Prediksi Dini Risiko Bunuh Diri melalui Data Teks

MAKALAH LAPORAN PROJECT

DISUSUN OLEH : KELOMPOK 6

NAHDLIYAH ZAHRAH	**22031554024**
RESHAR FALDI.J	**22031554025**
NUR HALIZAH AMRITA	**22031554039**

S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS NEGERI SURABAYA

2023

BAB I

PENDAHULUAN

1.1 Latar Belakang

Masalah bunuh diri adalah isu serius dalam kesehatan mental yang memerlukan perhatian khusus. Peningkatan angka bunuh diri dan dampaknya terhadap masyarakat menunjukkan perlunya pendekatan yang holistik dan inovatif dalam pencegahan dan intervensi. Dalam beberapa tahun terakhir, perkembangan teknologi dan platform daring telah membuka pintu bagi akses terhadap berbagai data teks yang mencakup ungkapan perasaan dan pikiran individu. Analisis data teks menjadi salah satu metode yang menjanjikan dalam mendeteksi potensi risiko bunuh diri, karena memberikan wawasan mendalam terkait keadaan mental seseorang.

Menangani masalah ini dapat membantu mencegah tragedi yang menghancurkan kehidupan individu dan keluarga. Salah satu pendekatan untuk mendeteksi potensi risiko bunuh diri adalah melalui analisis teks, khususnya dengan menggunakan metode Linear Support Vector Machine (SVM) dan model Bag-of-Words (BoW). Dengan merancang dan mengimplementasikan model menggunakan metode ini, diharapkan dapat dikembangkan alat yang efektif dalam mendukung upaya pencegahan bunuh diri dengan menganalisis potensi risiko melalui data teks.

1.2 Rumusan Masalah

1. Bagaimana menerapkan metode Linear Support Vector Machine (SVM) dan model Bag-of-Words (BoW) dalam menganalisis data teks untuk memprediksi potensi risiko bunuh diri?
2. Metode ekstraksi fitur apakah yang menghasilkan akurasi paling baik untuk proses klasifikasi?
3. Sejauh mana keefektifan metode SVM dan BoW dalam mengklasifikasikan data teks terkait risiko bunuh diri dibandingkan dengan metode klasifikasi lainnya?

1.3 Tujuan dan Manfaat

Proyek ini bertujuan untuk menyusun model prediktif menggunakan metode Linear Support Vector Machine (SVM) dan model Bag-of-Words (BoW) untuk menganalisis data teks terkait bunuh diri. Langkah-langkah utama melibatkan membersihkan data dari noise atau informasi tidak relevan dan mengoptimalkan data untuk meningkatkan kualitas analisis. Selanjutnya, proyek ini akan fokus pada pengembangan model klasifikasi yang dapat mengidentifikasi pola kata-kata yang mengindikasikan potensi risiko bunuh diri.

Dalam hal ini dapat memberikan manfaat kontribusi pada pemahaman dan deteksi dini potensi risiko bunuh diri melalui analisis teks. Dengan memiliki model yang bersih dan dioptimalkan, kita dapat meningkatkan akurasi prediksi risiko bunuh diri, yang pada gilirannya dapat mendukung upaya pencegahan bunuh diri dan memberikan wawasan yang lebih baik bagi praktisi kesehatan mental. Selain itu, model ini mempermudah integrasi hasil analisis dalam strategi pencegahan bunuh diri, memberikan dampak positif dalam meningkatkan kesejahteraan mental masyarakat.

BAB II

METODE

2.1 Pengumpulan data

Dalam projek ini, data yang digunakan membangun model deteksi bunuh diri diperolehk dari kaggle dengan jumlah baris sebanyak 1778 dan 2 kolom. Data ini berisi cuitan tweet yang terdapat pada kolom pertama dan kolom kedua merupakan label cuitan yang terdiri dari potential suicide post dan not potential suicide.
Link data: https://www.kaggle.com/datasets/aunanya875/suicidal-tweet-detection-dataset

2.2 Data cleansing

Data cleansing ini merupakan langkah mengelola data-data yang tidak lengkap, mengandung eror, dan tidak konsisten dibuang dari koleksi data, sehingga data yang telah bersih relevan untuk diproses (Zai,C. (2022)).

2.3 Preprocessing data

Preprocessing data merupakan langkah mempersiapkan data mentah dengan tujuan mengubah data yang tidak terstruktur menjadi terstruktur. Proses yang dilakukan dalam preprocessing sebagai berikut (Isnain dkk, 2021):

a. Lower casing, merupakan menyeragamkan semua huruf menjadi huruf kecil.
b. Merubah singkatan atau slang yang ada pada kamus.
c. Punctuation, merupakan menghilangkan tanda baca dari dokumen atau data.
d. Cleansing, yaitu membersihkan data atau dokumen dari kata yang tidak diperlukan seperti html, emoticon, hashtag, non-alphanumeric, mention, dan url.
e. Stopword, yaitu menghilangkan kata yang kurang efektif.
f. Contraction, merupakan memperluas kontraksi dalam teks.
g. Steaming, yaitu proses untuk menyaring kata yang terdapat kata sambung, kata ganti, kata depan, menjadi kata dasar dengan menghilangkan awalan atau akhiran.

