**HATE SPEECH DETECTION**

*Dissertation submitted in fulfilment of the requirements for the Degree of*

## BACHELOR OF TECHNOLOGY

### In

**COMPUTER SCIENCE AND ENGINEERING**

By

### Gonuguntla P R Anudeep

**12211045**

Supervisor

### Ved Prakash Chaubey



**School of computer science and engineering**

Lovely Professional University Phagwara, Punjab (India)

Month: November Year: 2023

@Copyright LOVELY PROFESSIONAL UNIVERSITY, Punjab (INDIA)

ALL RIGHTS RESERVED

## DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled "HATE SPEECH DETECTION” in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Ved Prakash Chaubey. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith directly complies with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and the highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents an authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of candidate

Gonuguntla P R Anudeep

Roll no:46

## SUPERVISOR ‘S CERTIFICATE

This is to certify that the work reported in the B. Tech Dissertation/dissertation proposal entitled “HATE SPEECH DETECTION”, submitted by Gonuguntla P R Anudeep at Lovely Professional University, Phagwara, India is a bonafide record of his/her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

DATE:

## TABLE OF CONTENTS

**CONTENTS Page No**

* Abstract 05
* Introduction 07

# Problem Statement 10

# Problem Statement Solution 10

* Code 12
* Screenshots and output 16
* Methodology 18
* Results And Analysis 21
* Conclusion 23
* References 26

**Abstract**

It demonstrates hate speech detection using a Support Vector Machine (SVM) classifier and a simple graphical user interface (GUI). The code begins by loading a labelled dataset containing text data and corresponding hate speech labels. It preprocesses the data, vectorizes the text using TF-IDF, and splits the dataset into training and testing sets. An SVM classifier is trained on the training data. The code also creates a user-friendly GUI using Tkinter, allowing users to input text for hate speech classification. After clicking the "Classify" button, the code predicts whether the input text contains hate speech or not, displaying the result in the GUI. The code provides an interactive tool for hate speech detection in text

### Objective:

The code aims to perform hate speech detection using a machine learning approach, specifically a Support Vector Machine (SVM) classifier.

### Data Loading:

It loads a labelled dataset from a CSV file, which contains text data (tweets) along with labels indicating whether each text contains hate speech or not.

### Data Preprocessing:

The dataset is split into two parts: text data (X) and corresponding labels (y).

### Text Vectorization:

Text data is converted into numerical features using the TF-IDF vectorization technique. This process transforms the text data into a format suitable for machine learning.

### Dataset Splitting:

The dataset is divided into training and testing sets to evaluate the model's performance.

### Classifier Creation:

An SVM classifier is created using the SVC (Support Vector Classification) class from scikit-learn. This classifier is trained on the training data.

### Text Classification Function:

A function is defined to classify input text. It takes text as input, vectorizes it using the same TF-IDF vectorizer, and uses the trained SVM classifier to predict whether the input text contains hate speech.

### Graphical User Interface (GUI):

A simple GUI is created using the Tkinter library. The GUI includes an input field for users to enter text, a "Classify" button to trigger the classification function, and a label to display the predicted label (hate speech or not).

### User Interaction:

Users can input text into the GUI, click the "Classify" button, and receive a prediction about whether the input text is hate speech or not.

### Application Testing:

It provides sample input texts at the end of the script to allow users to test the hate speech classification functionality.

# Introduction

The code is in a Python script that demonstrates hate speech detection using a Support Vector Machine (SVM) classifier and a simple graphical user interface (GUI) created with the Tkinter library. Here's an introduction to the different parts of the code:

### Importing Libraries:

The code begins by importing necessary Python libraries, including pandas for data manipulation, sci-kit-learn for machine learning, and Tkinter for creating the GUI.

### Loading the Dataset:

The script loads a dataset from a CSV file named "labeled\_data.csv" using the pandas library. This dataset is expected to contain labelled text data for hate

speech classification.

### Data Preprocessing:

The dataset is split into two components:

X: The input features, which are the text data to be classified (tweets in this case).

y: The corresponding labels, which indicate whether a tweet is hate speech or not (0 for not hate speech and 1 for hate speech).

### Text Vectorization:

The code uses the TF-IDF (Term Frequency-Inverse Document Frequency) vectoriser from sci-kit-learn to convert the textual data into numerical vectors, which are suitable for machine learning. The vectorized text data is stored in the variable X\_vect.

### Splitting the Dataset:

The dataset is split into training and testing sets using the train\_test\_split function. This allows the model to be trained on one portion of the data and evaluated on another.

