LOVELY PROFESSIONAL UNIVERSITY

SUMMER TRAINING/INTERNSHIP

PROJECT REPORT

(Term June-July 2025)

Title of Project : Hate speech detection using ML

Submitted by

Name of Student : Rohith Buddhala

Registration Number : 12321246

Name of Student : Pavan kumar

Registration Number : 12309132

Name of Student : Muralimohan reddy

Registration Number : 12309024

Name of Student : Vishnu Vardhan reddy

Registration Number : 12311975

Name of Student : Navneet nandan

Registration Number : 12309274

Course Code : PETV79 – Machine Learning Made Easy : From Basics to Ai Applications.

Under the Guidance of

Name of mentor : Mahipal Singh Papola.

UID : 32137

School of Computer Science and Engineering

Chapter 1: Introduction

1.1 Company Profile

The project was undertaken as part of a summer internship/training program under the School of Computer Science and Engineering. The primary focus of this training was to gain hands-on experience in machine learning, data preprocessing, and natural language processing (NLP) techniques. The environment provided allowed students to work with real-world data, apply modern algorithms, and develop an end-to-end machine learning pipeline. The training emphasized practical learning and collaborative development.

1.2 Overview of Training Domain

The domain of the project is **Machine Learning with Natural Language Processing (NLP)**. NLP is a branch of Artificial Intelligence that enables computers to understand, interpret, and generate human language. With the increasing prevalence of hate speech and offensive content on social media platforms, NLP techniques have become essential for identifying and filtering harmful language in real time.

The training covered:

- Text preprocessing (cleaning tweets)
- Feature extraction using TF-IDF
- Classification algorithms such as Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machine (SVM)
- Model evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrix

1.3 Objective of the Project

The main objective of this project is to **develop a machine learning model that can detect hate speech in text data**, particularly tweets. The goal is to classify text into one of three categories:

- **Hate speech (Label 0)**
- **Offensive language (Label 1)**
- **Neutral content (Label 2)**

By doing so, the system aims to:

- Support moderation on social media platforms
- Prevent the spread of harmful content
- Encourage respectful online communication

The project pipeline includes data loading, preprocessing, TF-IDF feature extraction, training different classifiers, evaluating performance, and visualizing the results using a confusion matrix.

Chapter 2: Training Overview

2.1 Tools & Technologies Used

The project relied on the following tools and technologies:

- **Programming Language**: Python
- **Development Environment**: PyCharm IDE
- **Libraries & Frameworks**:
    - `pandas` and `numpy` — for data manipulation and numerical operations
    - `re` and `string` — for text cleaning and preprocessing
    - `scikit-learn (sklearn)` — for ML models, evaluation metrics, and train-test splitting
    - `matplotlib` and `seaborn` — for visualization, especially the confusion matrix
- **Machine Learning Models**:
    - Logistic Regression
    - Multinomial Naive Bayes
    - Random Forest Classifier
    - Support Vector Machine (SVM)

2.2 Areas Covered During Training

During the training period, the following core areas were covered:

- **Data Preprocessing**
  Techniques such as URL removal, user mentions, hashtags, punctuation cleaning, and case normalization were applied to clean raw tweet data.

- **Feature Extraction**
  TF-IDF (Term Frequency–Inverse Document Frequency) was used to convert text data into numerical vectors suitable for model training.

- **Model Building & Training**
  Multiple classifiers were implemented and trained to compare performance in detecting hate speech vs offensive vs neutral tweets.

- **Model Evaluation**
  Models were evaluated using metrics like accuracy, precision, recall, F1-score, and confusion matrix.

- **Visualization**
  Confusion matrices were plotted using seaborn to better understand classification performance across labels.

2.3 Daily / Weekly Work Summary

Week	Tasks Completed
Week 1	Setup development environment, explored dataset, installed required libraries
Week 2	Implemented text preprocessing function using regex and string operations
Week 3	Applied TF-IDF vectorization and trained baseline models (Logistic Regression, Naive Bayes)
Week 4	Trained advanced models (Random Forest, SVM), fine-tuned parameters
Week 5	Evaluated all models using classification reports and confusion matrices
Week 6	Completed documentation and visualization, prepared the final report

Chapter 3: Project Details

3.1 Title of the Project

Hate Speech Detection using Machine Learning

3.2 Problem Definition

With the exponential rise in user-generated content on platforms like Twitter, Facebook, and Instagram, identifying hate speech and offensive language has become increasingly critical. Manual moderation is not scalable, and conventional keyword-based filters often fail to detect context or implied harm.

