_test)
4. y_pred = (prediction > 0.5)
```

CHANGING DATA FROM DATAFRAME FORM TO NUMPY ON SVM

```
1. x = df['tweet'].to_numpy()
2. y = df['label'].to_numpy()
```

DOING VECTORIZER PROCESS ON SVM

```
1. vectorizer = CountVectorizer()
2. x = vectorizer.fit_transform(x)
```

BUILD SVM MODEL

```
1. class SVM:
2. def __init__(self, n_iters= 10, learning_rate= 1, lambda_param= 1):
3. self.iterations = n_iters
4. self.lr = learning_rate
5. self.lambda_param = lambda_param
```

```
self.weight = None
self.bias = None
def fit(self, x, y):
y_ = np.where(y<=0, -1, 1)
n_samples, n_features = x.shape
self.weight = np.random.random_sample(n_features)
self.bias = np.random.random_sample()
for _ in range(self.iterations):
for index, x_value in enumerate(x):
condition = y_[index] * ((x_value @ self.weight) - self.bias) >= 1
if condition:
d_weight = 2 * self.lambda_param * self.weight
d_bias = 0
self.weight -= self.lr * d_weight
self.bias -= self.lr * d_bias
else:
y_[index] = y_[index].reshape((1, 1))
d_weight = (2 * self.lambda_param * self.weight) - (x_value.T *
y_[index])
d_bias = y_[index]
self.weight -= self.lr * d_weight
self.bias -= self.lr * d_bias
```

BUILD SVM MODEL TO KNOW SVM MODEL PREDICTION

```
def predict(self, x):
linear_output = x @ self.weight + self.bias
sign_matrix = np.sign(linear_output)
result = np.where(sign_matrix<=0, 0, 1).astype(np.float32)
return result
```

BUILDING SVM MODEL TO KNOW SVM ACCURACY RESULTS

```
def accuracy(self, y_pred, y_test):
return np.sum(y_pred == y_test)/ len(y_test)
```

SPLIT DATATRAINING AND DATA TESTING ON SVM

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

ADDING 1 DIMENSION TO NUMPY FORMS ON SVM

```
y_train = np.array(y_train)
y_train = np.expand_dims(y_train, axis=-1)
```

MAKING PREDICTIONS FOR DATA TRAINING ON SVM MODEL

```
1. svm = SVM()
2. svm.fit(x_train, y_train)
```

MAKING PREDICTIONS FOR TESTING DATA ON SVM MODEL

```
1. predictions = svm.predict(x_test)
```

SHOWING THE ACCURACY OF TESTING DATA THAT HAVE BEEN PREDICTED ON THE SVM MODEL

```
1. score = svm.accuracy(predictions, y_test)
2. score
```

SHOWING CONFUSSION MATRIX

```
1. from sklearn.metrics import confusion_matrix
2. print(confusion_matrix(y_test , predictions))
```

SHOWING CLASSIFICATION REPORT

```
1. from sklearn.metrics import classification_report
2. print(classification_report(y_test, predictions))
```

Similarity Report

PAPER NAME

18.K1.0020.docx

WORD COUNT	CHARACTER COUNT
8971 Words	**44650 Characters**
PAGE COUNT	FILE SIZE
43 Pages	**814.6KB**
SUBMISSION DATE	REPORT DATE
Jul 4, 2022 9:41 AM GMT+7	**Jul 4, 2022 9:42 AM GMT+7**

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 6% Publications database
- Crossref database
- Crossref Posted Content database
- 6% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 10 words)