DAFTAR PUSTAKA

Aisyah, S. (2020). *Klasifikasi Spam Pada Email Menggunakan Metode Support Vector Machine Dan Deteksi Anomaly*. Universitas Sriwijaya, Sistem Komputer, Malang, Jawa Timur.

Anugroho, P., Winarno, I., & M, N. R. (2011). *Klasifikasi Spam Dengan Metode Naive Bayes Classifier Menggunakan Java Programming.* Institut Teknologi Sepuluh November ITS Keputih Sukolilo Surabaya, 1-11.

Basyar, I. (2019). *Klasifikasi Email Spam Menggunakan Jaringan Gated Recurrent Unit Dan Long Short-Term Memory*. Universitas Telkom, Informatika, Bandung, Jawa Barat.

Brownlee, J. (2017, April 28). *Dropout with LSTM Networks for Time Series Forecasting.* Retrieved March 27, 2021, from Machine Learning Mastery: https://machinelearningmastery.com/use-dropout-lstm-networks-time-series-forecasting/

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). *Convolutional Reccurent Neural Networks For Music Classification.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1-5.

Dongre, S., & Patidar, K. (2019). *E-Mail Spam Classification Using Long Short-Term Memory Method.* International Journal of Scientific Research & ENgineering Trends, 1659-1665.

Fitriyanto, A. (2019). Klasifikasi E-Mail Spam Menggunakan Metode K-Nearest Neighbor. Universitas Muhammadiyah Jember, Teknik Informatika, Jember, Jawa Timur.

Fitriyanto, A., & Saifudin, I. (2019). *Klasifikasi E-Mail Spam Menggunakan Metode K-Nearest Neighbor.* Jurusan Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Jember.

Gomez, R. (2018, May 23). *Blog Raul Gomez*. Retrieved from Blog Raul Gomez: https://gombru.github.io/2018/05/23/cross_entropy_loss/

Hayuningtyas, R. Y. (2017). *Aplikasi Filtering Of Spam Email Menggunakan Naive Bayes.* IJCIT (Indonesia Journal on Computer and Information Technology), 53-60.

Imam. (2018, January 25). *Memahami Epoch Batch Size Dan Iteration*. Retrieved March 25, 2021, from Imam Digmi: https://imam.digmi.id/post/memahami-epoch-batch-size-dan-iteration/

Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., & Aurelio, M. (2015). *Learning Longer Memory In Reccurent Neural Networks*. Workshop Contribution at ICLR.

Poomka, P., Pongsena, W., Kerdprasop, N., & Kerdprasop, K. (2019). *Sms Spam Detection Based On Long Short-Term Memory And Gated Recurrent Unit*. International Journal of Future Computer and Communication, 8, 11-15.

Powers, D. (2020). *Evaluation : From Precision, Recall And F-Measure To ROC, Informedness, Markedness & Correlation*. Arxiv, 1 - 27.

Pratiwi, S. N., & Ulama, B. S. (2016). *Klasifikasi Email Spam Dengan Menggunakan Metode Support Vector Machine Dan K-Nearest Neighbor.* Jurnal Sains dan Seni ITS, *5*, 344-349.

Pudjiarti, E. (2016, September). *Prediksi Spam Email Menggunakan Metode Support Vector Machine dan Particle Swarm Optimization.* Jurnal Pilar Nusa Mandiri, XII, 171 - 181.

Raj, H., Weihong, Y., Banbhrani, S. K., & Dino, S. P. (2018). *LSTM Based Short Massage Service (SMS) Modeling For Spam Classification.* School of Computer Science and Technology Dalian University of Technology, 76-80.

Sandag, G. A., Sambur, R. J., & Bororing, J. (2018). *Klasifikasi Sms Spam Menggunakan Algoritma Support Vector Machine (SVM)*. Seminar nasional Sistem Informasi dan Teknologi Informasi 2018, 291-295.

Setiawan, M. (2018). *Klasifikasi Penyakit pada citra daun menggunakan convolutional neural network.* Retrieved from IPB University Bogor Indonesia: https://repository.ipb.ac.id/handle/123456789/92140

Setiono, A., & Pardede, H. F. (2019). *Klasifikasi SMS Spam Menggunakan Support Vector Machine.* Jurnal Pilar Nusa Mandiri, 15, 275-280.

Sharma, S. (2017, September 6). *Towards Data Science*. Retrieved from Towards Data Science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Sokolova, M., & Lapalme, G. (2009). *A Systematic Analysis Of Performance Measure For Classification Tasks.* Information Processing And Management, 427 - 437.

Vynaima S, R. I. (2018). *Identifikasi SMS Spam Berbahsa Indonesia Menggunakan Algoritma Support Vector Machine.* Universitas Sumatera Utara, Teknologi Informasi, Medan, Sumatera Utara.

Wanto, G.H, R. L., & Susilo, D. D. (2019, September). *Diagnosis Ketidaklurusan (Misalignment) Poros Menggunakan Metode MultiClass Support Vector Machine (SVM)*. Mekanika: Majalah Ilmiah Mekanika, 18, 39-43.

LAMPIRAN

Long Short-Term Memory Training Result

	Accuracy	Ham recall	Spam recall	Ham precision	Spam precision	Ham f1	Spam f1
0	0.854064	1	0.052288	0.868838	1	0.929816	0.099379
1	0.949035	0.999219	0.856209	0.977591	0.994307	0.988287	0.920105
2	0.984059	0.997657	0.94281	0.990951	0.984642	0.994293	0.963272
3	0.989672	0.998959	0.913399	0.986379	0.992895	0.992629	0.951489
4	0.993489	0.998699	0.980392	0.996882	0.991736	0.99779	0.986031
5	0.994836	0.999479	0.98366	0.997403	0.996689	0.99844	0.990132
6	0.993938	0.99974	0.977124	0.996368	0.998331	0.998051	0.987614
7	0.997081	0.99974	0.980392	0.996886	0.998336	0.998311	0.989283
8	0.99753	0.99974	0.982026	0.997144	0.998339	0.99844	0.990115
9	0.997306	0.99974	0.985294	0.997662	0.998344	0.9987	0.991776
10	0.99753	0.99974	0.986928	0.997922	0.998347	0.99883	0.992605
11	0.998653	1	0.988562	0.998181	1	0.99909	0.994248
12	0.998653	1	0.99183	0.9987	1	0.99935	0.995898
13	0.998653	1	0.99183	0.9987	1	0.99935	0.995898
14	0.998428	1	0.993464	0.99896	1	0.99948	0.996721
15	0.999102	1	0.995098	0.99922	1	0.99961	0.997543
16	0.999326	1	0.993464	0.99896	1	0.99948	0.996721
17	0.999102	1	0.995098	0.99922	1	0.99961	0.997543
18	0.999326	1	0.996732	0.99948	1	0.99974	0.998363
19	0.999326	1	0.996732	0.99948	1	0.99974	0.998363
20	0.999326	1	0.995098	0.99922	1	0.99961	0.997543
21	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
22	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
23	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
24	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
25	0.999326	1	0.996732	0.99948	1	0.99974	0.998363
26	0.999326	1	0.996732	0.99948	1	0.99974	0.998363
27	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
28	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
29	0.999551	1	0.998366	0.99974	1	0.99987	0.999182
30	0.999551	1	0.998366	0.99974	1	0.99987	0.999182
31	0.999551	1	0.996732	0.99948	1	0.99974	0.998363
32	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
33	0.999775	1	0.996732	0.99948	1	0.99974	0.998363
34	0.999775	1	0.996732	0.99948	1	0.99974	0.998363
35	0.999551	1	0.998366	0.99974	1	0.99987	0.999182

