Tugas Akhir Pemrosesan Bahasa Alami

Text Classification for Detecting Spam Message

Kelompok 9

Rahma Maesarah
19/439820/TK/48550

Dimas Mahendra N
19/444048/TK/49244

Rizky Alif R
19/446785/TK/49890

Outline

1. Background and Purpose of the NLP system.
2. Data Understanding and Preprocessing
3. AI Modeling
4. Conclusion & Recommendation

Background

Spam yang sering muncul pada pesan dapat mengganggu aktivitas dan dapat menimbun pesan-pesan penting yang seharusnya dibaca terlebih dahulu.

&

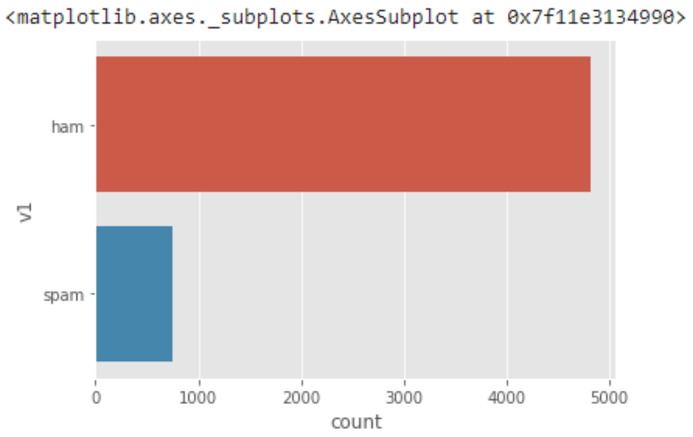
Purpose

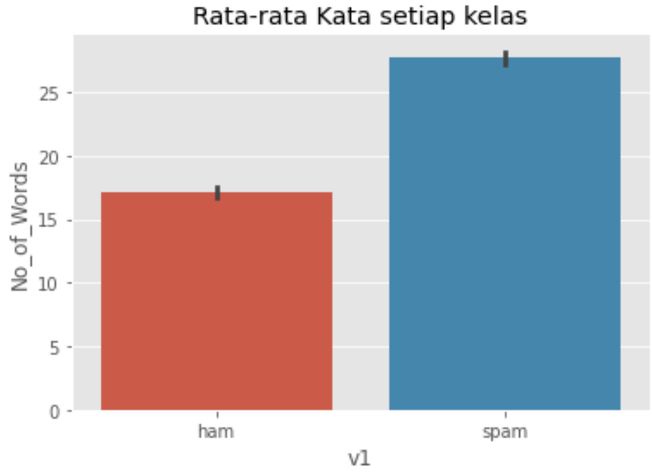
Membuat sebuah model yang dapat digunakan untuk mengotomatisasi aplikasi perpesanan untuk dapat mengkategorikan suatu pesan termasuk ke dalam spam atau tidak.

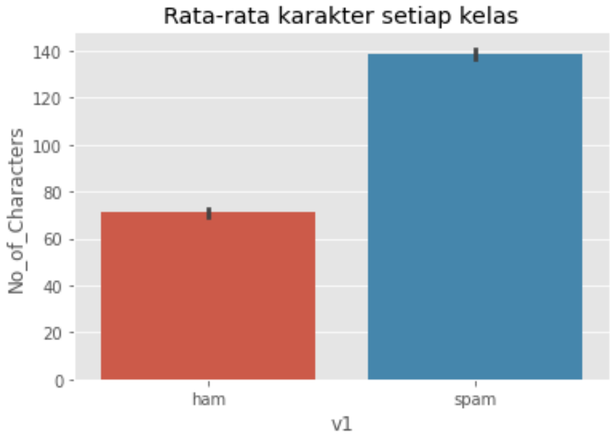
Data Understanding

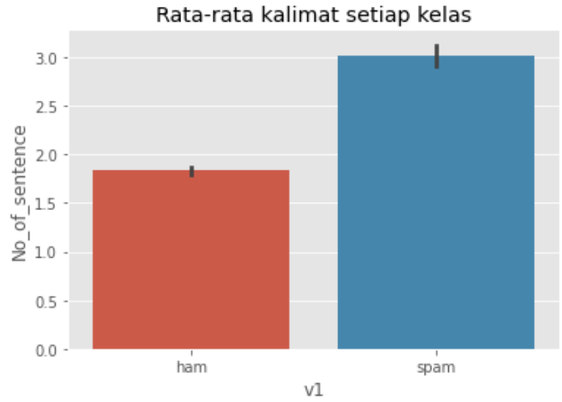
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns









https://www.kaggle.com/uciml/sms-spam-collection-dataset

Data Preprocessing

- Removing noise (symbol, numeric, etc) and Lowering Case

```
# Removing noise
def Clean(Text):
    sms = re.sub('[^a-zA-Z]', ' ', Text) #Replacing all non-alphabetic characters with a space
    sms = sms.lower() #converting to lowecase
    sms = sms.split()
    sms = ' '.join(sms)
    return sms

data["Clean_Text"] = data["v2"].apply(Clean)
```

