

Pengklasifikasi Spam pada E-mail

MAKALAH LAPORAN PROJECT



DISUSUN OLEH :

KELOMPOK 6

FADHILAH NURIA SHINTA

22031554003

ANALICIA

22031554007

RIVA DIAN ARDIANSYAH

22031554043

S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2023

DAFTAR ISI

BAB I.....	3
PENDAHULUAN	3
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah	3
1.3 Tujuan dan Manfaat	3
BAB II	4
METODE.....	4
2.1 SPACY	4
2.2 WORD2VEC	4
2.3 TF – IDF	4
BAB III.....	5
HASIL DAN ANALISIS	5
3.1 USER MANUAL GUIDE	5
3.2 LISTING CODE	6
3.3 HASIL AKURASI	15
BAB IV	17
KESIMPULAN	17
DAFTAR PUSTAKA	18
LAMPIRAN.....	19

BAB I

PENDAHULUAN

1.1 Latar Belakang

Email merupakan alat pengirim pesan dengan perantara teknologi komputer, laptop bahkan smartphone yang terhubung dengan kecepatan jaringan internet dan murah. Email spam sendiri merupakan pesan elektronik yang tidak diinginkan oleh pengguna dan masuk dalam jumlah besar kepada siapa pun sehingga memerlukan suatu klasifikasi berbasis teks supaya dengan mudah mendeteksi suatu email tersebut merupakan pesan berupa spam atau non-spam.

Klasifikasi spam pada email adalah tindakan penting dalam menjaga keamanan komunikasi online. Dengan mengidentifikasi dan memblokir pesan spam, kita melindungi diri dari potensi penipuan, menjaga keamanan data pribadi, dan meningkatkan efisiensi dalam pengelolaan email. Namun nyatanya, seringkali pesan-pesan yang seharusnya bukan spam juga teridentifikasi sebagai spam, yang mana hal ini mengganggu pengalaman pengguna. Masalah dalam klasifikasi spam berupa teks pada email ialah dengan melakukan pengklasifikasian menggunakan algoritma agar dalam pemfilteran untuk memisahkan teks tersebut berupa spam atau tidak menjadi lebih efisien. Dengan menggunakan beberapa algoritma, yaitu *K-Nearest Neighbour(KNN)*, *Support Vector Machine(SVM)*, dan *Naives Bayes* dan juga beberapa metode pemfilteran diantaranya Word2Vec dan TF-IDF yang digunakan untuk pengklasifikasian pada spam email tersebut.

1.2 Rumusan Masalah

1. Apakah tetap terdapat email yang terdeteksi spam meskipun termasuk pada tidak spam?
2. Diantara ketiga algoritma tersebut manakah yang paling bagus digunakan untuk pengklasifikasian spam?
3. Metode ekstraksi fitur apakah yang menghasilkan akurasi paling baik untuk proses klasifikasi?

1.3 Tujuan dan Manfaat

Ada pun project ini dilakukan ialah bertujuan untuk membersihkan data email dari macam jenis noise atau informasi yang tidak relevan dan mempersiapkan data email dengan menggunakan model klasifikasi email berdasarkan kriteris tertentu (spam dan non spam) untuk menggambarkan hubungan nantara kata-kata yang digunakan untuk analisis lebih lanjut.

Dalam hal ini dapat memberikan manfaat untuk meningkatkan kualitas data yang akan digunakan dalam analisis selanjutnya atau hasil analisis yang lebih akurat. Selain itu, kita memiliki data yang bersih untuk digunakan dan pengoptimasian data yang lebih mudah di proses untuk memahami pesan teks email oleh model klasifikasi.

BAB II

METODE

2.1 SPACY

Spacy adalah suatu pustaka library python yang terkenal untuk pemrosesan bahasa alami (NLP). Digunakannya ini untuk melakukan suatu pemrosesan dan memahami teks dengan cara pemisahan kalimatnya menjadi bagian yang kata yang lebih kecil hingga ke tahap klasifikasi pada teks.