2.4 Modelling

A. Bag-of-Words (BoW)

Bag of Words (BoW) adalah model yang serbaguna dan dapat digunakan sebagai algoritma pemilihan fitur dan klasifikasi dokumen serta gambar. Metode BoW merekam jumlah kemunculan setiap kata yang dibuat untuk setiap jenis instansi atau kata, tanpa memperhatikan urutan kata atau tata bahasa (Qader, Ameen, and Ahmed 2019).

B. Linear Support Vector Machine (SVM)

Linear SVM adalah metode klasifikasi Support Vector Machine (SVM) yang menggunakan hyperplane linier untuk memisahkan data (Jakkula 2011). Dalam konteks ini, SVM linier digunakan untuk memisahkan data menggunakan garis atau hyperplane linier. Menurut Tang (2013), linear SVM digunakan untuk masalah klasifikasi biner, di mana tujuannya adalah untuk menemukan hyperplane yang memisahkan dua kelas data dengan

margin maksimum. Dalam konteks deep learning, linear SVM digunakan sebagai pengganti lapisan softmax dalam model deep learning untuk meningkatkan kinerja dan efek regularisasi.

BAB III

HASIL DAN ANALISIS

3.1 Preprocessing data

Pada projek ini sebelum diolah, tahap pertama yang akan dilakukan adalah preprocessing meliputi: lowercase, punctuation, menghapus URL, menghapus angka, mengubah kata singkatan atau slang kebentuk aslinya, mengubah bentuk html ke bentuk asli, menghapus non-alphanumeric, stopword dan stemming menggunakan lemmatization.

3.2 Ekstraksi fitur

Setelah dilakukan preprocessing data maka data sudah siap untuk melakukan tahap ekstraksi fitur. Projek ini menggunakan bebrapa ekstraksi fitur Tf-Idf, BoW, dan Word2vec sebagai perbandingan. Sebelum ekstraksi fitur data di split dengan perbandingan data latih 80% dan data uji 20%.

3.3 Implementasi Algoritma

Algoritma yang digunakan pada projek ini didapat dari hasil K-fold atau validasi silang untuk mengukur kinerja model dengan membagi dataset menjadi beberapa subset atau lipatan (folds). Hasil K-fold terdapat pada gambar 1.

HASIL Kfold MENCARI MODEL TERBAIK														
Model	Decision Tree		Random Forest		SVM		Gaussian Naïve Bayes		Logistic Regression		KNN		Linear SVM	
Fitur	Akurasi	SD	Akurasi	SD	Akurasi	SD	Akurasi	SD	Akurasi	SD	Akurasi	SD	Akurasi	SD
Tf-Idf	0.87964	-0.00933	0.90063	-0.0107	0.87476	-0.01658	0.61931	-0.029721	0.85375	-0.01286	0.678783	-0.0196	0.89924	-0.01133
BOW	0.88313	-0.01713	0.90553	-0.0152	0.89712	-0.02476	0.61372	-0.035411	0.90204	-0.02026	0.787965	-0.02913	0.91183	-0.01224
Word2vec	0.704	-0.03076	0.82716	-0.0137	0.83696	-0.02	0.67673	-0.043287	0.63679	-0.01851	0.657812	-0.03255	0.62769	-0.02146

Gambar 1. Hasil K-fold perbandingan 7 model dengan 3 ekstraksi fitur

Dalam hasil K-Fold tersebut kemudian terdapat tiga model dengan akurasi terbaik dengan nilai akurasi sekitar 0.9 yang terdiri dari algorithma Random fores Logistic Regresion, dan LinearSVM. Hasil terbaik tersebut kemudian dihitung ulang menggunakan algoritma model secara langsung. Hasil pada perhitungan model terbaik didapat bahwa model terbaiknya adalah LinearSVM dengan ekstraksi fitur BoW sehingga model dan ekstraksi fitur itulah yang akan digunakan untuk membangun model deteksi bunuh diri, pada gambar 2.

Model Terbaik												
Model	Random Forest				Logistic Regression				Linear SVM			
Fitur	Akurasi	precision	recall	f1-score	Akurasi	precision	recall	f1-score	Akurasi	precision	recall	f1-score
Tf-Idf	0.90223	0.91	0.88	0.89	0.86313	0.9	0.83	0.85	0.89665	0.91	0.88	0.89
BOW	0.88547	0.92	0.87	0.89	0.88827	0.91	0.86	0.88	0.90503	0.93	0.88	0.9
Word2vec	0.82961	0.83	0.78	0.79								

Gambar 2. Hasil model terbaik.