- Spelling Correction [autocorrect]

```
# Spelling correction
from autocorrect import Speller
spell = Speller(lang='en')

correct = []
for x in data["Clean_Text"] :
  x = spell(x)
  correct.append(x)

data["Clean_Text"] = np.array(correct)
```

Data Preprocessing

- Removing Stopword

```
#Removing stopword
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize.treebank import TreebankWordDetokenizer

data["Tokenize_Text"]=data.apply(lambda row: nltk.word_tokenize(row["Clean_Text"]), axis=1)

def remove_stopwords(text):
    stop_words = set(stopwords.words("english"))
    filtered_text = [word for word in text if word not in stop_words]
    sentence = TreebankWordDetokenizer().detokenize(filtered_text)
    return sentence

data["Clean_Text"] = data["Tokenize_Text"].apply(remove_stopwords)
```

- Removing word with one char (ex : I, u, a)

```
#Removing word with one char (I, u,)
def del_one_char(input):
  s = ' '.join( [w for w in input.split() if len(w)>1] )
  return s

data["Clean_Text"] = data["Clean_Text"].apply(del_one_char)
```

Data Preprocessing

- Lemmatization

```
#Lemmatization
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.tokenize.treebank import TreebankWordDetokenizer

lemmatizer = WordNetLemmatizer()

sentence_stemm = []
for i in range(data.shape[0]):
    word_data = data.Clean_Text[i]
    nltk_tokens = nltk.word_tokenize(word_data)
    word=[]
    for w in nltk_tokens:
        word.append(lemmatizer.lemmatize(w))
    sentence = TreebankWordDetokenizer().detokenize(word)
    sentence_stemm.append(sentence)

data["Clean_Text"] = np.array(sentence_stemm)
```

- Label Encoder

```
#Label Encoder
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
data['LabelEnc'] = labelencoder.fit_transform(data['v1'])
```

- Tf-idf

```
#tfidf
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, stop_words='english')

# We transform each text into a vector
features_train = tfidf.fit_transform(data.Clean_Text).toarray()
features_train_name = tfidf.get_feature_names()

labels = data.LabelEnc

print("Jumlah Feature Setelah di Ekstrak : "+str(features_train.shape[1]))
```