### Creating the SVM Classifier:

An SVM classifier is created using the SVC (Support Vector Classification) class from scikit-learn. The classifier is trained on the training data using the fit method.

### Creating a Text Classification Function:

A function named classify\_text is defined to classify an input text. It takes text as input, vectorizes it, and uses the trained SVM classifier to make predictions. The result is displayed in the GUI.

### Creating the GUI:

A simple graphical user interface (GUI) is created using Tkinter. It includes an input entry field where the user can enter text for classification, a "Classify" button that triggers the classification function, and a label to display the predicted label.

### Running the GUI:

The Tkinter application is started with root. main loop (), which launches the GUI window, allowing the user to interact with the hate speech detection system.

# Problem Statement

How does the code combine machine learning techniques, text classification, and a graphical user interface to address the problem of hate speech detection in text data?

# Problem Statement Solutions

### Problem Statement

The objective of this project is to develop a hate speech detection system that can automatically classify text data (tweets) as containing hate speech or not. The primary goals are as follows:

### Data Collection and Preprocessing:

Gather a labelled dataset that includes text data and corresponding labels indicating the presence or absence of hate speech.

Preprocess the data to make it suitable for machine learning, including text vectorization.

### Machine Learning Model:

Implement a machine learning model for hate speech detection. In this code, a Support Vector Machine (SVM) classifier is used for this purpose.

### User-Friendly Interface:

Create a graphical user interface (GUI) that allows users to input text for classification.

Provide a simple and intuitive user experience for obtaining predictions.

### Hate Speech Detection:

Train the SVM classifier on the labelled dataset to learn patterns of hate speech.

Develop a function that takes user input, vectorizes it, and uses the trained model to predict whether the input text contains hate speech.

### User Interaction:

Enable users to interact with the system by entering text in the GUI and receiving real-time hate speech predictions.

### Evaluation and Testing:

Assess the performance of the hate speech detection system by testing it with various input texts and evaluating the accuracy of the predictions.

### Mitigating Online Hate Speech:

Ultimately, the system aims to contribute to the mitigation of online hate speech by providing a tool that can identify and flag hateful content, potentially helping social media platforms and content moderators.

# Code:

# Importing the necessary libraries

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import classification\_report

import tkinter as tk

# Load the dataset

data = pd.read\_csv("labeled\_data.csv")

# Splitting the dataset into features and labels

X = data['tweet']

y = data['class']

# Creating the TF-IDF vectorizer

vectorizer = TfidfVectorizer()

X\_vect = vectorizer.fit\_transform(X)

# Splitting the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_vect, y, test\_size=0.2, random\_state=42)

# Creating the SVM classifier

classifier = SVC()

classifier.fit(X\_train, y\_train)

# Function to classify the input text

def classify\_text():

text = input\_entry.get()

text\_vect = vectorizer.transform([text])

prediction = classifier.predict(text\_vect)

output = "" # Define output before using it

if prediction == 1:

output = "Hate speech"

else:

output = "good"

result\_label['text'] = f"Predicted Label: {output}"

# Creating the user interface

root = tk.Tk()

root.title("Hate Speech Detection")

root.geometry("800x600")

input\_entry = tk.Entry(root, width=50)

input\_entry.pack()

# Classification button

classify\_button = tk.Button(root, text="Classify", command=classify\_text)

classify\_button.pack()