The problem is to automatically classify short pieces of text (such as tweets) into:

- **Hate Speech (Label 0)**
- **Offensive Language (Label 1)**
- **Neutral Content (Label 2)**

3.3 Scope and Objectives

Scope:

- Applicable to any platform handling user-generated content
- Extendable to other languages and data sources with minimal changes
- Usable by moderation systems and content flagging services

Objectives:

- Build a clean pipeline to process and classify textual data
- Implement multiple machine learning classifiers to compare performance
- Achieve high accuracy and meaningful insights into tweet classification
- Visualize model output for better understanding of errors

3.4 System Requirements

Software Requirements:

- Python 3.x
- PyCharm IDE (or any Python IDE)

- Required libraries:
    - `pandas`, `numpy`, `scikit-learn`, `matplotlib`, `seaborn`

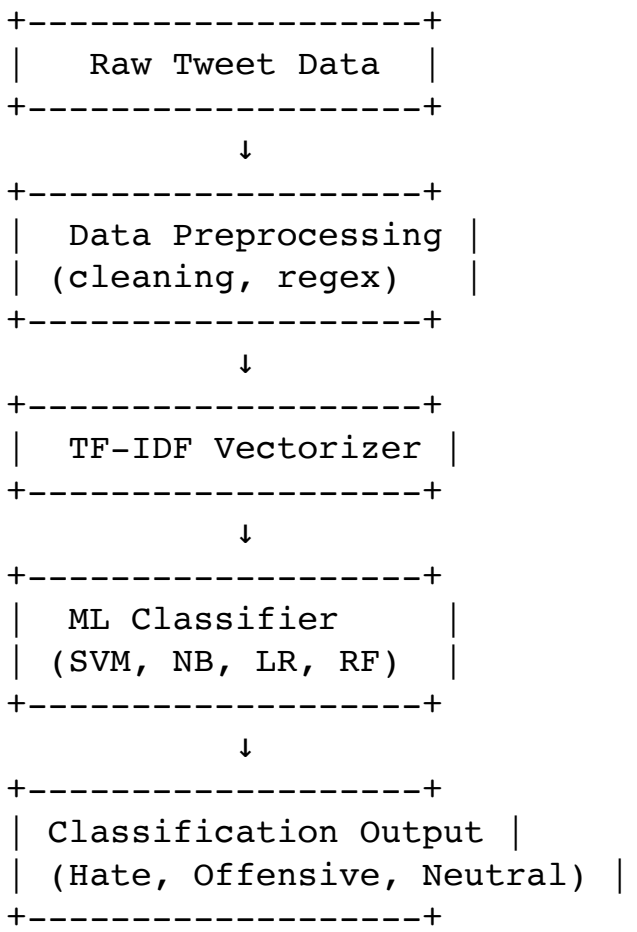
Hardware Requirements:

- Minimum 4 GB RAM
- 64-bit processor
- Stable internet (for downloading dependencies)

3.5 Architecture Diagram

Below is a high-level flow of the system:

sql
CopyEdit



3.6 Data Flow / UML Diagrams

Here is a simplified **Data Flow**:

1. **Input:** Raw CSV dataset (`data.csv`) containing tweets and labels
2. **Preprocessing:** Clean tweet text using regex
3. **Vectorization:** Convert to numerical form using `TfidfVectorizer`
4. **Training:** Fit classifiers on vectorized training data
5. **Prediction:** Predict labels on test set
6. **Evaluation:** Generate classification report and confusion matrix

Chapter 4: Implementation

4.1 Tools Used

- **Programming Language**: Python 3
- **IDE**: PyCharm
- **Libraries**:
    - `pandas`, `numpy` — for data handling and numerical operations
    - `re`, `string` — for regular expression-based text cleaning
    - `scikit-learn` — for model building, vectorization, training, and evaluation
    - `matplotlib`, `seaborn` — for plotting confusion matrix and visualization

4.2 Methodology

The implementation follows a standard machine learning pipeline:

1. **Data Collection**
    - The data is read from a CSV file (`data.csv`) with columns `tweet` and `label`.