36	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
37	0.999551	1	0.998366	0.99974	1	0.99987	0.999182
38	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
39	0.999551	0.99974	0.998366	0.99974	0.998366	0.99974	0.998366
40	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
41	0.999775	0.99974	0.998366	0.99974	0.998366	0.99974	0.998366
42	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
43	0.999551	1	0.998366	0.99974	1	0.99987	0.999182
44	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
45	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
46	0.999775	1	0.998366	0.99974	1	0.99987	0.999182
47	0.999551	1	0.998366	0.99974	1	0.99987	0.999182
48	0.999551	0.99974	0.998366	0.99974	0.998366	0.99974	0.998366
49	0.999775	1	0.998366	0.99974	1	0.99987	0.999182

Long Short-Term Memory Validation Result

	Accuracy	**Ham recall**	**Spam recall**	**Ham precision**	**Spam precision**	**Ham f1**	**Spam f1**
0	0.887792	1	0.074074	0.886775	1	0.93999	0.137931
1	0.969479	0.998979	0.755556	0.967359	0.990291	0.982915	0.857143
2	0.982047	0.996936	0.874074	0.98288	0.975207	0.989858	0.921875
3	0.978456	0.998979	0.82963	0.977023	0.99115	0.987879	0.903226
4	0.986535	0.995914	0.918519	0.988844	0.96875	0.992366	0.942966
5	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
6	0.98474	0.997957	0.888889	0.984879	0.983607	0.991375	0.933852
7	0.985637	0.997957	0.896296	0.985873	0.98374	0.991878	0.937984
8	0.985637	0.997957	0.896296	0.985873	0.98374	0.991878	0.937984
9	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
10	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
11	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
12	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
13	0.990126	0.998979	0.925926	0.989879	0.992063	0.994408	0.957854
14	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
15	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
16	0.990126	0.998979	0.925926	0.989879	0.992063	0.994408	0.957854
17	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
18	0.98833	0.995914	0.933333	0.990854	0.969231	0.993377	0.950943
19	0.987433	0.995914	0.925926	0.989848	0.968992	0.992872	0.94697
20	0.990126	0.998979	0.925926	0.989879	0.992063	0.994408	0.957854
21	0.989228	0.996936	0.933333	0.990863	0.976744	0.99389	0.954545
22	0.989228	0.996936	0.933333	0.990863	0.976744	0.99389	0.954545
23	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198

24	0.989228	0.996936	0.933333	0.990863	0.976744	0.99389	0.954545
25	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
26	0.989228	0.996936	0.933333	0.990863	0.976744	0.99389	0.954545
27	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
28	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
29	0.990126	0.997957	0.933333	0.990872	0.984375	0.994402	0.958175
30	0.990126	0.997957	0.933333	0.990872	0.984375	0.994402	0.958175
31	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
32	0.989228	0.997957	0.925926	0.989868	0.984252	0.993896	0.954198
33	0.987433	0.997957	0.911111	0.987867	0.984	0.992886	0.946154
34	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
35	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
36	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
37	0.986535	0.997957	0.903704	0.986869	0.983871	0.992382	0.942085
38	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
39	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
40	0.986535	0.997957	0.903704	0.986869	0.983871	0.992382	0.942085
41	0.991023	0.997957	0.940741	0.991878	0.984496	0.994908	0.962121
42	0.985637	0.997957	0.896296	0.985873	0.98374	0.991878	0.937984
43	0.986535	0.997957	0.903704	0.986869	0.983871	0.992382	0.942085
44	0.987433	0.997957	0.911111	0.987867	0.984	0.992886	0.946154
45	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
46	0.985637	0.997957	0.896296	0.985873	0.98374	0.991878	0.937984
47	0.985637	0.997957	0.896296	0.985873	0.98374	0.991878	0.937984
48	0.98833	0.997957	0.918519	0.988866	0.984127	0.993391	0.950192
49	0.986535	0.997957	0.903704	0.986869	0.983871	0.992382	0.942085

Support Vector Machine Training Result

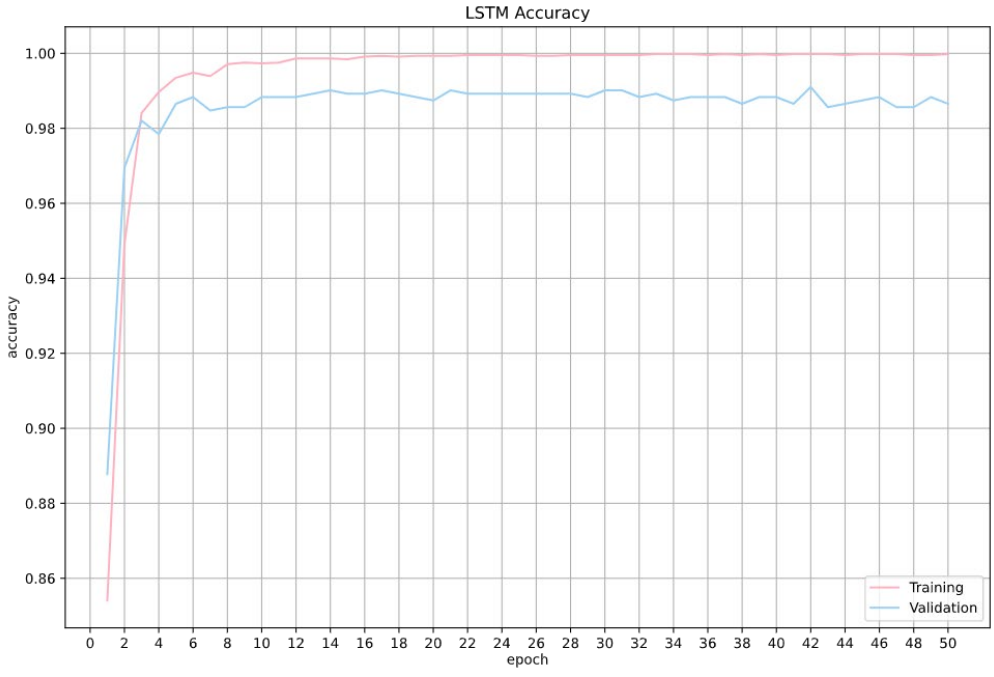
	C	Accuracy	Ham recall	Spam recall	Ham precision	Spam precision	Ham f1	Spam f1
0	0.5	0.988101	1	0.913399	0.986393	1	0.99315	0.95474
1	1	0.995061	1	0.964052	0.994306	1	0.997145	0.981697
2	5	0.999102	1	0.993464	0.99896	1	0.99948	0.996721
3	10	1	1	1	1	1	1	1
4	20	1	1	1	1	1	1	1
5	50	1	1	1	1	1	1	1
6	100	1	1	1	1	1	1	1
7	200	1	1	1	1	1	1	1
8	500	1	1	1	1	1	1	1
9	1000	1	1	1	1	1	1	1