Spacy sendiri memiliki kecepatan yang baik dan efisien dalam memproses teks, biasanya digunakan dalam jumlah data teks yang besar dan memiliki model Bahasa yang telah dilatih sehingga dalam menganalisis teksnya menjadi lebih mudah.

2.2 WORD2VEC

Word2Vec termasuk juga dalam pemrosesan bahasa alami (NLP), ini merupakan model untuk mempresentasikan setiap kata kedalam bentuk sebagai vector. Word2vec mengimplementasiakan NLP untuk menghitung kesamaan kontekstual dan semantik dari kata.

Dalam memproses teks menjadi lebih mudah dengan memahami teks yang lebih terstruktur sehingga dapat dilakukan analisis lebih lanjut juga akurat dan memiliki pemahaman yang cukup dalam terhadap teks yang diproses.

2.3 TF – IDF

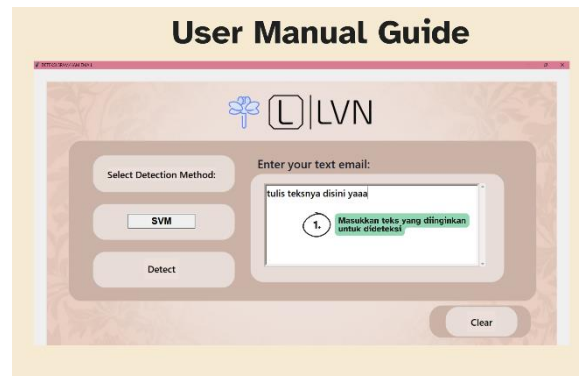
TF – IDF atau Term Frequency – Inverse Document Frequency merupakan gabungan dari dua proses. TF – IDF ini sendiri merupakan ekstraksi fitur untuk pemrosesan teks yang digunakan untuk mengevaluasi kata penting pada teks dengan jumlah yang besar. TF – IDF ini juga dengan mengubah data teks tersebut menjadi vector.

TF (*Term Frequency*) ini digunakan untuk menghitung frekuensi dari banyak jumlah kata yang muncul pada sebuah teks. IDF (*Inverse Document Frequency*) ini merupakan teknik untuk menghitung seberapa penting sebuah kata berdasarkan kemunculan kata pada teks sehingga semakin sering kata itu muncul maka nilai IDF yang rendah, sedangkan kata yang jarang muncul memiliki nilai IDF yang tinggi. Maka dengan gabungan TF – IDF ini membantu dalam menemukan informasi penting dalam pemrosesan atau klasifikasi teks.

BAB III

HASIL DAN ANALISIS

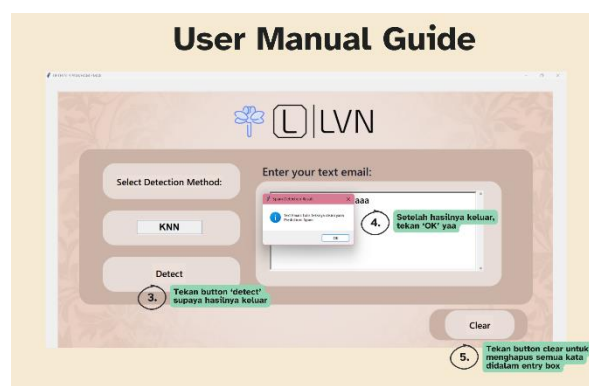
3.1 USER MANUAL GUIDE



- (1) Dapat dilihat pada tahap pertama dengan memasukan teks yang kita inginkan untuk dideteksi apakah teks tersebut merupakan spam atau non-spam



- (2) Pada tahap yang kedua ini dapat memilih jenis algoritma apa yang ingin digunakan, yaitu SVM, KNN, dan Naives Bayes.