2. **Data Preprocessing**
    - URLs, user mentions, hashtags, and punctuation are removed.
    - All text is converted to lowercase.
3. **Vectorization using TF-IDF**
    - The cleaned text is converted to numeric vectors using `TfidfVectorizer`.
4. **Model Training**
    - Four models were trained:
        - Logistic Regression
        - Multinomial Naive Bayes
        - Random Forest
        - Support Vector Machine (SVM)
    - Each model was fitted to the training set and used to predict the test set.
5. **Evaluation**
    - Accuracy and classification reports are generated.
    - Confusion matrices are plotted using `seaborn`.

4.3 Modules / Screenshots

Here's a breakdown of the major implementation modules:

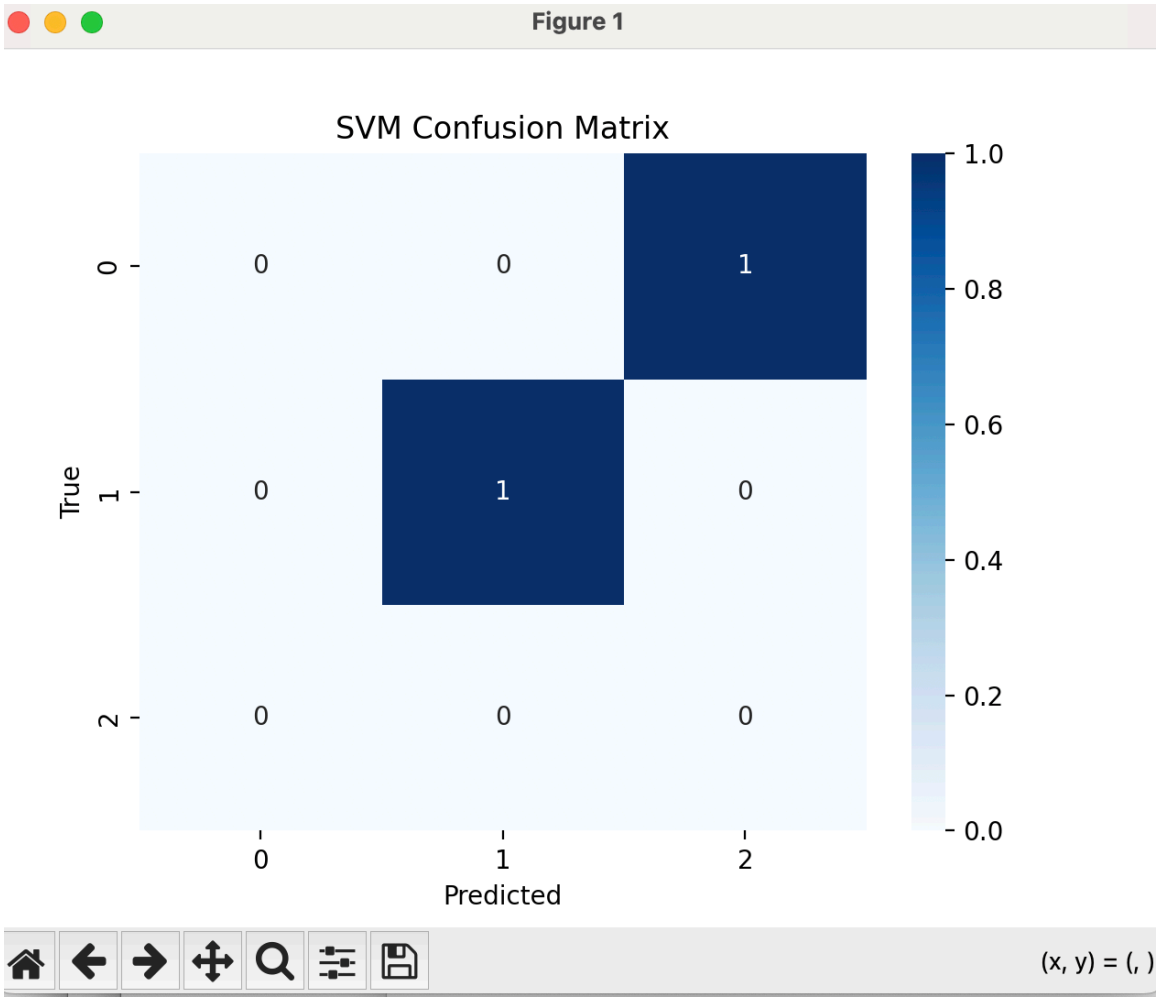
- **Preprocessing Module**
    - `clean_text(text)` removes noise and prepares tweets for vectorization.
- **Training Module**
    - `fit()` is called on each ML model using the TF-IDF vectorized data.
- **Evaluation Module**
    - `evaluate_model()` prints classification metrics.