5572 rows × 1434 columns

AI Modelling

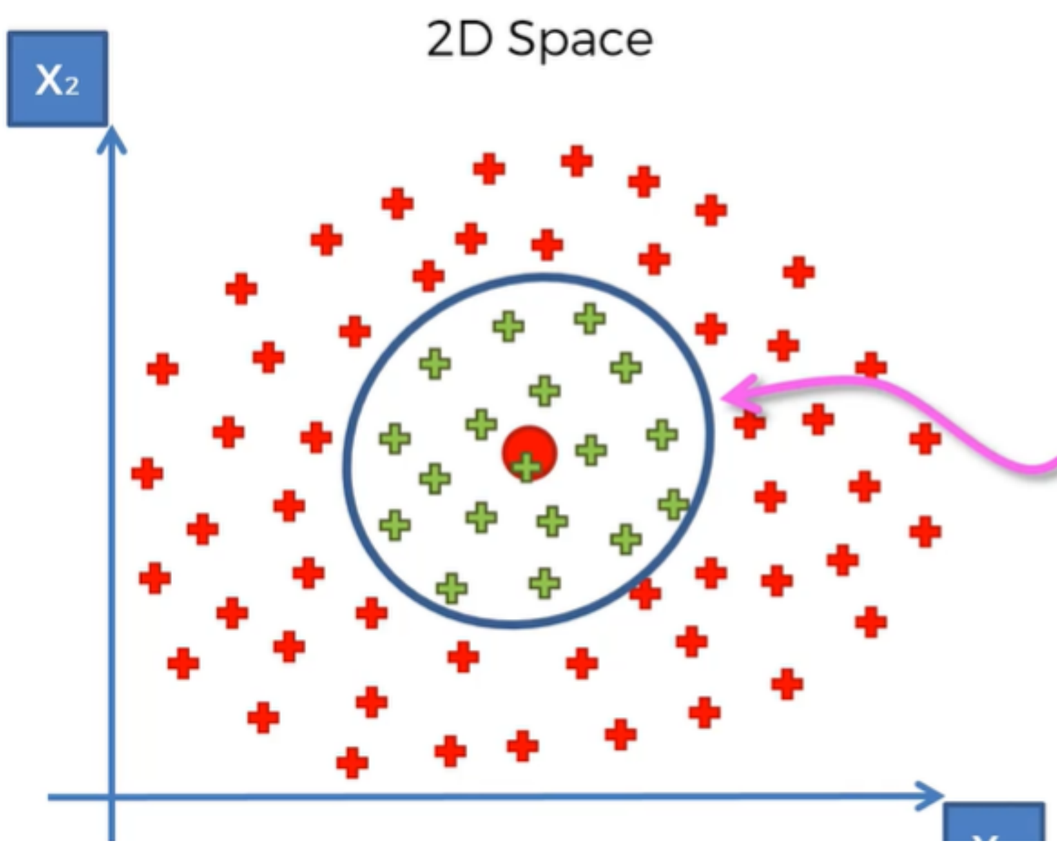
Support Vector Machine

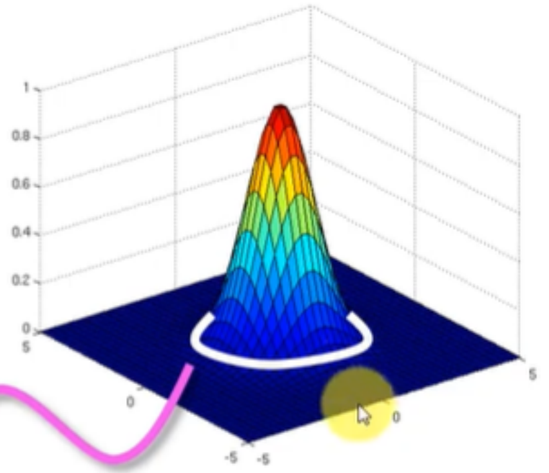
karena SVM cocok digunakan untuk binary classification, selain itu metode klasik ini juga mempunyai run time yang cepat dan penggunaan memory yang irit. SVM bekerja relatif baik ketika ada margin pemisahan yang jelas antar kelas.

How its works?

- support vector machine bekerja dengan cara memisahkan 2 class menggunakan hyper-plane
- Disini kita menggunakan trik kernel, kernel SVM merupakan fungsi yang ruang input berdimensi rendah dan mengubahnya menjadi ruang berdimensi lebih tinggi

AI Modelling





$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^{2}}{2\sigma^{2}}}$$

AI Modelling

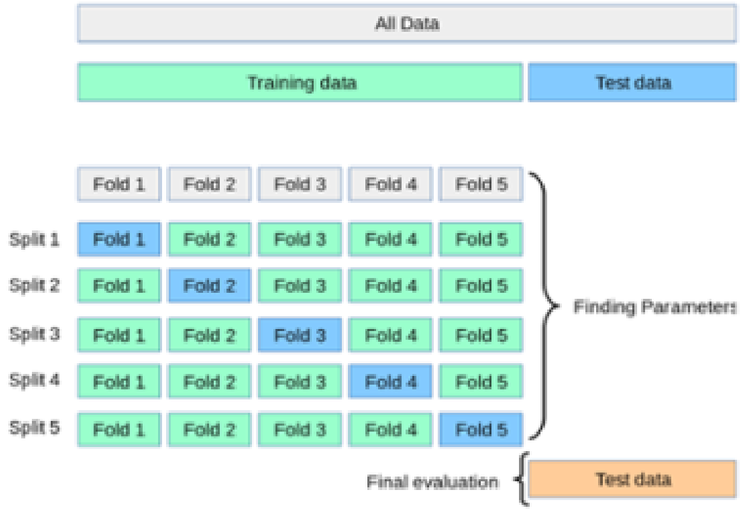
Hyperparameter tuning

Memilih satu set hiperparameter optimal untuk algoritma pembelajaran, untuk argumen model, karena kami menggunakan SVM kami menggunakan konstanta Regularisasi, tipe kernel, dan konstanta. jadi kita bisa mendapatkan akurasi yang lebih tinggi