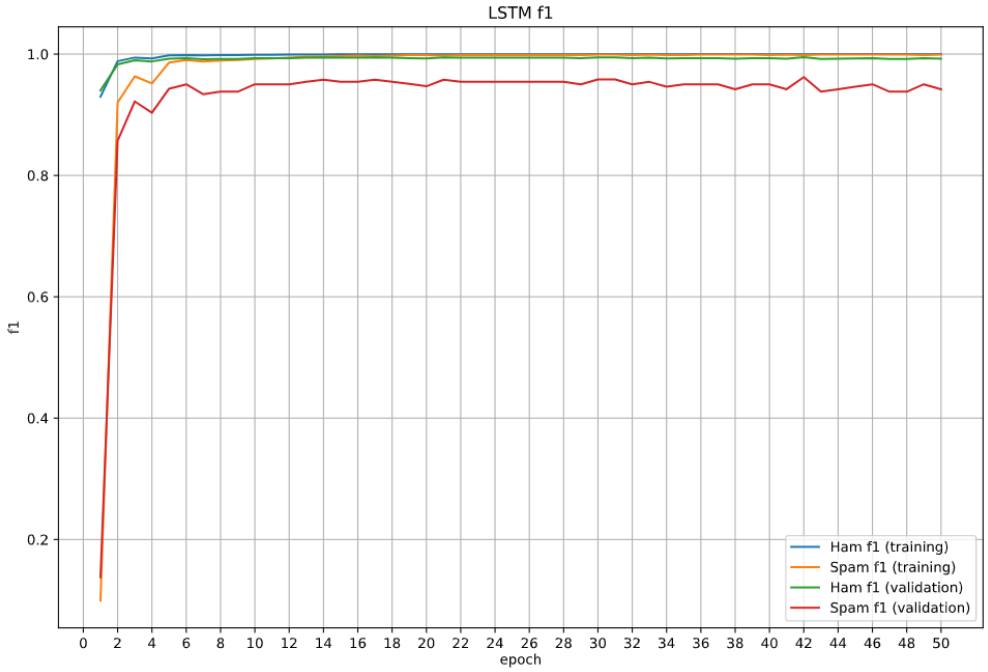
Support Vector Machine Validation Result

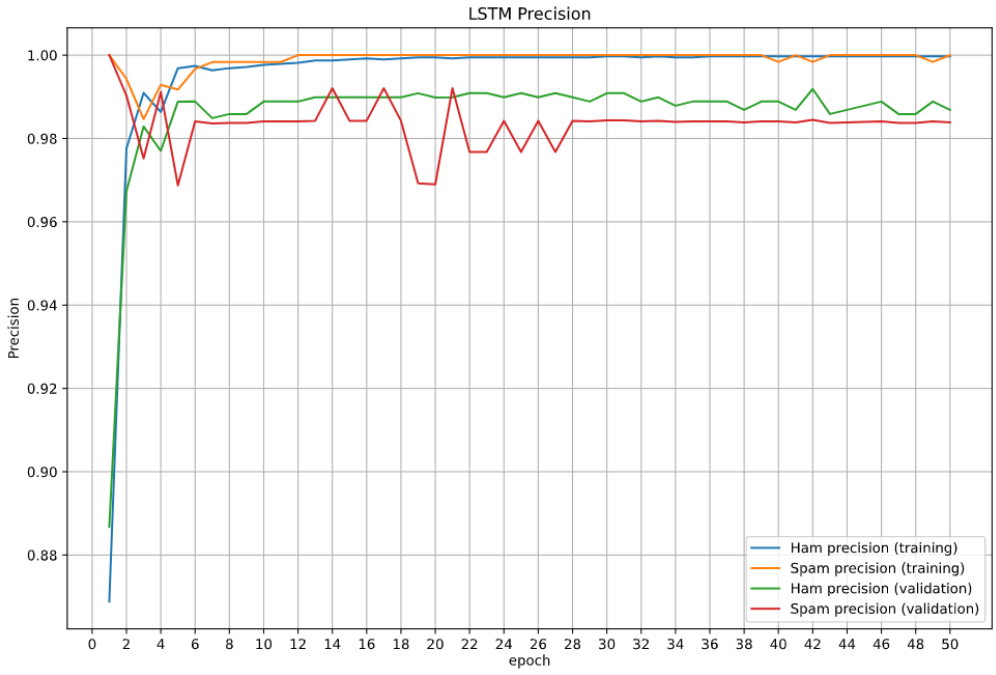
	C	Accuracy	Ham recall	Spam recall	Ham precision	Spam precision	Ham f1	Spam f1
0	0.5	0.964093	1	0.703704	0.960746	1	0.97998	0.826087
1	1	0.971275	0.998979	0.77037	0.969277	0.990476	0.983903	0.866667
2	5	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
3	10	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
4	20	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
5	50	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
6	100	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
7	200	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
8	500	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246
9	1000	0.974865	0.998979	0.8	0.973134	0.990826	0.985887	0.885246