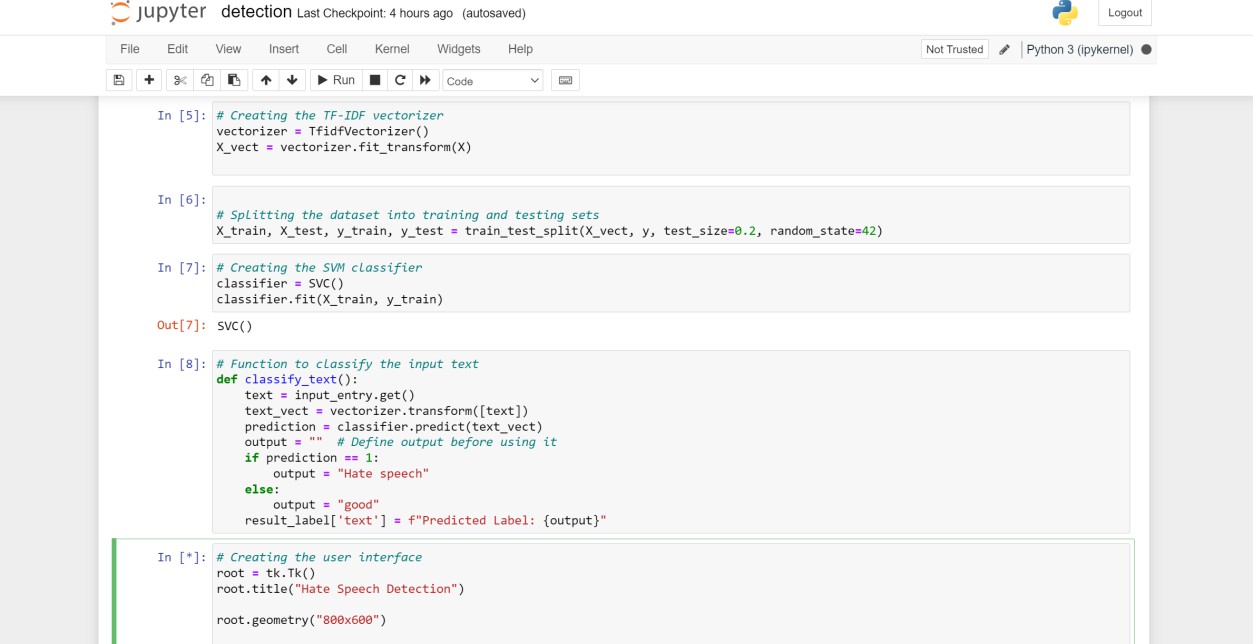
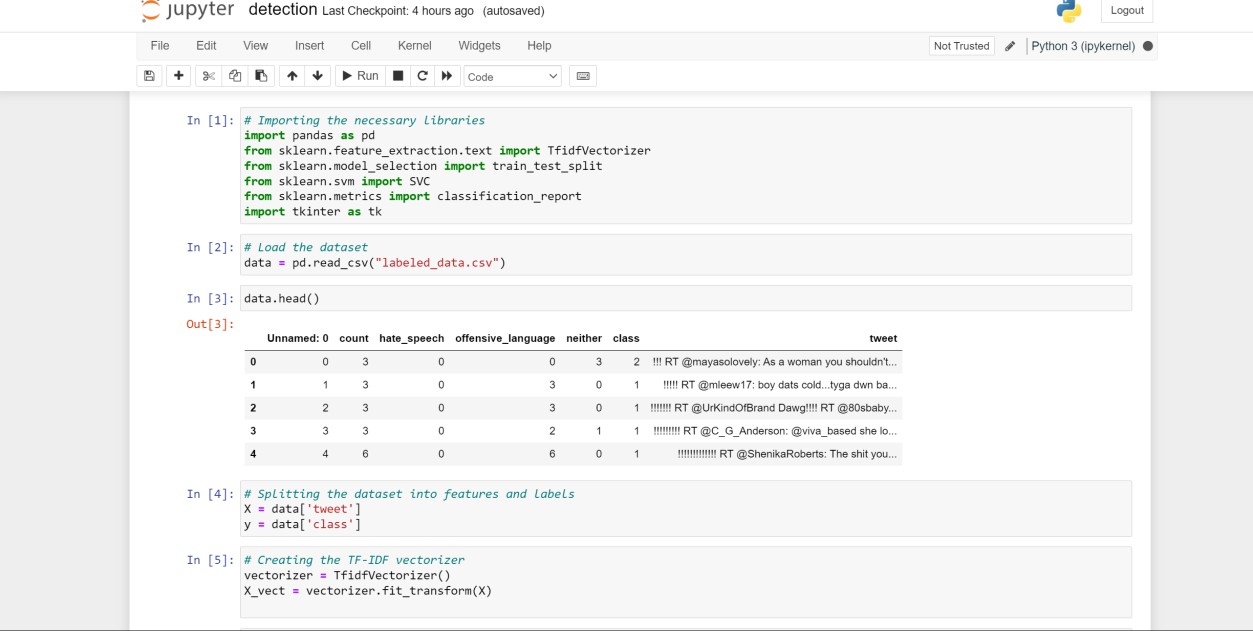
# Result label