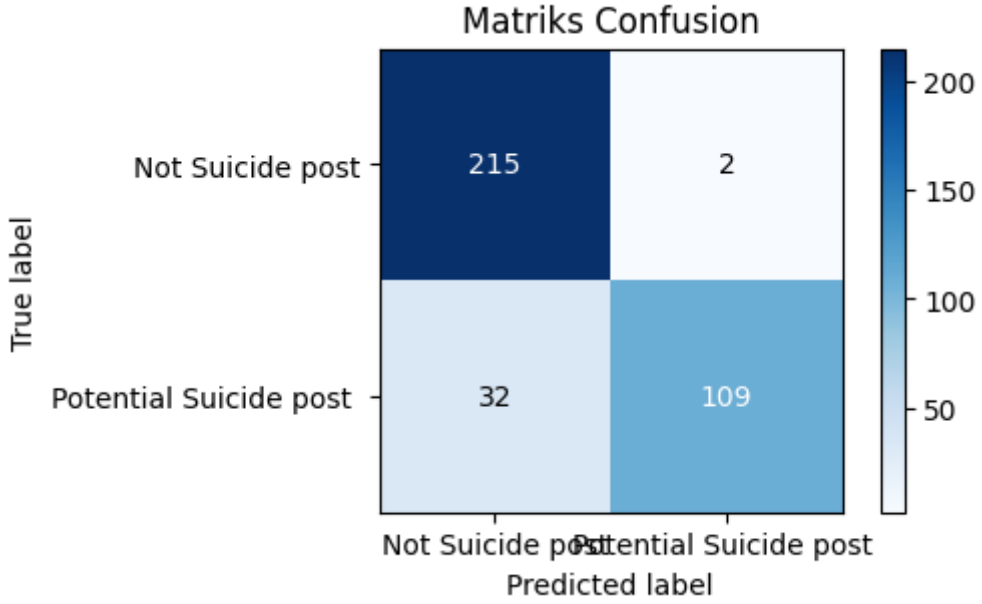
Hasil modelling LinearSVM di dapatkan nilai akurasi, precision, recall, f1-Score yang terdapat pada gambar 3 dan Confusion Matriks pada gambar 4.

```
Akurasi: 0.9050279329608939
Laporan Klasifikasi:
                        precision    recall  f1-score   support

       Not Suicide post       0.87      0.99      0.93       217
Potential Suicide post        0.98      0.77      0.87       141

               accuracy                           0.91       358
              macro avg       0.93      0.88      0.90       358
           weighted avg       0.91      0.91      0.90       358
```

Gambar 3. Hasil akurasi model LinearSVM dengan BoW,



Gambar 4. Confusion mariks model LinearSVM dengan BoW.

KESIMPULAN

Dalam laporan project deteksi potensi bunuh diri ini, prosses dimulai dengan observasi data, membersihkan data dan preprocessing data. Langkah selanjutnya melibatkan ekstraksi fitur menggunakan metode Bag-of-Words (BOW), TF-IDF, dan Word2Vec. Model yang dievaluasi melibatkan Decision Tree, Random Forest, SVM, Naive Bayes, Logistic Regression, KNN, dan Linear SVM. Setelah membandingkan performa, ditemukan bahwa algoritma Linear SVM dengan metode BOW memberikan hasil terbaik, mencapai akurasi sebesar 0.90503."

DAFTAR PUSTAKA

Jakkula, Vikramaditya. 2011. "Tutorial on Support Vector Machine (SVM)." *School of EECS, Washington State University*, 1–13. http://www.ccs.neu.edu/course/cs5100f11/resources/jakkula.pdf.

Tang, Yichuan. 2013. "Deep Learning Using Linear Support Vector Machines." http://arxiv.org/abs/1306.0239.

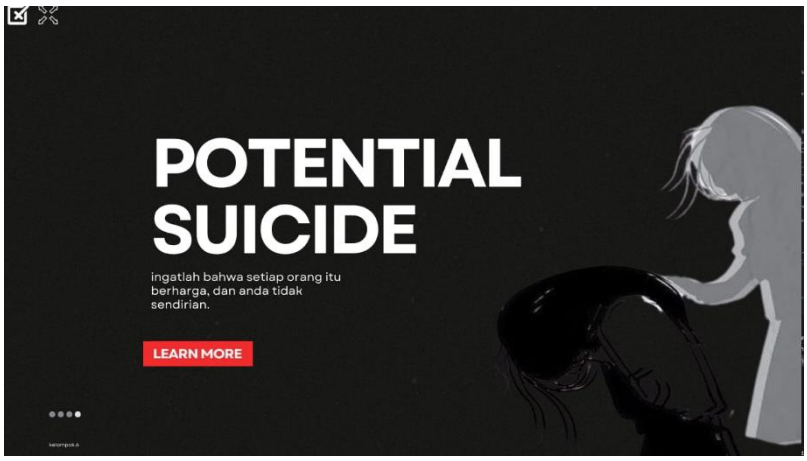
Qader, Wisam A., Musa M. Ameen, and Bilal I. Ahmed. 2019. "An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges." *Proceedings of the 5th International Engineering Conference, IEC 2019*, no. June: 200–204. https://doi.org/10.1109/IEC47844.2019.8950616.

Zai, C. (2022). Implementasi Data Mining Sebagai Pengolahan Data. *Portaldata.org,* 2(3).