- (3) Tahap ketiga ini ditujukan untuk menghapus teks pada button clear dan menampilkan hasil teks tersebut spam atau non-spam dengan button detect untuk mengetahui hasil dari teks yang telah kita ketik. Kemudian akan menampilkan spam detection result yang memberitahukan bahwa teks tersebut spam atau non-spam.

3.2 LISTING CODE

```
import tkinter as tk

from tkinter import scrolledtext, StringVar, OptionMenu, Button, Label, Frame, messagebox

from PIL import Image, ImageTk

import re

import pickle

import csv

import string

from sklearn.feature_extraction.text import TfidfVectorizer

from num2words import num2words

from sklearn.neighbors import KNeighborsClassifier

from gensim.models import Word2Vec # Import Word2Vec

from gensim.models import KeyedVectors # Import Word2Vec

import joblib # Import joblib for loading the pre-trained KNN model


class SpamDetectorApp:

    def __init__(self, master):

        self.window = master

        self.window.state('zoomed')

        self.window.title('DETEKSI SPAM/HAM EMAIL')


        self.register_frame = Frame(self.window, bg="#ede0da", width=1460, height=779)

        self.bgrn_image =
ImageTk.PhotoImage(Image.open('gambar/bgrnn3.png').resize((1460, 779)))

        self.bgrnlabl = Label(self.register_frame, image=self.bgrn_image,
background="#ede0da")

        self.bgrnlabl.place(x=0, y=0)

        self.register_frame.place(x=35, y=35)


        self.label = Label(self.window, text="Enter your text email:", bg="#cbb2a6",
fg="#181b29", font=("yu gothic ui", 25, "bold"), borderwidth=0, relief="solid")

        self.label.place(x=800, y=270, anchor="center")
```

```
self.entry = scrolledtext.ScrolledText(self.window, font=("yu gothic ui", 20, "bold"),
width=40, height=6, wrap='word', bd=5, relief="sunken", insertbackground="#999999",
highlightthickness=2, highlightcolor="#cbb2a6")
```

```
self.entry.place(x=658, y=327)
```

```
self.detect_label = Label(self.window, text="Select Detection Method:", font=("yu
gothic ui", 20, 'bold'), bg="#ede0da", fg="black", borderwidth=0)
```

```
self.detect_label.place(x=365, y=300, anchor="center")
```

```
self.detect_methods = ["SVM", "KNN", "Naive Bayes"]
```

```
self.selected_method_var = StringVar()
```

```
self.selected_method_var.set(self.detect_methods[0])
```

```
self.detect_option_menu = OptionMenu(self.window, self.selected_method_var,
*self.detect_methods)
```

```
self.detect_option_menu.config(indicatoron=0, compound='right', font=("yu gothic ui ",
20,'bold'), width=12, bd=1, relief="solid")
```

```
self.detect_option_menu["menu"].config(font=("yu gothic ui ", 20))
```

```
self.detect_option_menu.place(x=365, y=435, anchor="center")
```

```
self.detect_button = Button(master, text="Detect",
command=self.detect_selected_method, font=("yu gothic ui", 20, 'bold'), bg="#ede0da",
fg="black", borderwidth=0)
```

```
self.detect_button.place(x=365, y=570, anchor="center")
```

```
self.clear_button = Button(master, text="Clear", command=self.clear_entry, font=("yu
gothic ui", 20, 'bold'), bg='#ede0da', fg="black", borderwidth=0)
```

```
self.clear_button.place(x=1280, y=722, anchor="center")
```

```
self.load_models()
```

```
def remove_emoji(self, text):
```

```
    emoji_pattern = re.compile("[
```

```
        u"\U0001F600-\U0001F64F"
```

```

u"\U0001F300-\U0001F5FF"
u"\U0001F680-\U0001F6FF"
u"\U0001F1E0-\U0001F1FF"
u"\U00002500-\U00002BEF"
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"
"]+", flags=re.UNICODE)
return emoji_pattern.sub(r", text)