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
SVC(C=10, gamma=0.1)
```

AI Modelling

Cross-validation



F1 is calculated as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

where:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

In "macro" F1 a separate F1 score is calculated for each `species` value and then averaged.

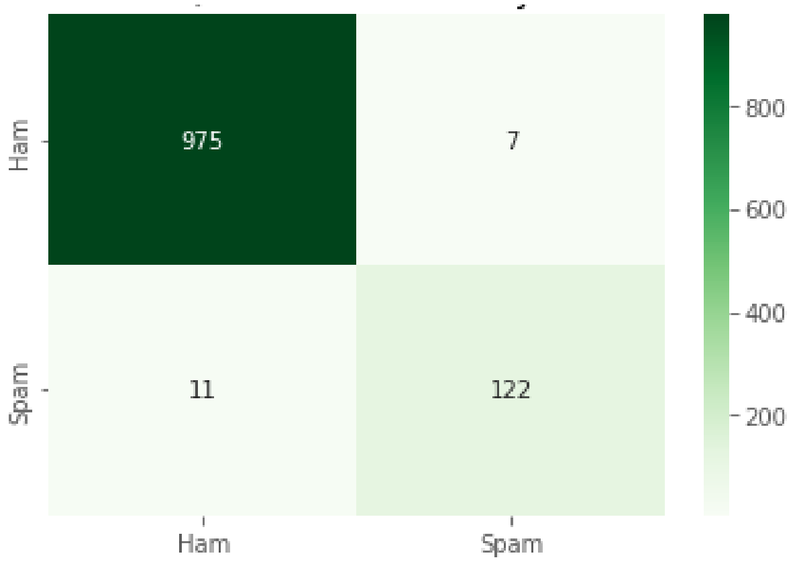
model_name	Mean F1_Macro	Standard deviation
SVC	0.957419	0.00453

AI Modelling

Testing

```
#Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(features_train, labels, test_size=0.2, random_state=123)
```

	precision	recall	f1-score	support
Ham	0.99	0.99	0.99	982
Spam	0.95	0.92	0.93	133
accuracy			0.98	1115
macro avg	0.97	0.96	0.96	1115
weighted avg	0.98	0.98	0.98	1115



AI Modelling

Input-output in example

```
def svm_sc(text):
  text = Clean(text)
  print("clean: " + text)

  text = spell(text)
  print("spell: " +text)

  text = nltk.word_tokenize(text)
  text = remove_stopwords(text)
  print("stopword remove: " +text)

  text = del_one_char(text)
  print("del_one_char: " +text)

  nltk_tokens = nltk.word_tokenize(text)

  word=[]
  for w in nltk_tokens:
      word.append(lemmatizer.lemmatize(w))
  text = TreebankWordDetokenizer().detokenize(word)
  print("Lemmatization: "+text)

  text = [text]
  text = tfidf.transform(text).toarray()

  predict = model.predict(text)

  if predict == 1 :
    output = "Spam"
  if predict == 0 :
    output = "Ham"
  return print("Text tersebut termasuk " + output)
```

```
text1 = "Halo, hw are u today, i hope u re OK :)"
svm_sc(text1)
```

```
clean: halo hw are u today i hope u re ok
spell: halo hw are u today i hope u re ok
stopword remove: halo hw u today hope u ok
del_one_char: halo hw today hope ok
Lemmatization: halo hw today hope ok
Text tersebut termasuk Ham
```

```
text2 = 'URGENT Your grandson was arrested last night in Mexico. Need bail money immediately Western Union Wire $9,500 http://goo.gl/ndf4g5'
svm_sc(text2)
```

```
clean: urgent your grandson was arrested last night in mexico need bail money immediately western union wire http goo gl ndf g
spell: urgent your grandson was arrested last night in mexico need bail money immediately western union wire http goo gl df g
stopword remove: urgent grandson arrested last night mexico need bail money immediately western union wire http goo gl df g
del_one_char: urgent grandson arrested last night mexico need bail money immediately western union wire http goo gl df
Lemmatization: urgent grandson arrested last night mexico need bail money immediately western union wire http goo gl df
Text tersebut termasuk Spam
```

Conclusion & Recommendation

- SVM Classifer dapat mengkategorikan sebuah Text Message apakah text tersebut spam atau bukan dengan sangat baik, hal ini terbukti dengan skor validasi yang didapat yaitu f1 score = 0.9574
- Model ini juga mampu mengkategorisasikan Text Message diluar dataset yang ada, hal ini terbukti dari 2 contoh text yang tidak berada pada dataset tersebut
- Untuk memperbesar akurasi dan presisi, penambahan dataset diperlukan, dengan memperbesar dataset diharapkan model dapat lebih mengenali lebih banyak Text Message sehingga akan lebih variatif. Dengan data yang lebih bervariasi tersebut akan memperbesar akurasi dan presisi.