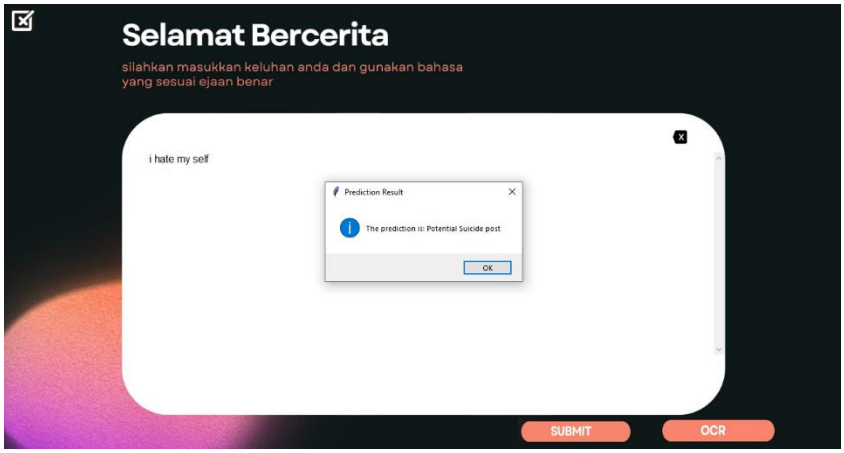
Isnain, A. R., Sakti, A. I., Alita, D., & Marga, N. S. (2021). Sentimen Analisis Publik Terhadap Kebijakan Lockdown Pemerintah Jakarta Menggunakan Algoritma SVM. JDMSI (Journal of Data Mining and Statistical Informatics), 2(1), 31-37. ISSN: 2745-8458.

LAMPIRAN

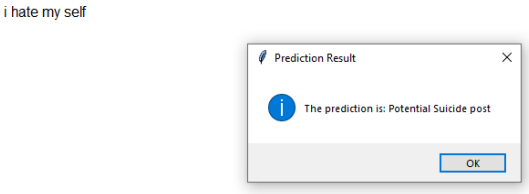
1. Screenshot GUI



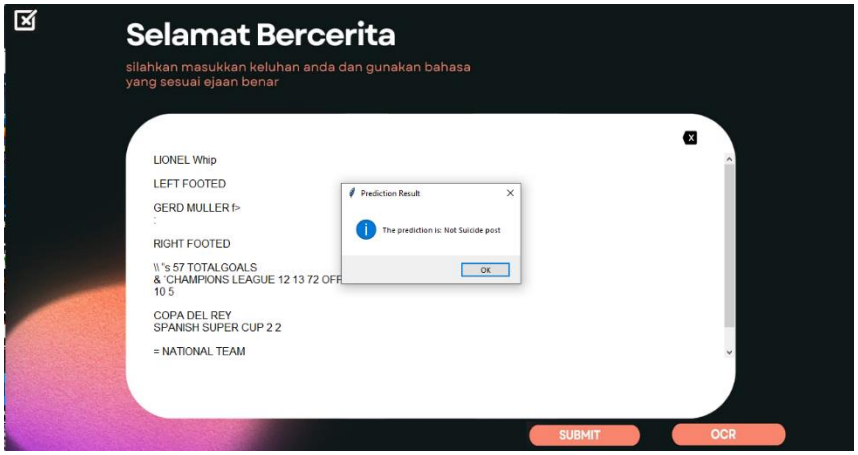
Gambar 5. Tampilan awal



Gambar 6. Tampilan kedua input kata manual



Gambar 7. Hasil prediksi



Gambar 8. Pengguna memasukkan kata melalui tombol "UCR"