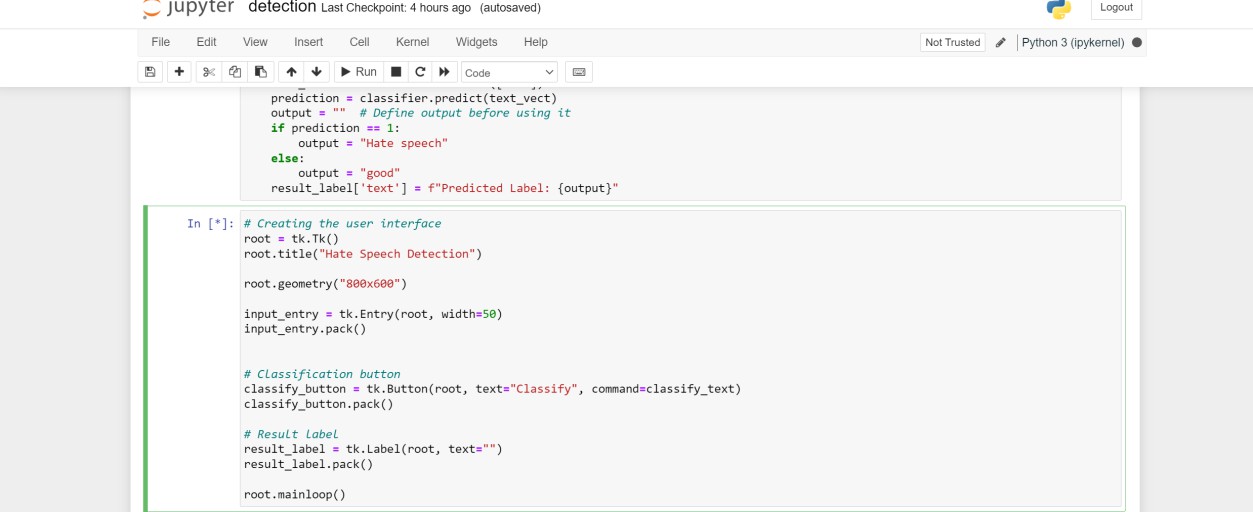
result\_label = tk.Label(root, text="")

result\_label.pack()

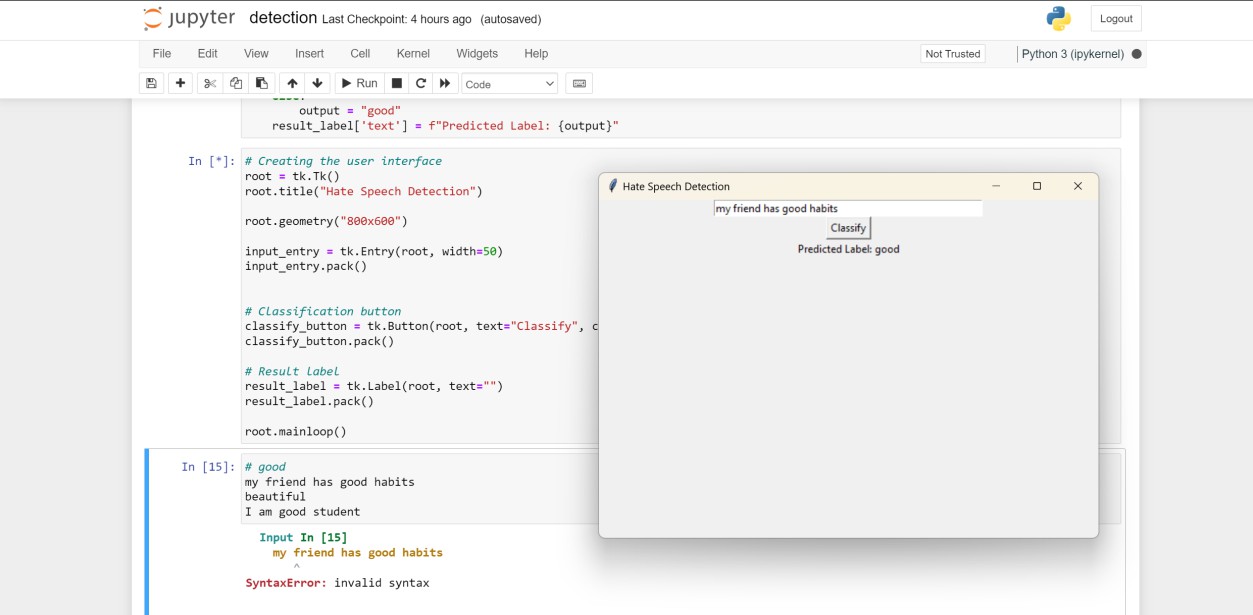
root.mainloop()