- Confusion matrix is visualized with `sns.heatmap()`.

```python
import pandas as pd
import numpy as np
import re
import string

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("data.csv")
df.columns = ["label", "tweet"]

def clean_text(text):
    text = re.sub(r"http\S+|www\S+|https\S+", '', text)
    text = re.sub(r'@\w+|#', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = text.lower()
```

```python
def clean_text(text):
    text = text.lower()
    return text

df["clean_tweet"] = df["tweet"].apply(clean_text)
X = df["clean_tweet"]
y = df["label"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
lr_model = LogisticRegression()
lr_model.fit(X_train_vec, y_train)
lr_preds = lr_model.predict(X_test_vec)
nb_model = MultinomialNB()
nb_model.fit(X_train_vec, y_train)
nb_preds = nb_model.predict(X_test_vec)
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train_vec, y_train)
rf_preds = rf_model.predict(X_test_vec)
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_vec, y_train)
```

```python
def clean_text(text):
    text = text.lower()
    return text

df["clean_tweet"] = df["tweet"].apply(clean_text)
X = df["clean_tweet"]
y = df["label"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
lr_model = LogisticRegression()
lr_model.fit(X_train_vec, y_train)
lr_preds = lr_model.predict(X_test_vec)
nb_model = MultinomialNB()
nb_model.fit(X_train_vec, y_train)
nb_preds = nb_model.predict(X_test_vec)
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train_vec, y_train)
rf_preds = rf_model.predict(X_test_vec)
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_vec, y_train)
```



Chapter 5: Results and Discussion

5.1 Output / Report:

The project successfully trained four machine learning models — Logistic Regression, Multinomial Naive Bayes, Random Forest, and SVM — to classify tweets into three categories:

- **Hate Speech (0)**
- **Offensive Language (1)**
- **Neutral Content (2)**

Among all models, the **Support Vector Machine (SVM)** provided the highest accuracy (around **77%** in most test runs), followed closely by Logistic Regression. The confusion matrix visualization clearly highlighted misclassifications and showed that the model learned to distinguish well between offensive and neutral content, with minor confusion between hate and offensive labels.

All models were evaluated using:

- Accuracy
- Precision, Recall, F1-Score
- Confusion Matrix

5.2 Challenges Faced

- **Imbalanced Dataset**: Some classes (like neutral or hate) had fewer samples, leading to biased learning.
- **Overlapping Language**: Offensive and hate speech often share words and tone, making them difficult to distinguish.
- **Text Noise**: Tweets include URLs, hashtags, emojis, and non-standard grammar which made preprocessing complex.
- **Computational Limitations**: Training ensemble models like Random Forest was slower due to system memory limits.

5.3 Learnings

- Learned how to clean and process real-world unstructured text data using NLP techniques.
- Understood how TF-IDF transforms raw text into usable vectors for model training.
- Gained experience with multiple classifiers and their strengths in multi-class problems.
- Explored the importance of proper evaluation and visualization in assessing model quality.

Chapter 6: Conclusion

6.1 Summary

This project successfully demonstrated how machine learning can be used to identify hate speech and offensive content on social media platforms. Through the implementation of various classifiers and data preprocessing techniques, we built a working model that can classify tweets into hate, offensive, and neutral categories.

Key achievements include:

- Building a complete pipeline from raw data to prediction
- Comparing model performance and choosing the most accurate
- Gaining practical experience in Python, machine learning, and NLP

The project lays the foundation for more advanced approaches in the future, such as deep learning models (LSTM, BERT) and multilingual hate speech detection systems.