2. Listing Code

```
from tkinter import *
from tkinter import Tk, Text, Button, Label, filedialog
from tkinter import messagebox
from PIL import Image, ImageTk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from    sklearn.metrics    import    accuracy_score,    classification_report,
confusion_matrix
import re
import string
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from PIL import Image
import pytesseract as tess
tess.pytesseract.tesseract_cmd=r'C:\Program Files\Tesseract-OCR\tesseract.exe'


class IntroLogin:
    def __init__(self, master):
        self.master = master
        master.geometry('1280x800+50+40')
        master.resizable(False, False)
        master.overrideredirect(True)
        master.config(bg='#3d3d3d')
        self.loginPage_pad = None
        self.loginInterface()

    def loginInterface(self):
        loginPage = Image.open('Login.jpeg')
```

```
        loginpage_resize = loginPage.resize((1280, 800))
        loginPage = ImageTk.PhotoImage(loginpage_resize)
        self.loginPage_pad = Label(image=loginPage, text='haha')
        self.loginPage_pad.image = loginPage
        self.loginPage_pad.pack()

        learnmoreButton = Image.open('LearnMore.jpeg')
        learnmoreButton_resize = learnmoreButton.resize((175, 40))
        learnmoreButton = ImageTk.PhotoImage(learnmoreButton_resize)
        learnmoreButton_pad  =  Button(image=learnmoreButton,  borderwidth=0,
highlightthickness=0,
                                       bd=0, command=self.tampilkanHalamanKedua,
activebackground='grey',
                                       cursor='hand2')
        learnmoreButton_pad.image = learnmoreButton
        learnmoreButton_pad.place(x=230, y=550)

        moveBar = Image.open('movebar.jpeg')
        moveBar_resize = moveBar.resize((40, 40))
        moveBar = ImageTk.PhotoImage(moveBar_resize)
        moveBar_pad = Label(image=moveBar, bd=0)
        moveBar_pad.image = moveBar
        moveBar_pad.place(x=58, y=10)
        moveBar_pad.bind('<B1-Motion>', self.moveApp)

        closeLogo = Image.open('close.jpeg')
        closeLogo_resize = closeLogo.resize((40, 40))
        closeLogo = ImageTk.PhotoImage(closeLogo_resize)
        closePad = Button(image=closeLogo, borderwidth=0, highlightthickness=0,
bd=0,
                          command=self.closeProgram, activebackground='grey')
        closePad.image = closeLogo
        closePad.place(x=10, y=10)

    def moveApp(self, e):
        root.geometry(f'+{(e.x_root)-58}+{(e.y_root)-10}')

    def closeProgram(self):
        root.quit()

    def tampilkanHalamanKedua(self):
        self.master.destroy()
        HalamanKedua()

class HalamanKedua:
    def __init__(self):
        self.root = Tk()
        self.root.geometry('1280x800+50+40')
```

```
        self.root.resizable(False, False)
        self.root.overrideredirect(True)
        self.root.config(bg='#3d3d3d')
        self.tampilkanHalamanKedua()

    def closeProgram(self):
        root.quit()

    def tampilkanHalamanKedua(self):
        gambarHalamanKedua = Image.open('bag2fix.jpeg')
        gambarHalamanKedua_resize = gambarHalamanKedua.resize((1280, 800))
        gambarHalamanKedua = ImageTk.PhotoImage(gambarHalamanKedua_resize)
        padGambarHalamanKedua = Label(self.root, image=gambarHalamanKedua,
compound='center',
                                      font=('Helvetica', 20), fg='white',
bg='#3d3d3d')
        padGambarHalamanKedua.image = gambarHalamanKedua
        padGambarHalamanKedua.pack()

        closeLogo = Image.open('close.jpeg')
        closeLogo_resize = closeLogo.resize((40, 40))
        closeLogo = ImageTk.PhotoImage(closeLogo_resize)
        closePad = Button(image=closeLogo, borderwidth=0, highlightthickness=0,
bd=0,
                          command=self.closeProgram, activebackground='grey')
        closePad.image = closeLogo
        closePad.place(x=10, y=10)

        frame_isian = Frame(self.root, bd=0, width=700, height=280, bg='white')
        frame_isian.place(x=220, y=230)

        self.text_input = Text(frame_isian, font=('arial', 12), width=95,
height=17, bg='white', fg='black', bd=0)
        self.text_input.pack(side=LEFT, fill=BOTH, padx=2)
        scrollbar = Scrollbar(frame_isian)
        scrollbar.pack(side=RIGHT, fill=BOTH)

        self.text_input.config(yscrollcommand=scrollbar.set)
        scrollbar.config(command=self.text_input.yview)
        submitButton = Image.open('submit.jpeg')
        submitButton_resize = submitButton.resize((175, 40))
        submitButton = ImageTk.PhotoImage(submitButton_resize)
        submitButton_pad = Button( image=submitButton, borderwidth=0,
highlightthickness=0,
                                   bd=0,         command=self.predict_input,
activebackground='black',
                                   cursor='hand2')
        submitButton_pad.image = submitButton
```

```
        submitButton_pad.place(x=784, y=634)
        # Create a button to delete the content of the Text widget