```

```

def nums_to_words(self, text):
    new_text = []
    for word in text.split():
        if word.isdigit():
            word_in_indonesian = num2words(int(word), lang='id', to='year')
            new_text.append(word_in_indonesian)
        else:
            new_text.append(word)
    return " ".join(new_text)

```



```

def remove_punctuation(self, text):

    no_punct = [char for char in text if char not in string.punctuation]

    words_wo_punct = ".join(no_punct)

    return words_wo_punct


def preprocess_text(self, text):

    text = re.sub("\S*@ \S*\s?", "", text) # Remove emails

    text = re.sub("\S*(http[s]?://|www\.)\S*", "", text) # Remove URL links

    text = re.sub(r"<.*?>", "", text) # Remove HTML tags

    text = re.sub("[^a-zA-Z]", " ", text) # Remove special characters and numbers

    text = re.sub(r"\b\w{1,2}\b", "", text) # Remove too short (2- characters) words

    text = re.sub(r"\b\w{17,}\b", "", text) # Remove too long (17+ characters) words

    text = re.sub(' +', ' ', text).strip() # Remove multiple spaces

    text = text.lower() # Convert to lowercase

    text = self.remove_punctuation(text)

    text = self.nums_to_words(text)

    text = self.remove_emoji(text)

    return text


def load_models(self):

    # Load SVM model and TF-IDF vectorizer from the tuple

    try:

        with open("svm_model.pkl", "rb") as svm_file:

            model_data = pickle.load(svm_file)

            self.svm_model = model_data[0] # Assuming SVM model is at index 0

            self.tfidf_vectorizer = model_data[1] # Assuming TF-IDF vectorizer is at index 1

    except Exception as e:

        messagebox.showerror("Error", f"Error loading SVM model and TF-IDF vectorizer: {str(e)}")

        self.svm_model = None

        self.tfidf_vectorizer = None

```

```

# Load Word2Vec model

try:
    self.knn_model = Word2Vec.load("w2v_model.bin")
except Exception as e:
    messagebox.showerror("Error", f"Error loading Word2Vec model: {str(e)}")
    self.knn_model = None

# Load the Naive Bayes model

try:
    with open("Naive_model.pkl", "rb") as naive_file:
        self.naive_model = pickle.load(naive_file)
except Exception as e:
    messagebox.showerror("Error", f"Error loading Naive Bayes model: {str(e)}")
    self.naive_model = None

# Fungsi untuk Deteksi Spam menggunakan SVM:
def detect_svm(self):
    self.selected_model = "svm"
    preprocessed_text = self.preprocess_text(self.entry.get())
    self.detect_spam(preprocessed_text, self.selected_model)

# Fungsi untuk Deteksi Spam menggunakan KNN:
def detect_knn(self):
    self.selected_model = "knn"
    preprocessed_text = self.preprocess_text(self.entry.get())
    self.detect_spam(preprocessed_text, self.selected_model)

# Fungsi untuk Deteksi Spam menggunakan Naive Bayes:
def detect_naive_bayes(self):
    self.selected_model = "naive_bayes"

```

```

preprocessed_text = self.preprocess_text(self.entry.get())
self.detect_spam(preprocessed_text, self.selected_model)

def detect_svm_logic(self, preprocessed_text):
    print("Preprocessed Text:", preprocessed_text)
    if self.svm_model is None or self.tfidf_vectorizer is None:
        messagebox.showerror("Error", "SVM model or TF-IDF vectorizer not loaded.")
        return

    try:
        text_vectorized = self.tfidf_vectorizer.transform([preprocessed_text])
        decision_function_scores = self.svm_model.decision_function(text_vectorized)
        threshold = 0.0
        predictions = [1 if score > threshold else 0 for score in decision_function_scores]
        prediction = max(set(predictions), key=predictions.count)

        result_message = f"Text Email: {preprocessed_text}\nPrediction: {'Spam' if
prediction == 1 else 'Not Spam'}"
        messagebox.showinfo("Spam Detection Result", result_message)

        self.save_to_csv(preprocessed_text, prediction)

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {str(e)}")

def get_word2vec_vector(self, preprocessed_text):
    words = [word for word in preprocessed_text.split() if
self.knn_model.wv.has_index_for(word)]
    if not words:
        return None # Jika tidak ada kata yang ada dalam vektor Word2Vec, kembalikan
None