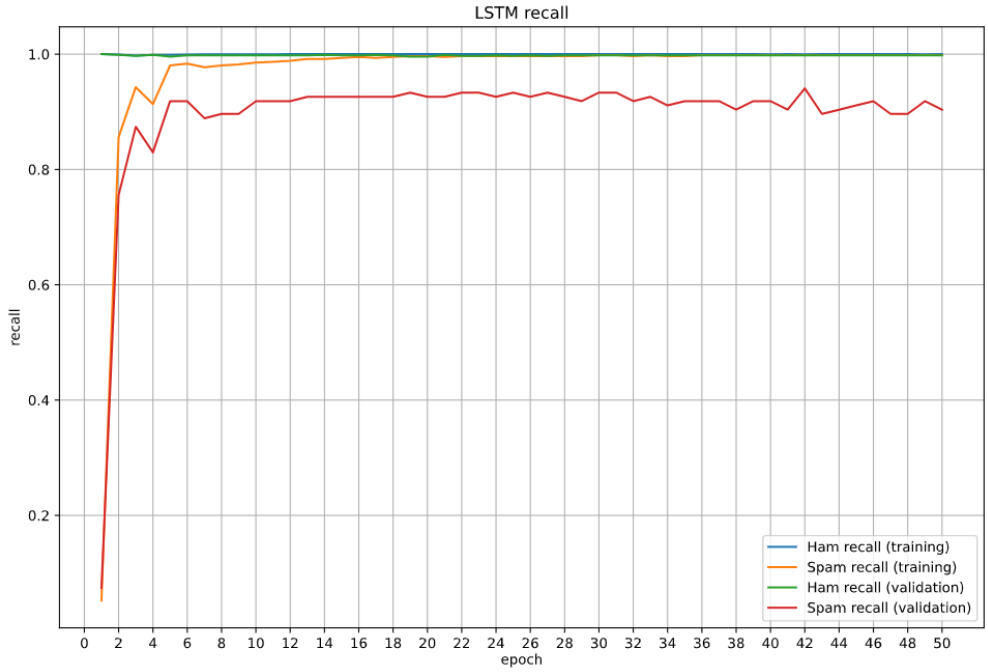
Plot Sumbu Y Tidak Dari 0

Long Short—Term Memory

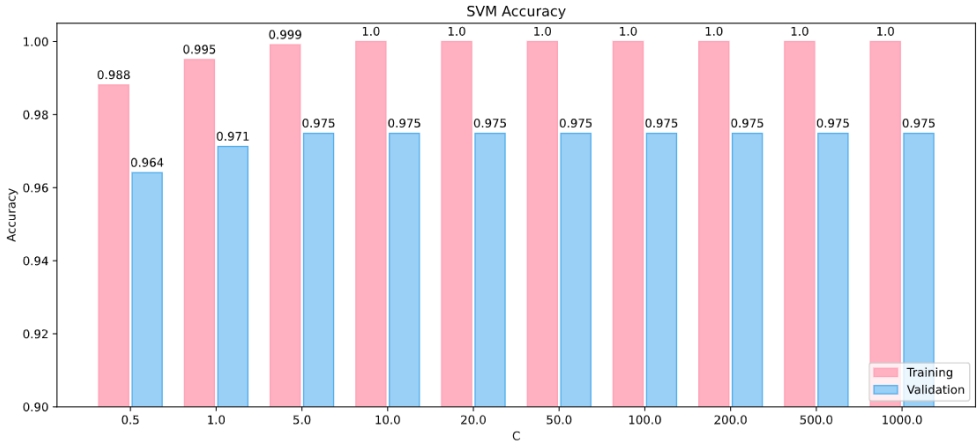


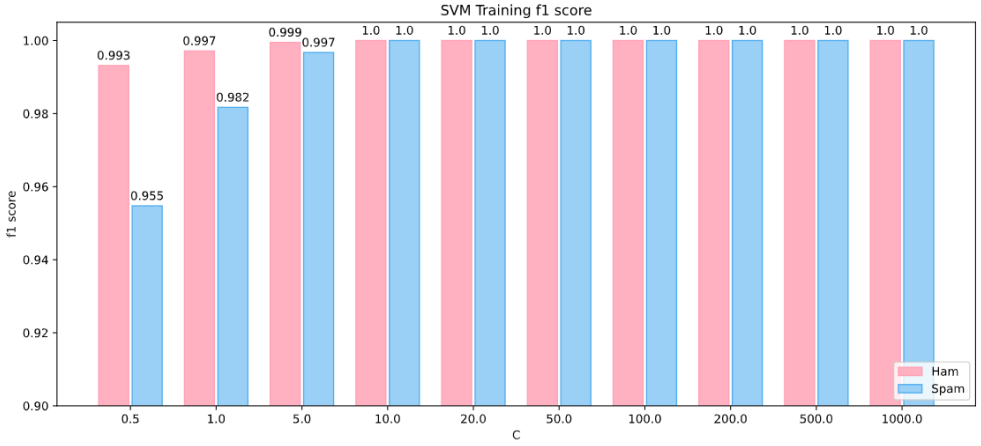

LSTM f1
f1
epoch
Ham f1 (training)
Spam f1 (training)
Ham f1 (validation)
Spam f1 (validation)

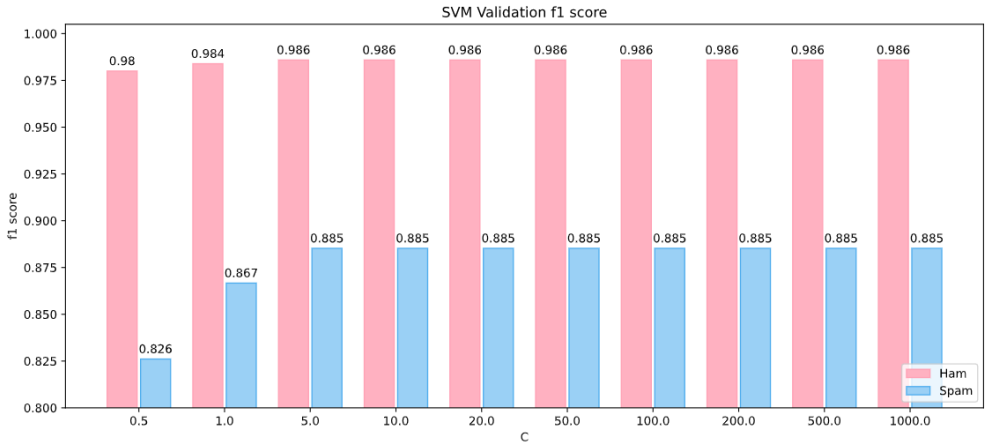

LSTM Precision
Precision
epoch
Ham precision (training)
Spam precision (training)
Ham precision (validation)
Spam precision (validation)