        # Create a button to delete the content of the Text widget
        deleteButtonImage = Image.open('delete.jpg')
        deleteButtonImage_resize = deleteButtonImage.resize((22, 22))
        deleteButtonImage = ImageTk.PhotoImage(deleteButtonImage_resize)
   deleteButton_pad    =    Button(image=deleteButtonImage,    borderwidth=0,
highlightthickness=0,
                                   bd=0,         command=self.delete_content,
activebackground='black',
                                   cursor='hand2')
        deleteButton_pad.image = deleteButtonImage
        deleteButton_pad.place(x=1019, y=196)

        ocrButton = Image.open('OCR.jpg')
        ocrButton_resize = ocrButton.resize((175, 40))
        ocrButton = ImageTk.PhotoImage(ocrButton_resize)
        ocrButton_pad     =     Button(     image=ocrButton,     borderwidth=0,
highlightthickness=0,
                                    bd=0,             command=self.openimage,
activebackground='black',
                                    cursor='hand2')
        ocrButton_pad.image = ocrButton
        ocrButton_pad.place(x=1000, y=634)
        closeLogo = Image.open('close.jpeg')
        closeLogo_resize = closeLogo.resize((40, 40))
        closeLogo = ImageTk.PhotoImage(closeLogo_resize)
        closePad = Button(image=closeLogo, borderwidth=0, highlightthickness=0,
bd=0,
                          command=self.closeProgram, activebackground='grey')
        closePad.image = closeLogo
        closePad.place(x=10, y=10)

    def openimage(self):
        filename = filedialog.askopenfilename()
        img1 = Image.open(filename)
        get_txt = tess.image_to_string(img1, lang='eng')
        print(get_txt)
        self.text_input.insert('0.0',get_txt)

    def delete_content(self):
        # Clear the content of the Text widget
        self.text_input.delete("1.0", END)

    def predict_input(self):
        # Get the input from the Text widget
        input_text = self.text_input.get("1.0", END)
```

```
        # Preprocess the input
        preprocessed_text = preprocess(input_text)

        # Count Vectorization
        input_vectorized = count_vectorizer.transform([preprocessed_text])
        # Predict
        prediction = modelLSBOW.predict(input_vectorized)
        messagebox.showinfo("Prediction   Result",   f"The   prediction   is:
{prediction[0]}")

import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import classification_report,confusion_matrix
import pandas as pd
import numpy as np
import os
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import seaborn as sns

df = pd.read_csv('ds.csv')
df = df.loc[df['Tweet'] != '[]']
count_columns_with_empty_string = (df == '[]').sum()

id_stopword_dict = pd.read_csv('ST.csv', header=None, encoding='latin-1')
id_stopword_dict = id_stopword_dict.rename(columns={0: 'stopword'})
abbreviations = {
    "$" : " dollar ",
    "€" : " euro ",
    "4ao" : "for adults only",
    "a.m" : "before midday",
    "a3" : "anytime anywhere anyplace",
    "aamof" : "as a matter of fact",
    "acct" : "account",
    "adih" : "another day in hell",
    "afaic" : "as far as i am concerned",
    "afaict" : "as far as i can tell",
    "afaik" : "as far as i know",
    "afair" : "as far as i remember",
    "afk" : "away from keyboard",
    "app" : "application",
    "approx" : "approximately",
    "apps" : "applications",
    "asap" : "as soon as possible",
    "asl" : "age, sex, location",
```

```
"atk" : "at the keyboard",
"ave." : "avenue",
"aymm" : "are you my mother",
"ayor" : "at your own risk",
"b&b" : "bed and breakfast",
"b+b" : "bed and breakfast",
"b.c" : "before christ",
"b2b" : "business to business",
"b2c" : "business to customer",
"b4" : "before",
"b4n" : "bye for now",
"b@u" : "back at you",
"bae" : "before anyone else",
"bak" : "back at keyboard",
"bbbg" : "bye bye be good",
"bbc" : "british broadcasting corporation",
"bbias" : "be back in a second",
"bbl" : "be back later",
"bbs" : "be back soon",
"be4" : "before",
"bfn" : "bye for now",
"blvd" : "boulevard",
"bout" : "about",
"brb" : "be right back",
"bros" : "brothers",
"brt" : "be right there",
"bsaaw" : "big smile and a wink",
"btw" : "by the way",
"bwl" : "bursting with laughter",
"c/o" : "care of",
"cet" : "central european time",
"cf" : "compare",
"cia" : "central intelligence agency",
"csl" : "can not stop laughing",
"cu" : "see you",
"cul8r" : "see you later",
"cv" : "curriculum vitae",
"cwot" : "complete waste of time",
"cya" : "see you",
"cyt" : "see you tomorrow",
"coz" : "because",
"dae" : "does anyone else",
"dbmib" : "do not bother me i am busy",
"diy" : "do it yourself",
"dm" : "direct message",
"dwh" : "during work hours",
"didn" : "didn't",
"e123" : "easy as one two three",
```

```
"eet" : "eastern european time",
"eg" : "example",
"embm" : "early morning business meeting",
"encl" : "enclosed",
"encl." : "enclosed",
"etc" : "and so on",
"eng":"english",
"faq" : "frequently asked questions",
"fawc" : "for anyone who cares",
"fb" : "facebook",