```

```

vectors = [self.knn_model.wv.get_vector(word) for word in words]
return sum(vectors) / len(vectors)

def detect_knn_logic(self, preprocessed_text):
    print("Preprocessed Text:", preprocessed_text)
    try:
        text_vector = self.get_word2vec_vector(preprocessed_text)

        if text_vector is None:
            messagebox.showerror("Error", "Word2Vec vector not found for preprocessed
text.")
            return 0

        similar_words = [word[0] for word in
self.knn_model.wv.most_similar(positive=[text_vector], topn=5)]

        # Memastikan kata yang ada dalam vektor Word2Vec sebelum melakukan evaluasi
        similar_words = [word for word in similar_words if
self.knn_model.wv.has_index_for(word)]

        if any(value > 0.5 for word in similar_words for value in self.knn_model.wv[word]):
            return 1 # Spam terdeteksi
        else:
            return 0 # Bukan spam

    except Exception as e:
        # Menampilkan pesan kesalahan jika terjadi kesalahan saat deteksi KNN
        messagebox.showerror("Error", f"An error occurred during KNN detection: {str(e)}")
        return 0

def detect_selected_method(self):

```

```

preprocessed_text = self.preprocess_text(self.entry.get("1.0", "end-1c"))

try:
    if self.selected_method_var.get() == "SVM":
        self.detect_svm_logic(preprocessed_text)
    elif self.selected_method_var.get() == "KNN":
        prediction = self.detect_knn_logic(preprocessed_text) # Fix the parameter passing
        self.handle_prediction_result(preprocessed_text, prediction)
    elif self.selected_method_var.get() == "Naive Bayes":
        self.detect_naive_bayes_logic(preprocessed_text)
    else:
        raise ValueError("Invalid model type")

except Exception as e:
    messagebox.showerror("Error", f"An error occurred: {str(e)}")

def handle_prediction_result(self, preprocessed_text, prediction):
    if prediction == 1:
        result_message = f"Text Email: {preprocessed_text}\nPrediction: Spam"
    else:
        result_message = f"Text Email: {preprocessed_text}\nPrediction: Not Spam"

    messagebox.showinfo("Spam Detection Result", result_message)
    self.save_to_csv(preprocessed_text, prediction)

def detect_naive_bayes_logic(self, preprocessed_text):
    print("Preprocessed Text:", preprocessed_text)
    try:
        prediction = self.naive_model.predict([preprocessed_text])[0]
        result_message = f"Text Email: {preprocessed_text}\nPrediction: {'Spam' if
prediction == 1 else 'Not Spam'}"

```

```

        messagebox.showinfo("Spam Detection Result", result_message)

    self.save_to_csv(preprocessed_text, prediction)

except Exception as e:
    messagebox.showerror("Error", f"An error occurred during Naive Bayes detection: {str(e)}")

def save_to_csv(self, email_text, prediction):
    try:
        with open("detection_result.csv", mode="a", newline="", encoding="utf-8") as file:
            writer = csv.writer(file)
            if file.tell() == 0:
                writer.writerow(["Text Email", "Prediction", "Detection Method"])
            writer.writerow([email_text, "Spam" if prediction == 1 else "Not Spam",
self.selected_method_var.get()])
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred while saving to CSV: {str(e)}")

def clear_entry(self):
    self.entry.delete(1.0, tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    app = SpamDetectorApp(root)
    root.mainloop()