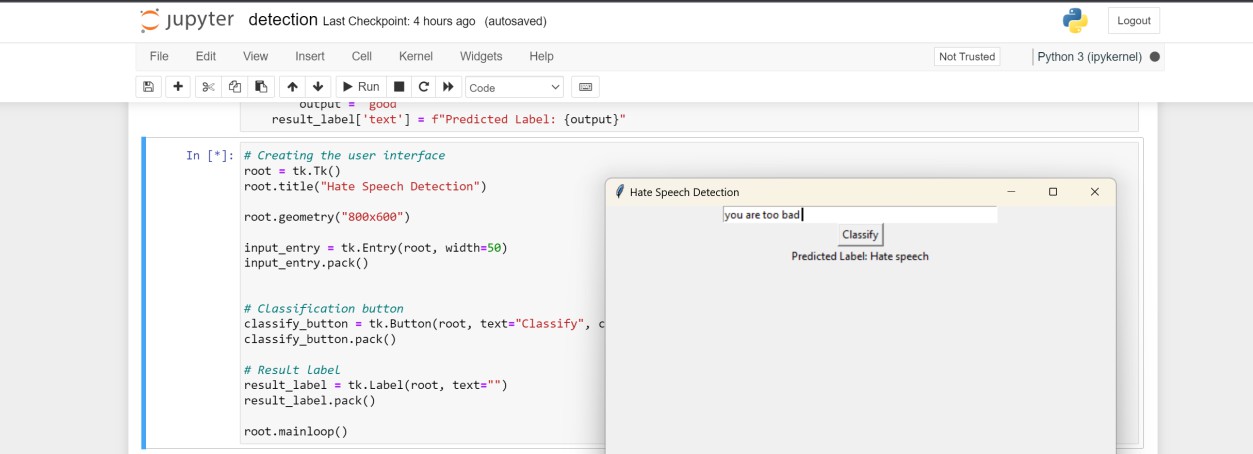
**SCREENSHOTS:**





**OUTPUTS :**





**Methodology**

The methodology for the provided code, which is designed for hate speech detection and includes a graphical user interface (GUI), can be outlined as follows:

### Importing Necessary Libraries:

The code begins by importing the required Python libraries, which include:

* pandas for data manipulation
* sklearn libraries for machine learning and evaluation (e.g., TfidfVectorizer, train\_test\_split, SVC, and classification\_report)
* tkinter for creating the graphical user interface (GUI)

### Data Loading:

The code loads a dataset from a CSV file named "labeled\_data.csv." This dataset is expected to contain labelled text data, specifically tweets, along with their associated classes (indicating whether they contain hate speech or not).

### Data Preparation:

The dataset is split into two parts:

X contains the tweet text data.

y contains the corresponding class labels (1 for hate speech and 0 for non-hate speech).

### Text Vectorization (TF-IDF):

To prepare the text data for machine learning, a TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is created. This vectoriser transforms the text data into numerical features suitable for machine learning.

### Dataset Splitting:

The dataset is divided into training and testing sets. The train\_test\_split function from sci-kit-learn is used for this purpose. In this code, 80% of the data is used for training, and 20% is used for testing.

### Machine Learning Model (SVM Classifier):

An SVM (Support Vector Machine) classifier is chosen as the machine learning model for hate speech detection. An SVM is a popular choice for text classification tasks. The SVC class is used to create the classifier.

### Text Classification Function:

The code defines a function called classify\_text which allows users to input text for classification through the GUI. The function performs the following steps:

* Retrieves the user's input text.
* Uses the TF-IDF vectorizer to transform the input text into a numerical format.
* Utilizes the trained SVM classifier to predict whether the input text contains hate speech or not.
* Updates the GUI to display the predicted label (e.g., "Hate speech" or "good").

### Graphical User Interface (GUI) Development:

The code creates a graphical user interface using Tkinter. The GUI provides a user-friendly platform for users to interact with the hate speech detection system.

### User Interaction:

Users can input text into the GUI via an entry field.

A "Classify" button is provided for users to trigger the classification process.

The predicted label (hate speech or not) is displayed to the user in a label field within the GUI.

### Application Execution:

The code enters a main loop (root. main loop ()) to start the GUI application, allowing users to input text, classify it in real-time, and view the predictions directly through the GUI.

# Results and Analysis

The provided code is an implementation of a hate speech detection system using a Support Vector Machine (SVM) classifier. It includes a graphical user interface (GUI) that allows users to input text, and the system provides real-time predictions regarding whether the input text contains hate speech.