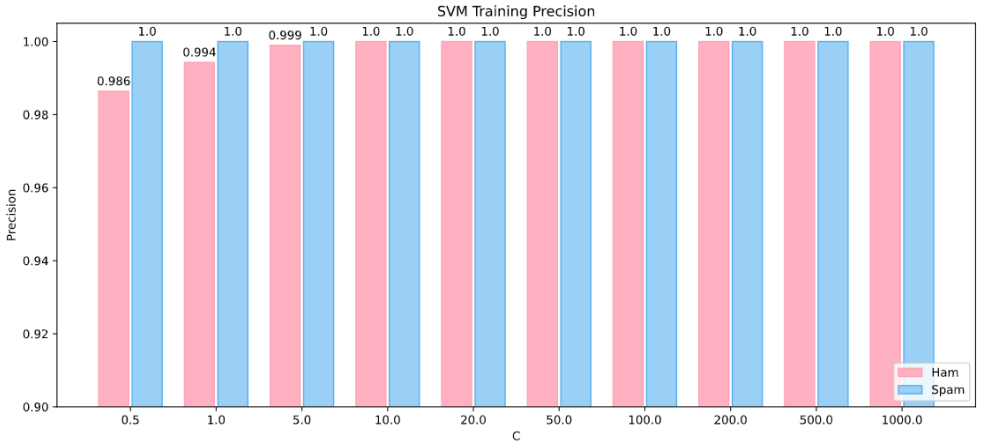


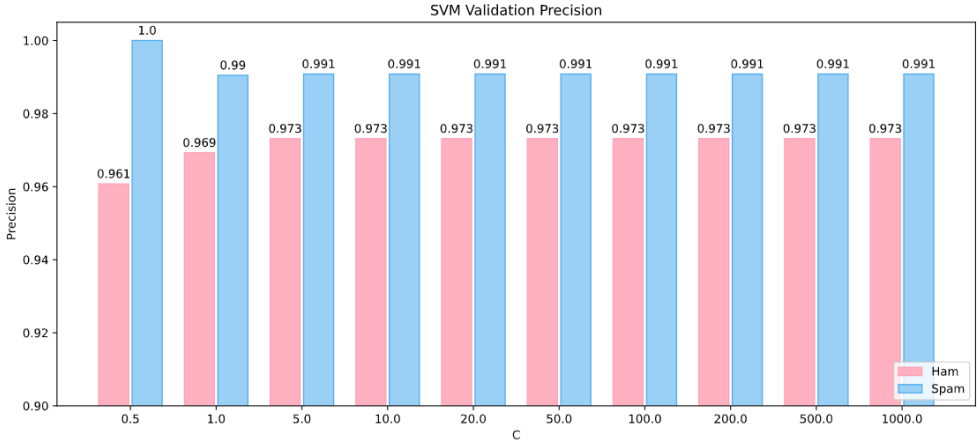
Support Vector Machine

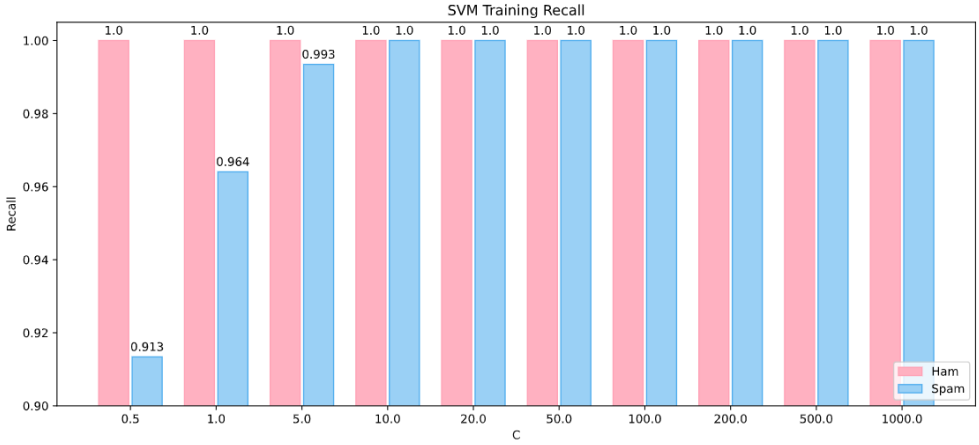


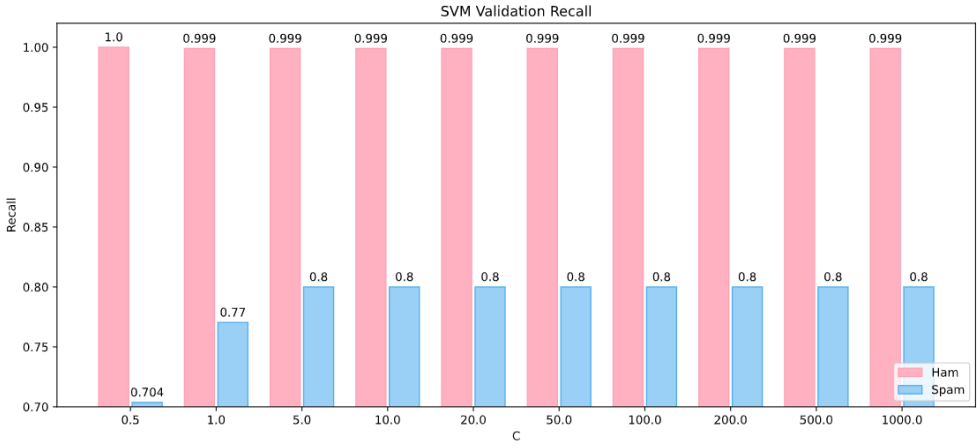

SVM Training f1 score
f1 score
C
Ham
Spam
0.993
0.955
0.997
0.982
0.999
0.997
1.0
1.0
0.5
1.0
5.0
10.0
20.0
50.0
100.0
200.0
500.0
1000.0


SVM Validation f1 score
f1 score
C
Ham
Spam
0.98
0.826
0.984
0.867
0.986
0.885
0.5
1.0
5.0
10.0
20.0
50.0
100.0
200.0
500.0
1000.0

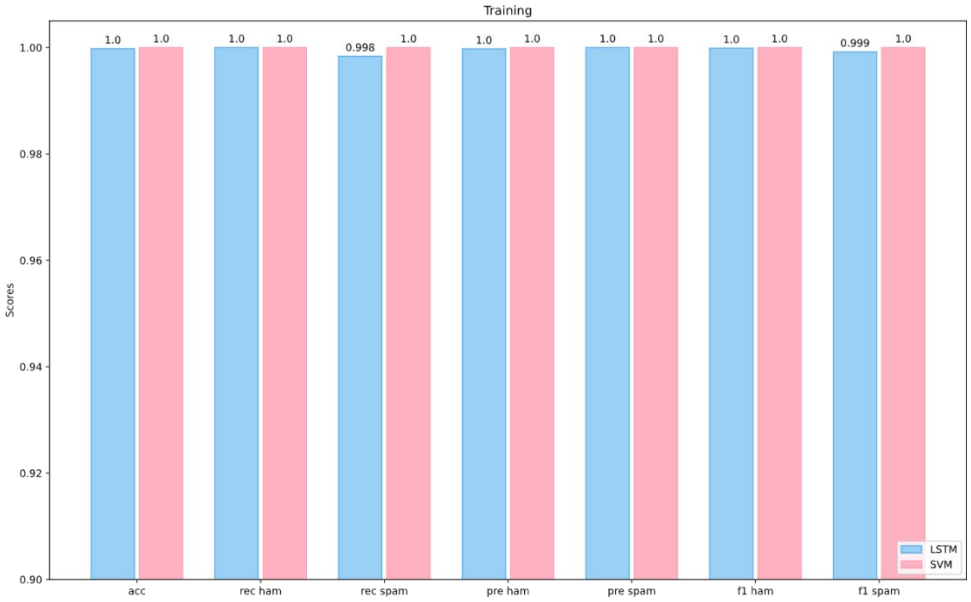

SVM Training Precision
Precision
C
Ham
Spam
0.986
0.994
0.999
1.0
0.5
1.0
5.0
10.0
20.0
50.0
100.0
200.0
500.0
1000.0

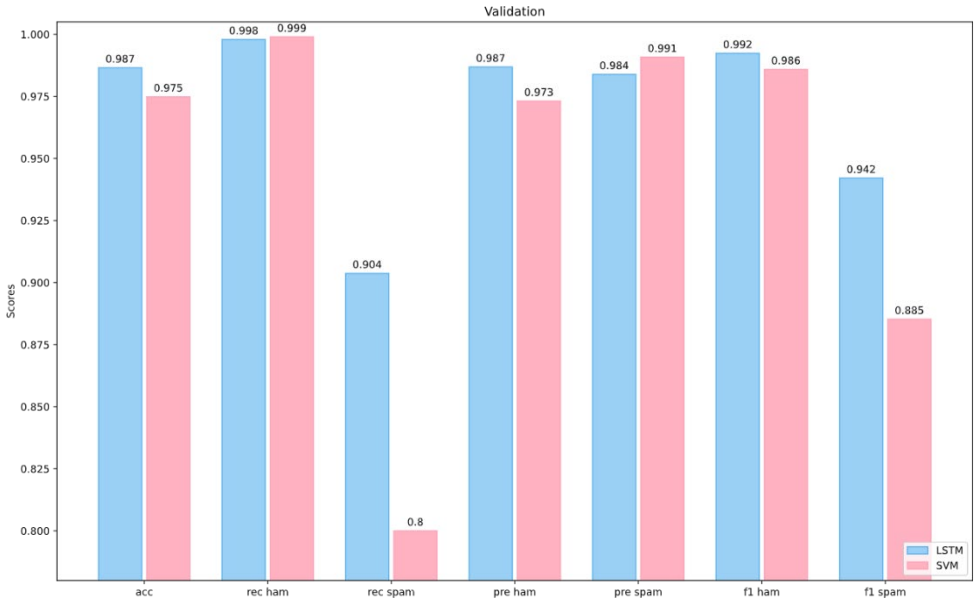






Hasil Training dan Validation LSTM dan SVM





Source Code

Source Code yang digunakan pada pengimplementasian metode *Long Short-Term Memory* dan *Support Vector Machine* dalam mengklasifikasikan spam email.