```

3.3 HASIL AKURASI

1. KNN (TF - IDF)

KNN (TF-IDF)	0	1
PRECISION	1.00	0.53
RECALL	0.65	0.99
ACCURACY	0.74	0.74
F1-SCORE	0.78	0.69
SUPPORT	742	293

Hasil dari KNN dengan Ekstraksi fitur menggunakan TF-IDF didapatkan akurasi 74%

2. KNN (WORD2VEC)

KNN (Word2Vec)	0	1
PRECISION	0.97	0.79
RECALL	0.90	0.93
ACCURACY	0.91	0.91
F1-SCORE	0.93	0.85
SUPPORT	735	300

Didapatkan best akurasi KNN Word2vec terbaik yakni K = 1, metric : euclidean dengan akurasi 91%

3. NAÏVE BAYES (TF - IDF)

NAIVE BAYES (TF-IDF)	0	1
PRECISION	0.90	1.00
RECALL	1.00	0.72
ACCURACY	0.92	0.92
F1-SCORE	0.95	0.84
SUPPORT	741	294

Hasil model Naive Bayes dengan Ekstraksi fitur menggunakan TF-IDF. menghasilkan akurasi 92%

4. NAÏVE BAYES (WORD2VEC)

NAIVE BAYES (Word2Vec)	0	1
PRECISION	0.98	0.97
RECALL	0.99	0.94
ACCURACY	0.97	0.97
F1-SCORE	0.98	0.95
SUPPORT	732	303

Didapatkan akurasi Naive Bayes Word2vec terbaik yakni dengan akurasi 97%

5. SVM (TF-IDF)

SVM (TF-IDF)	0	1
PRECISION	0.99	0.96
RECALL	0.98	0.98
ACCURACY	0.98	0.98
F1-SCORE	0.99	0.97
SUPPORT	735	300

Hasil model SVM dengan Ekstraksi fitur menggunakan TF-IDF didapatkan akurasi 98%

6. SVM (WORD2VEC)

SVM (Word2Vec)	0	1
PRECISION	0.97	0.73
RECALL	0.86	0.93
ACCURACY	0.88	0.88
F1-SCORE	0.91	0.82
SUPPORT	735	300

Didapatkan best akurasi SVM Word2vec yakni dengan akurasi 88%

BAB IV

KESIMPULAN

Dalam laporan project ini dengan melakukan klasifikasi spam pada email. Dengan beberapa tahap dalam pengklasifikasi yang dilakukan, yaitu observing data, cleaning data, kemudian melakukan preparasi data dan ekstraksi fitur seperti Word2Vec dan TF – IDF, melakukan evaluasi juga melihat confusion matrix nya hingga kepada klasifikasi reportnya.


Berdasarkan hasil keseluruhan yang telah kami lakukan. Pada klasifikasi data spam dan ham kami menggunakan 3 algoritma diantaranya SVM, KNN, Naive Bayes dengan ekstraksi fitur menggunakan TFIDF dan Word2Vec. Kami mengkomparasi model ketiga algoritma dengan ekstraksi fitur yang berbeda, sehingga menunjukkan performa yang terbaik dari ketiga model algoritma. Dari ketiga algoritma didapatkan hasil terbaik dari SVM dengan ekstraksi fitur TFIDF yang menghasilkan akurasi sebesar 98%