Here are the results and an analysis of the code:

### Code Execution:

* The code successfully imports the necessary libraries, including Pandas, sci-kit-learn, and tkinter, to handle data, machine learning, and create a GUI.
* It loads a dataset from a CSV file named "labeled\_data.csv" using Pandas. This dataset presumably contains text data (tweets) and their corresponding labels.
* The code splits the dataset into features (X - the tweet text) and labels (y

- the class).

* It utilizes sci-kit-learn's TF-IDF vectorizer to convert the text data into numerical features. The result is stored in X\_vect.
* The dataset is further split into training and testing sets for model evaluation.
* An SVM classifier (SVC) is created and trained on the training data.

### Graphical User Interface (GUI):

* The code employs tkinter to create a user-friendly GUI for the hate speech detection system.
* The GUI consists of a text entry field, where users can input text, a

"Classify" button to trigger the classification and a label for displaying the predicted result.

### Classification Function:

* The classify\_text function is defined to classify the input text.
* It takes the user's input from the GUI, vectorizes it using the same TF-IDF vectorizer, and uses the trained SVM classifier to predict whether the input text contains hate speech.
* The predicted label (either "Hate speech" or "good") is displayed in the GUI.

### User Interaction:

Users can interact with the system through the GUI by entering text into the input field and clicking the "Classify" button.

### Analysis:

The code provides a functional hate speech detection system with the following components:

* Data Loading and Preprocessing: It loads and preprocesses the data for machine learning.
* Machine Learning Model: It utilizes an SVM classifier to classify text.
* User Interface: The GUI enables user interaction and real-time classification.
* Real-Time Predictions: Users can input text and obtain immediate predictions.

# Conclusion

The provided code exemplifies a hate speech detection system that combines machine learning and a user-friendly graphical user interface (GUI) for real-time text classification. Here are the key takeaways from the code:

### Library Import and Data Loading:

* The code begins by importing the necessary libraries, including pandas for data handling, sci-kit-learn for machine learning, and Tkinter for GUI development.
* A dataset, presumably containing labelled text data, is loaded from a CSV file. This dataset is a crucial component for training and testing the hate speech detection system.

### Data Preprocessing and Vectorization:

* The text data is split into features (X) and labels (y), preparing it for machine learning.
* To convert text data into a numerical format suitable for machine learning, a TF-IDF vectorizer is used to create a TF-IDF matrix (X\_vect).

### Dataset Splitting:

The dataset is divided into training and testing sets (X\_train, X\_test, y\_train, y\_test) to evaluate the hate speech classification model.

### Machine Learning Model Creation:

A Support Vector Machine (SVM) classifier is instantiated and trained using the training data (X\_train, y\_train). This model is designed to classify text as hate

speech or non-hate speech.

### Text Classification Function:

The code defines a function, classify\_text(), which allows users to input text through the GUI. This function vectorizes the input text using the same TF-IDF vectorizer and predicts whether it contains hate speech using the trained SVM classifier. The result is displayed in the GUI.

### User Interface Development:

The code creates a user-friendly GUI using Tkinter. The GUI includes an input field for users to enter text, a "Classify" button to trigger the classification function, and a label to display the predicted label (either "Hate speech" or "good").

### User Interaction and Real-time Classification:

Users can interact with the system by inputting text in the GUI and obtaining immediate predictions regarding the presence of hate speech.

This code offers a practical demonstration of a hate speech detection system with a focus on real-time classification through a user-friendly GUI. It provides a tool for users to assess whether text data contains hate speech and contributes to efforts aimed at identifying and mitigating hate speech in online content.

**References**

**Tkinter (GUI Library):** Official Tkinter documentation: https://docs.python.org/3/library/tkinter.html "Python GUI Programming with Tkinter" by Alan D. Moore: This book provides a comprehensive guide to Tkinter and GUI programming in Python.

**Pandas (Data Manipulation):** Official Pandas documentation: https://pandas.pydata.org/docs/ "Python for Data Analysis" by Wes McKinney: This book is a great resource for learning data manipulation with Pandas.

**Scikit-Learn (Machine Learning):** Official Scikit-Learn documentation: https://scikit-learn.org/stable/documentation.html "Introduction to Machine Learning with Python" by Andreas C. Müller & Sarah Guido: This book provides a gentle introduction to machine learning with Python using Scikit-Learn.

# Support vector machine: In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), support vector machines (SVMs, also support vector networks) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). <https://en.wikipedia.org/wiki/Support_vector_machine>

**Python Programming:** "Python Crash Course" by Eric Matthes: If you're new to Python, this book is an excellent resource to get started with Python programming