Mengimport Library

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
import math
from collections import Counter

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn import feature_extraction, svm
from sklearn.metrics import classification_report, precision_recal
l_fscore_support as score

import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense, Sim
pleRNN, LSTM, Bidirectional, Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import Callback
%matplotlib inline

import nltk
import nltk.classify.util
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, RegexpTokenizer

nltk.download('stopwords')

seed = 101
np.random.seed(seed)
```

```
tf.random.set_seed(seed)
```

Membaca Data

```
df = pd.read_csv('Dataset.csv',delimiter=',',encoding='latin-1')
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=
True)
df.head(10)
df.info()
```

Menampilkan Plot Data

```
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

Preprocessing

Missing Value

```
df.isna().sum()
```

Label Encoding

```
print(df['v1'][0:5])
df['v1'] = df['v1'].map({'spam':1,'ham':0})
print('Encoded:')
print(df['v1'][0:5])
```

LowerCase, Tokenizer,* dan *Stop Word

```
def string_unlist(strlist):
    return " ".join(strlist)
tokenizer = RegexpTokenizer(r'[a-zA-Z]{2,}')
stop_words = set(stopwords.words('english'))


df['tokenized'] = df['v2'].astype(str).str.lower() # Turn into low
er case text
df['tokenized'] = df.apply(lambda row: tokenizer.tokenize(row['tok
enized']), axis=1) # Apply tokenize to each row
```

```
df['tokenized'] = df['tokenized'].apply(lambda x: [w for w in x if
 not w in stop_words]) # Remove stopwords from each row
df["tokenized_unlist"] = df["tokenized"].apply(string_unlist)# tok
enized but joined with spaces
```

Text to Sequence, Padding

```
samples = df['tokenized']
maxlen = 150
max_words = 2000
tokenizer2 = Tokenizer(num_words=max_words)
tokenizer2.fit_on_texts(samples)
sequences = tokenizer2.texts_to_sequences(samples)

word_index = tokenizer2.word_index
print('Found %s unique tokens.' % len(word_index))
padded = pad_sequences(sequences, maxlen=maxlen)
#df['preprocessed'] = padded
word_index
df['preprocessed'] = ''*len(df)
for i in range(len(df)):
    df.at[i,'preprocessed'] = padded[i]
```

Menampilkan Hasil *LowerCse, Tokenizer, StopWord, Text to Sequence* dan *Padding*

```
# original text
df['v2'].loc[2]
# after lowercase
df['v2'].astype(str).str.lower()[2]
# after tokenizer
tokenizer.tokenize(df['v2'].astype(str).str.lower()[2])
# after removing stop words
df['tokenized'][2]
# after text to sequence
sequences[2]
# after padding (length 150)
df['preprocessed'][2]
```

Count Vectorizer

```
f = feature_extraction.text.CountVectorizer(stop_words = 'english'
)
print(f.get_feature_names())
print(len(f.get_feature_names()))
```

Pembagian Data

```
lstm_X = padded
svm_X = f.fit_transform(df['tokenized_unlist'])
y = df['v1']
# lstm
lstm_X_train, lstm_X_val, lstm_y_train, lstm_y_val = train_test_sp
lit(lstm_X, y, test_size=0.20, random_state=seed)
# svm
svm_X_train, svm_X_val, svm_y_train, svm_y_val = train_test_split(
svm_X, y, test_size=0.20, random_state=seed)
```

Modeling

```
# callback to find metrics at epoch end
class Metrics(Callback):
    def __init__(self, x, y):
        self.x = x
        self.y = y if (y.ndim == 1 or y.shape[1] == 1) else np.arg
max(y, axis=1)
        self.reports = []
        self.confusion_matrices = []

    def on_epoch_end(self, epoch, logs={}):
        y_hat = np.asarray(self.model.predict(self.x))
        y_hat = np.where(y_hat > 0.5, 1, 0) if (y_hat.ndim == 1 or
 y_hat.shape[1] == 1)  else np.argmax(y_hat, axis=1)
        report = classification_report(self.y,y_hat,output_dict=Tr
ue)
        self.reports.append(report)

        _confusion_matrix = confusion_matrix(self.y,y_hat)
```

```
        self.confusion_matrices.append(_confusion_matrix)
        return

    # Utility method
    def get(self, metrics, of_class):
        return [report[str(of_class)][metrics] for report in self.
reports]

    def get_cm(self):
        return self.confusion_matrices
```

Long Short Term Memory

```
EPOCHS = 50
BATCH_SIZE = 128

def lstm():
    model = Sequential()
    model.add(Embedding(max_words, 50,input_length=maxlen)) # inpu
t layer
    model.add(LSTM(64)) # hidden layer
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid')) # output layer
    model.compile(optimizer='adam', loss='binary_crossentropy', me
trics=['acc'])
    return model

lstm = lstm()
lstm.summary()
```

Pelatihan dan Evaluasi Metode *Long Short-Term Memory*

```
metrics_binary_train = Metrics(lstm_X_train, lstm_y_train)
metrics_binary_val = Metrics(lstm_X_val, lstm_y_val)
time1 = time.time()
history = lstm.fit(lstm_X_train, lstm_y_train, epochs=EPOCHS, batc
h_size=BATCH_SIZE, validation_data=(lstm_X_val, lstm_y_val),callba
cks=[metrics_binary_train, metrics_binary_val])
time2 = time.time()
computation_time_lstm = time2 - time1
```

Hasil *Training Long Short-Term Memory*

```
lstm_train_result = pd.DataFrame([history.history['acc'], metrics_
binary_train.get('recall',0),metrics_binary_train.get('recall',1),
metrics_binary_train.get('precision',0),metrics_binary_train.get('
precision',1),
metrics_binary_train.get('f1-
score',0),metrics_binary_train.get('f1-
score',1)],index=['Accuracy','Ham recall', 'Spam recall', 'Ham pre
cision', 'Spam precision', 'Ham f1', 'Spam f1']).T
lstm_train_result.to_excel('lstm_train_result.xlsx')
lstm_train_result
```

Hasil *Validation Long Short-Term Memory*

```
lstm_val_result = pd.DataFrame([history.history['val_acc'], metric
s_binary_val.get('recall',0),metrics_binary_val.get('recall',1),
metrics_binary_val.get('precision',0),metrics_binary_val.get('prec
ision',1),
metrics_binary_val.get('f1-score',0),metrics_binary_val.get('f1-
score',1)],index=['Accuracy','Ham recall', 'Spam recall', 'Ham pre
cision', 'Spam precision', 'Ham f1', 'Spam f1']).T
lstm_val_result.to_excel('lstm_val_result.xlsx')
lstm_val_result
```

***Precision* plot**

```
fig, ax = plt.subplots(figsize=(12,8))
ax.axes.set_ylim(0, 1.005)
plt.plot(range(1, EPOCHS+1), lstm_train_result['Ham precision'])
plt.plot(range(1, EPOCHS+1), lstm_train_result['Spam precision'])
plt.plot(range(1, EPOCHS+1), lstm_val_result['Ham precision'])
plt.plot(range(1, EPOCHS+1), lstm_val_result['Spam precision'])
plt.title('LSTM Precision')
plt.ylabel('Precision')
plt.xlabel('epoch', labelpad=2)
plt.xticks(np.arange(0, EPOCHS+1, 2))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.legend(['Ham precision (training)', 'Spam precision (training)
','Ham precision (validation)', 'Spam precision (validation)'], lo
c='lower right')
plt.grid(True)
```

```
plt.show()
```