"fc" : "fingers crossed",
"fig" : "figure",
"fimh" : "forever in my heart",
"ft." : "feet",
"ft" : "featuring",
"ftl" : "for the loss",
"ftw" : "for the win",
"fwiw" : "for what it is worth",
"fyi" : "for your information",
"ftw" : "for the win",
"g9" : "genius",
"gahoy" : "get a hold of yourself",
"gal" : "get a life",
"gcse" : "general certificate of secondary education",
"gfn" : "gone for now",
"gg" : "good game",
"gl" : "good luck",
"glhf" : "good luck have fun",
"gmt" : "greenwich mean time",
"gmta" : "great minds think alike",
"gn" : "good night",
"g.o.a.t" : "greatest of all time",
"goat" : "greatest of all time",
"goi" : "get over it",
"gps" : "global positioning system",
"gr8" : "great",
"gratz" : "congratulations",
"gyal" : "girl",
"ger" : "German",
"ger/rus/eng" : "german,rusian,english",
"h&c" : "hot and cold",
"hp" : "horsepower",
"hr" : "hour",
"hrh" : "his royal highness",
"ht" : "height",
"hwo're":"how are",
"hav" : "have",
"hoe" : "whore",
```

```
"herme" : "hear me",
"ibrb" : "i will be right back",
"ic" : "i see",
"icq" : "i seek you",
"icymi" : "in case you missed it",
"idc" : "i do not care",
"idgadf" : "i do not give a damn fuck",
"idgaf" : "i do not give a fuck",
"idk" : "i do not know",
"ie" : "that is",
"i.e" : "that is",
"ifyp" : "i feel your pain",
"IG" : "instagram",
"iirc" : "if i remember correctly",
"ilu" : "i love you",
"ily" : "i love you",
"ill" : "i'll",
"imho" : "in my humble opinion",
"imo" : "in my opinion",
"imu" : "i miss you",
"iow" : "in other words",
"irl" : "in real life",
"j4f" : "just for fun",
"jic" : "just in case",
"jk" : "just kidding",
"jsyk" : "just so you know",
"jst": "just",
"l8r" : "later",
"lb" : "pound",
"lbs" : "pounds",
"ldr" : "long distance relationship",
"lmao" : "laugh my ass off",
"lmfao" : "laugh my fucking ass off",
"lol" : "laughing out loud",
"ltd" : "limited",
"ltns" : "long time no see",
"m8" : "mate",
"mf" : "motherfucker",
"mfs" : "motherfuckers",
"mfw" : "my face when",
"mofo" : "motherfucker",
"mph" : "miles per hour",
"mr" : "mister",
"mrw" : "my reaction when",
"ms" : "miss",
"mte" : "my thoughts exactly",
"nagi" : "not a good idea",
"nbc" : "national broadcasting company",
```

```
"nbd" : "not big deal",
"nfs" : "not for sale",
"ngl" : "not going to lie",
"nhs" : "national health service",
"nrn" : "no reply necessary",
"nsfl" : "not safe for life",
"nsfw" : "not safe for work",
"nth" : "nice to have",
"nvr" : "never",
"nyc" : "new york city",
"oc" : "original content",
"og" : "original",
"ohp" : "overhead projector",
"oic" : "oh i see",
"omdb" : "over my dead body",
"omg" : "oh my god",
"omw" : "on my way",
"p.a" : "per annum",
"p.m" : "after midday",
"pm" : "prime minister",
"poc" : "people of color",
"pov" : "point of view",
"pp" : "pages",
"ppl" : "people",
"prw" : "parents are watching",
"ps" : "postscript",
"pt" : "point",
"ptb" : "please text back",
"pto" : "please turn over",
'pls' : "please",
"plz" : "please",
"qpsa" : "what happens", #"que pasa",
"ratchet" : "rude",
"rbtl" : "read between the lines",
"rlrt" : "real life retweet",
"rofl" : "rolling on the floor laughing",
"roflol" : "rolling on the floor laughing out loud",
"rotflmao" : "rolling on the floor laughing my ass off",
"rt" : "retweet",
"ruok" : "are you ok",
"rus" :"rusian",
"re" : "are",
"sfw" : "safe for work",
"sk8" : "skate",
"smh" : "shake my head",
"sq" : "square",
"srsly" : "seriously",
"ssdd" : "same stuff different day",
```

```
    "tbh" : "to be honest",
    "tbs" : "tablespooful",
    "tbsp" : "tablespooful",
    "tfw" : "that feeling when",
    "tf" : "the fuck",
    "thks" : "thank you",
    "tho" : "though",
    "thx" : "thank you",
    "tia" : "thanks in advance",
    "til" : "today i learned",
    "tl;dr" : "too long i did not read",
    "tldr" : "too long i did not read",
    "tmb" : "tweet me back",
    "tntl" : "trying not to laugh",
    "ttyl" : "talk to you later",
    "u" : "you",
    "u2" : "you too",
    "u4e" : "yours for ever",
    "utc" : "coordinated universal time",
    "w/" : "with",
    "w/o" : "without",
    "w8" : "wait",
    "wot" : "what",
    "wassup" : "what is up",
    "wb" : "welcome back",
    "wtf" : "what the fuck",
    "wtg" : "way to go",
    "wtpa" : "where the party at",
    "wuf" : "where are you from",
    "wuzup" : "what is up",
    "wywh" : "wish you were here",
    "wit" : "with",
    "yd" : "yard",
    "ygtr" : "you got that right",
    "ynk" : "you never know",
    "youve" : "you've",
    "yr" : "your",
    "ve" : "have",
    "zzz" : "sleeping bored and tired"
}

def convert_abbrev(word):
    if ',' in word:
        # Jika ada koma, pisahkan kata dan cek kamus
        words = word.split(',')
        return ', '.join(abbreviations.get(w.lower(), w) for w in words)
    else:
        # Menggunakan kamus untuk mengganti kata
```

```
        return abbreviations.get(word.lower(), word)

# Mengaplikasikan fungsi pada DataFrame
df['preprocess']      =      df['Tweet'].astype(str).apply(lambda      x:      '