DAFTAR PUSTAKA

- Zamil, Y. K., Ali, S. A., & Naser, M. A. (2019). Spam image email filtering using K-NN and SVM. *International Journal of Electrical and Computer Engineering (IJECE)*, 9(1), 245. <https://doi.org/10.11591/ijece.v9i1.pp245-254>
- Srinivasulu, N., & Sabitha, R. (n.d.). Comparison of Support Vector Machine and K-Nearest Neighbor in Detecting Spam Sms for Improved Accuracy Section A-Research paper COMPARISON OF SUPPORT VECTOR MACHINE AND K-NEAREST NEIGHBOR IN DETECTING SPAM SMS FOR IMPROVED ACCURACY. *Chem. Bull*, 2023(S1), 4327–4334. <https://www.eurchembull.com/uploads/paper/7e38579f083ea0352ecabbd013d170d8.pdf>
- Dai, Y., Tada, S., Ban, T., Nakazato, J., Shimamura, J., & Ozawa, S. (2014). Detecting Malicious Spam Mails: An Online Machine Learning Approach. *Neural Information Processing*, 365–372. https://doi.org/10.1007/978-3-319-12643-2_45
- Omotehinwa, T. O., & Oyewola, D. O. (2023). Hyperparameter Optimization of Ensemble Models for Spam Email Detection. *Applied Sciences*, 13(3), 1971. <https://doi.org/10.3390/app13031971>
- AbdulNabi, I., & Yaseen, Q. (2021). Spam Email Detection Using Deep Learning Techniques. *Procedia Computer Science*, 184, 853–858. <https://doi.org/10.1016/j.procs.2021.03.107>
- Harisinghaney, A., Dixit, A., Gupta, S., & Arora, A. (2014). Text and image based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm. *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. <https://doi.org/10.1109/icroit.2014.6798302>
- RAZA, M., Jayasinghe, N. D., & Muslam, M. M. A. (2021). A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms. *2021 International Conference on Information Networking (ICOIN)*. <https://doi.org/10.1109/icoin50884.2021.9334020>
- Ma, T. M., YAMAMORI, K., & Thida, A. (2020, October 1). A Comparative Approach to Naïve Bayes Classifier and Support Vector Machine for Email Spam Classification. *IEEE Xplore*. <https://doi.org/10.1109/GCCE50665.2020.9291921>
- Nagaraj, P., Muneeswaran, V., Shyam Sundar Reddy, G., Bharath Kumar, V., Madhan Mohan, B., & Kumar, S. (2023, January 1). Automatic Email Spam Classification Using Naïve Bayes. *IEEE Xplore*. <https://doi.org/10.1109/ICCCI56745.2023.10128233>
- Laksono, E., Basuki, A., & Bachtiar, F. (2020). Optimization of K Value in KNN Algorithm for Spam and Ham Email Classification. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 4(2), 377–383. <https://doi.org/10.29207/resti.v4i2.1845>
- Anuradha Reddy, Dr M Umamaheswari, Dr A Viswanathan, G Vikram, & Mamatha K. (2022). Using Support Vector Machine for Classification and Feature Extraction of Spam in Email. *International Journal of Advanced Research in Science, Communication and Technology*, 85–89. <https://doi.org/10.48175/ijarsct-5904>
- Singh, M., Pamula, R., & shekhar, S. kumar. (2018, September 1). Email Spam Classification by Support Vector Machine. *IEEE Xplore*. <https://doi.org/10.1109/GUCON.2018.8674973>
- Bassiouni, M., Ali, M., & El-Dahshan, E. A. (2018). Ham and Spam E-Mails Classification Using Machine Learning Techniques. *Journal of Applied Security Research*, 13(3), 315–331. <https://doi.org/10.1080/19361610.2018.1463136>

LAMPIRAN

DETEKSI SPAM/HAM EMAIL



Select Detection Method:

SVM


Detect

Enter your text email:

berikan "teks" disini untuk mendeteksi apakah termasuk "spam" atau "non-spam"

Clear

DETEKSI SPAM/HAM EMAIL



Select Detection Method:

SVM

SVM

KNN

Naive Bayes


Detect

Enter your text email:

berikan "teks" disini untuk mendeteksi apakah termasuk "spam" atau "non-spam"

Clear

Spam Detection Result



Text Email: berikan teks disini untuk mendeteksi apakah termasuk spam atau non spam

Prediction: Spam

OK