***Recall* plot**

```
fig, ax = plt.subplots(figsize=(12,8))
ax.axes.set_ylim(0, 1.005)
plt.plot(range(1, EPOCHS+1), lstm_train_result['Ham recall'])
plt.plot(range(1, EPOCHS+1), lstm_train_result['Spam recall'])
plt.plot(range(1, EPOCHS+1), lstm_val_result['Ham recall'])
plt.plot(range(1, EPOCHS+1), lstm_val_result['Spam recall'])
plt.title('LSTM recall')
plt.ylabel('recall')
plt.xlabel('epoch', labelpad=2)
plt.xticks(np.arange(0, EPOCHS+1, 2))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.legend(['Ham recall (training)', 'Spam recall (training)','Ham
 recall (validation)', 'Spam recall (validation)'], loc='lower rig
ht')
plt.grid(True)
plt.show()
```

F1 *Score* Plot

```
fig, ax = plt.subplots(figsize=(12,8))
ax.axes.set_ylim(0, 1.005)
plt.plot(range(1, EPOCHS+1), lstm_train_result['Ham f1'])
plt.plot(range(1, EPOCHS+1), lstm_train_result['Spam f1'])
plt.plot(range(1, EPOCHS+1), lstm_val_result['Ham f1'])
plt.plot(range(1, EPOCHS+1), lstm_val_result['Spam f1'])
plt.title('LSTM f1')
plt.ylabel('f1')
plt.xlabel('epoch', labelpad=2)
plt.xticks(np.arange(0, EPOCHS+1, 2))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.legend(['Ham f1 (training)', 'Spam f1 (training)','Ham f1 (val
idation)', 'Spam f1 (validation)'], loc='lower right')
plt.grid(True)
plt.show()
```

Accuracy Plot

```
fig, ax = plt.subplots(figsize=(12,8))
ax.axes.set_ylim(0, 1.005)
```

```
plt.plot(range(1, EPOCHS+1), lstm_train_result['Accuracy'], color
='#FFB1C1')
plt.plot(range(1, EPOCHS+1), lstm_val_result['Accuracy'], color='#
9AD0F5')

plt.title('LSTM Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch', labelpad=2)
plt.xticks(np.arange(0, EPOCHS+1, 2))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.legend(['Training', 'Validation'], loc='lower right')
plt.grid(True)
plt.show()
```

Support Vector Machine

```
list_C = [0.5, 1, 5, 10, 20, 50, 100, 200, 500, 1000]

acc_train = np.zeros(len(list_C))
acc_test = np.zeros(len(list_C))

recall_train_0 = np.zeros(len(list_C))
recall_train_1 = np.zeros(len(list_C))
recall_test_0 = np.zeros(len(list_C))
recall_test_1 = np.zeros(len(list_C))

precision_train_0 = np.zeros(len(list_C))
precision_train_1 = np.zeros(len(list_C))
precision_test_0 = np.zeros(len(list_C))
precision_test_1 = np.zeros(len(list_C))

f1_train_0 = np.zeros(len(list_C))
f1_train_1 = np.zeros(len(list_C))
f1_test_0 = np.zeros(len(list_C))
f1_test_1 = np.zeros(len(list_C))

svm_computation_time = np.zeros(len(list_C))
svm_cm_train = []
svm_cm_val = []
svc = []

count = 0
```

```
for C in list_C:
    svc += [svm.SVC(C=C, probability=True)]

    time1 = time.time()
    svc[count].fit(svm_X_train, svm_y_train)
    time2 = time.time()

    acc_train[count] = svc[count].score(svm_X_train, svm_y_train)
    acc_test[count] = svc[count].score(svm_X_val, svm_y_val)

    y_train_pred = svc[count].predict(svm_X_train)
    precision, recall, f1, support = score(svm_y_train, y_train_pr
ed)
    recall_train_0[count] = recall[0]
    recall_train_1[count] = recall[1]
    precision_train_0[count] = precision[0]
    precision_train_1[count] = precision[1]
    f1_train_0[count] = f1[0]
    f1_train_1[count] = f1[1]

    y_val_pred = svc[count].predict(svm_X_val)
    precision, recall, f1, support = score(svm_y_val, y_val_pred)
    recall_test_0[count] = recall[0]
    recall_test_1[count] = recall[1]
    precision_test_0[count] = precision[0]
    precision_test_1[count] = precision[1]
    f1_test_0[count] = f1[0]
    f1_test_1[count] = f1[1]

    svm_computation_time[count] = time2-time1
    svm_cm_train += [confusion_matrix(svm_y_train, y_train_pred)]
    svm_cm_val += [confusion_matrix(svm_y_val, y_val_pred)]
    count = count + 1
```

Hasil *Support Vector Machine*

```
svm_train_result = pd.DataFrame([list_C, acc_train, recall_train_0
, recall_train_1, precision_train_0, precision_train_1, f1_train_0
, f1_train_1],
index=['C', 'Accuracy','Ham recall', 'Spam recall', 'Ham precision
', 'Spam precision', 'Ham f1', 'Spam f1']).T
svm_train_result.to_excel('svm_train_result.xlsx')
```

```
svm_train_result

svm_val_result = pd.DataFrame([list_C, acc_test, recall_test_0, re
call_test_1, precision_test_0, precision_test_1, f1_test_0, f1_tes
t_1],
index=['C', 'Accuracy','Ham recall', 'Spam recall', 'Ham precision
', 'Spam precision', 'Ham f1', 'Spam f1']).T
svm_val_result.to_excel('svm_val_result.xlsx')
svm_val_result

def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying i
ts height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(round(height,3)),
                    xy=(rect.get_x() + rect.get_width() / 2, heigh
t),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

labels = svm_train_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
rects1 = ax.bar(x - 0.02 - width/2, svm_train_result['Accuracy'],
width, label='Training', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x + 0.02 + width/2, svm_val_result['Accuracy'], wi
dth, label='Validation', color='#9AD0F5', edgecolor='#4FADEE')
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0.95, 1.005)
#ax.axes.margins(0.05, 0.75)
ax.set_ylabel('Accuracy')
ax.set_xlabel('C')
ax.set_title('SVM Accuracy')
ax.set_xticks(x)
```

```
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);

labels = svm_train_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
rects1 = ax.bar(x - 0.02 - width/2, svm_train_result['Ham recall']
, width, label='Ham', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x  + 0.02 + width/2, svm_train_result['Spam recall
'], width, label='Spam', color='#9AD0F5', edgecolor='#4FADEE')
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0, 1.005)
#ax.axes.margins(0.05, 0.05)
ax.set_ylabel('Recall')
ax.set_xlabel('C')
ax.set_title('SVM Training Recall')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);

labels = svm_train_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
rects1 = ax.bar(x - 0.02 - width/2, svm_train_result['Ham precisio
n'], width, label='Ham', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x  + 0.02 + width/2, svm_train_result['Spam precis
ion'], width, label='Spam', color='#9AD0F5', edgecolor='#4FADEE')
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0.95, 1.005)
```

```
#ax.axes.margins(0.05, 0.75)
ax.set_ylabel('Precision')
ax.set_xlabel('C')
ax.set_title('SVM Training Precision')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);

labels = svm_train_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
rects1 = ax.bar(x - 0.02 - width/2, svm_train_result['Ham f1'], wi
dth, label='Ham', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x + 0.02 + width/2, svm_train_result['Spam f1'], w
idth, label='Spam', color='#9AD0F5', edgecolor='#4FADEE')
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0.92, 1.005)
#ax.axes.margins(0.05, 0.75)
ax.set_ylabel('f1 score')
ax.set_xlabel('C')
ax.set_title('SVM Training f1 score')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);

labels = svm_val_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
rects1 = ax.bar(x - 0.02 - width/2, svm_val_result['Ham recall'],
width, label='Ham', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x + 0.02 + width/2, svm_val_result['Spam recall'],
 width, label='Spam', color='#9AD0F5', edgecolor='#4FADEE')
```

```
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0.60, 1.005)
ax.set_ylabel('Recall')
ax.set_xlabel('C')
ax.set_title('SVM Validation Recall')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);

labels = svm_val_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
rects1 = ax.bar(x - 0.02 - width/2, svm_val_result['Ham precision'
], width, label='Ham', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x + 0.02 + width/2, svm_val_result['Spam precision
'], width, label='Spam', color='#9AD0F5', edgecolor='#4FADEE')
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0.95, 1.005)
#ax.axes.margins(0.05, 0.75)
ax.set_ylabel('Precision')
ax.set_xlabel('C')
ax.set_title('SVM Validation Precision')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);

labels = svm_val_result['C']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(14,6))
```

```
rects1 = ax.bar(x - 0.02 - width/2, svm_val_result['Ham f1'], widt
h, label='Ham', color ='#FFB1C1',edgecolor='#FFA1B5')
rects2 = ax.bar(x + 0.02 + width/2, svm_val_result['Spam f1'], wid
th, label='Spam', color='#9AD0F5', edgecolor='#4FADEE')
# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
autolabel(rects2)
#ax.axes.set_ylim(0.72, 1.005)
#ax.axes.margins(0.05, 0.75)
ax.set_ylabel('f1 score')
ax.set_xlabel('C')
ax.set_title('SVM Validation f1 score')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.legend(loc='lower right')
ax.set_xticklabels(labels);
```