'.join(convert_abbrev(word) for word in x.split()))

#========================================================================
import contractions
import pandas as pd
import html
# Fungsi untuk memperluas kontraksi
def expand_contractions(text):
    text = html.unescape(text)
    return contractions.fix(text)
df['preprocess'] = df['preprocess'].apply(expand_contractions)

import contractions
import pandas as pd

# Fungsi untuk memperluas kontraksi
def expand2_contractions(text):
    return contractions.fix(text)
df['preprocess'] = df['preprocess'].apply(expand2_contractions)
# df

#========================================================================
import re
import string
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import html

nltk.download('punkt')

from nltk.stem import PorterStemmer

def lowercase(text):
    return text.lower()

punct = string.punctuation
def remove_punctuation(text):
    no_punct= [words for words in text if words not in punct]
    words_wo_punct = ''.join(no_punct)

    return words_wo_punct

def remove_unnecessary_char(text):
```

```
    text = re.sub('\n', ' ', text) # Remove every '\n'
    text = re.sub('user', ' ', text) # Remove every username
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', ' ', text) # Remove every
url
    text = re.sub('  +', ' ', text) # Remove extra spaces
    text = re.sub('@[\w_]+', ' ', text) # Remove @
    text = re.sub('#[\w_]+', ' ', text) # Remove tag hashtag
    text = re.sub('Ã¢Â€Â™', "'",text)
    text = re.sub('â', "",text)
    text = re.sub('\d+', '', text)

    return text

def remove_nonaplhanumeric(text):
    text = re.sub('[^0-9a-zA-Z]+', ' ', text)
    text = re.sub(r"(.)\1{2,}", r"\1", text)
    return text

def remove_stopword(text):
    text = ' '.join([ ' ' if word in id_stopword_dict.stopword.values else word
for word in text.split(' ')])
    text = re.sub('  +', ' ', text) # Remove extra spaces
    text = text.strip()
    return text

def stemming(text):
    # Tokenisasi teks menjadi kata-kata
    words = word_tokenize(text)

    # Inisialisasi stemmer Porter
    porter = PorterStemmer()

    # Melakukan stemming pada setiap kata
    stemmed_words = [porter.stem(word) for word in words]

    # Menggabungkan kata-kata yang telah distem menjadi sebuah teks kembali
    stemmed_text = ' '.join(stemmed_words)

    return stemmed_text

import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

def english_lemmatization(text):
    # Tokenisasi teks menjadi kata-kata
    words = word_tokenize(text)
```

```
    # Inisialisasi lemmatizer WordNet
    lemmatizer = WordNetLemmatizer()

    # Melakukan lemmatization pada setiap kata
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]

    # Menggabungkan kata-kata yang telah dilemat menjadi sebuah teks kembali
    lemmatized_text = ' '.join(lemmatized_words)

    return lemmatized_text

from nltk.stem import LancasterStemmer

def lancaster_stemming(text):
    words = word_tokenize(text)
    lancaster = LancasterStemmer()
    stemmed_words = [lancaster.stem(word) for word in words]
    stemmed_text = ' '.join(stemmed_words)
    return stemmed_text

import contractions
import pandas as pd

# Fungsi untuk memperluas kontraksi
def expand2_contractions(text):
    return contractions.fix(text)

def convert_abbrev(text):
    return abbreviations[text.lower()] if text.lower() in abbreviations.keys()
else text

def preprocess(text):
    text = html.unescape(text)
    text= convert_abbrev(text)
    text = lowercase (text) # 1
    text = remove_unnecessary_char(text)
    text = remove_nonaplhanumeric(text)
    text = expand2_contractions(text)
    text = remove_stopword (text)
    text= remove_punctuation(text)
    text = english_lemmatization(text)
    return text

df['preprocess'] = df['preprocess'].apply(preprocess)
#================================================================================
==================
import pandas as pd
```

```
data1 = pd.read_csv("clean.csv")
data1

data1['preprocess'].fillna('', inplace=True)
data1['Tweet'].fillna('', inplace=True)

list_corpus = data1["preprocess"].tolist()
list_labels = data1["Suicide"].tolist()

X_train, X_test, Y_train, Y_test = train_test_split(list_corpus, list_labels,
test_size=0.2)

def cv(data):
    count_vectorizer = CountVectorizer()

    emb = count_vectorizer.fit_transform(data)

    return emb, count_vectorizer

X_train_counts, count_vectorizer = cv(X_train)
X_test_counts = count_vectorizer.transform(X_test)

#============================================================================
from sklearn.svm import LinearSVC
# Inisialisasi model
modelLSBOW = LinearSVC()

# Latih model dengan data pelatihan
modelLSBOW.fit(X_train_counts, Y_train)

from    sklearn.metrics    import    accuracy_score,    classification_report,
confusion_matrix
import scikitplot as skplt

# Prediksi menggunakan data pengujian
yy_predd = modelLSBOW.predict(X_test_counts)

root = Tk()
SUICIDEIntro = IntroLogin(root)
root.mainloop()
```