Perbandingan LSTM dan SVM

```
BEST_EPOCH = np.argmax(lstm_val_result['Accuracy'])
BEST_C = 10 #10, 20, 50 same

labels = ['acc', 'rec ham', 'rec spam', 'pre ham', 'pre spam', 'f1
 ham', 'f1 spam']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(16,10))
rects1 = ax.bar(x - 0.02 - width/2, lstm_train_result.loc[BEST_EPO
CH-1], width, label='LSTM', color='#9AD0F5', edgecolor='#4FADEE')
rects2 = ax.bar(x + 0.02 + width/2, svm_train_result.set_index('C'
).loc[BEST_C], width, label='SVM', color ='#FFB1C1',edgecolor='#FF
A1B5')
# Add some text for labels, title and custom x-
axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Training')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.set_xticklabels(labels)
```

```
ax.legend(loc='lower right')
autolabel(rects1)
autolabel(rects2)


labels = ['acc', 'rec ham', 'rec spam', 'pre ham', 'pre spam', 'f1
 ham', 'f1 spam']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots(figsize=(16,10))
rects1 = ax.bar(x - 0.02 - width/2, lstm_val_result.loc[BEST_EPOCH
-1], width, label='LSTM', color='#9AD0F5', edgecolor='#4FADEE')
rects2 = ax.bar(x + 0.02 +  width/2, svm_val_result.set_index('C')
.loc[BEST_C], width, label='SVM', color ='#FFB1C1',edgecolor='#FFA
1B5')
# Add some text for labels, title and custom x-
axis tick labels, etc.
#ax.set_ylim(0.75, 1)
ax.set_ylabel('Scores')
ax.set_title('Validation')
ax.set_xticks(x)
ax.set_yticks(np.arange(0, 1.1, 0.1))
ax.set_xticklabels(labels)
ax.legend(loc='lower right')
autolabel(rects1)
autolabel(rects2)
```

Confusion Matrix LSTM

```
ax =  plt.subplot()
cmap = sns.dark_palette('#9AD0F5', as_cmap=True)
sns.heatmap(metrics_binary_train.get_cm()[BEST_EPOCH-
1], annot=True, fmt='g', ax=ax, cmap=cmap)
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('LSTM Training (Epoch '+str(BEST_EPOCH)+')');
ax.xaxis.set_ticklabels(['ham', 'spam']); ax.yaxis.set_ticklabels(
['ham', 'spam']);

ax =  plt.subplot()
cmap = sns.dark_palette('#9AD0F5', as_cmap=True)
```

```
sns.heatmap(metrics_binary_val.get_cm()[BEST_EPOCH-
1], annot=True, fmt='g', ax=ax, cmap=cmap)
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('LSTM Validation (Epoch '+str(BEST_EPOCH)+')');
ax.xaxis.set_ticklabels(['ham', 'spam']); ax.yaxis.set_ticklabels(
['ham', 'spam']);
```

Confusion Matrix SVM

```
ax =  plt.subplot()
cmap = sns.dark_palette('#FFB1C1', as_cmap=True)
sns.heatmap(svm_cm_train[np.where(svm_train_result['C'] == BEST_C)
[0][0]], annot=True, fmt='g', ax=ax, cmap=cmap)
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('SVM Training (C = '+str(BEST_C)+')');
ax.xaxis.set_ticklabels(['ham', 'spam']); ax.yaxis.set_ticklabels(
['ham', 'spam']);
ax =  plt.subplot()
cmap = sns.dark_palette('#FFB1C1', as_cmap=True)
sns.heatmap(svm_cm_val[np.where(svm_val_result['C'] == BEST_C)[0][
0]], annot=True, fmt='g', ax=ax, cmap=cmap)
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('SVM Validation (C = '+str(BEST_C)+')');
ax.xaxis.set_ticklabels(['ham', 'spam']); ax.yaxis.set_ticklabels(
['ham', 'spam']);
```

Computation Time

```
lstm_computation_time = time2-time1
print('lstm computation time (%d epochs): %f seconds' % (EPOCHS,ls
tm_computation_time))

svm_computation_time

labels = ['LSTM', 'SVM']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars
```

```
fig, ax = plt.subplots(figsize=(8,6))
rects1 = ax.bar(x, [lstm_computation_time, np.max(svm_computation_
time)], width)

# Add some text for labels, title and custom x-
axis tick labels, etc.
autolabel(rects1)
ax.set_ylabel('Seconds')
ax.set_xlabel('Model')
ax.set_title('Computation Time')
ax.set_xticks(x)
ax.set_xticklabels(labels);
```

Kurva ROC

```
fpr, tpr, thresholds = roc_curve(lstm_y_val, lstm.predict(lstm_X_v
al))
fpr2, tpr2, thresholds = roc_curve(svm_y_val, svc[list_C.index(BES
T_C)].predict_proba(svm_X_val).T[1])
plt.subplots(figsize=(8,8))
plt.plot(fpr, tpr, color='#9AD0F5', label='LSTM (area = %0.4f)' %
auc(fpr, tpr))
plt.plot(fpr2, tpr2, color='#FFB1C1', label='SVM (area = %0.4f)' %
 auc(fpr2, tpr2))
plt.xlim([0.0, 1.0]);plt.ylim([0.0, 1.05]);
plt.xlabel('False Positive Rate');plt.ylabel('True Positive Rate')
;
plt.yticks(np.arange(0, 1.1, 0.1))
plt.legend(loc="lower right")
plt.title('ROC Curve')
```