Progress Report: Hate Speech Detection in Social Media

1. Introduction

This report explains the progress of my individual research project titled **"Hate Speech Detection in Social Media."** The main aim of the project is to study how well a **Decision Tree algorithm** can detect hate speech in short and informal texts like tweets.

Social media posts are often written in casual language, with slang, emojis, and misspellings. This makes them difficult for machines to understand. Detecting hate speech is especially challenging because it may look similar to offensive or sarcastic language. For this reason, I use tools from **Natural Language Processing (NLP)** to help the model understand these short texts.

In this project, I follow a simple weekly plan that started with research and then moved on to working with the dataset, learning the Decision Tree method, and building a basic working model. This report covers what I have completed so far, what results I got, and what steps I will take next.

2. Summary of Work Completed

Here is a summary of my tasks, when they were planned, and the current status:

Task	Planned Week	Status
Review Literature	Week 1	Completed
Learn Decision Tree	Week 2	Completed
Explore Dataset	Week 3	Completed
Build a Basic Model	Week 4	Completed
Evaluate Model	Week 5	In Progress
Compare with Other Models	Week 6	Not Started

Final Report	Week 6	Not Started

3. Literature Review

I started this project by reading several research papers to understand how hate speech detection works. These papers explained that many models have been used in this field, including:

- **Classical Machine Learning:** Logistic Regression, Naive Bayes, SVM
- **Deep Learning:** CNN, RNN, LSTM
- **Advanced Models:** BERT and other pre-trained language models

The most useful paper was **Davidson et al. (2017)**, which also provided the dataset I used in this project. I also studied a survey paper by **Schmidt & Wiegand (2017)** and a recent paper introducing **TweetNLP**, a tool built for social media text. These helped me understand both the data and the models.

4. Learning the Decision Tree Algorithm

In Week 2, I focused on learning the **Decision Tree** algorithm. I watched tutorial videos and followed Python examples. A Decision Tree is a simple model that splits data based on questions like:
"Does this tweet contain a hate word?"
"Is this tweet long or short?"

Each question splits the data into branches, and the final prediction is based on the path through the tree. Important concepts I learned include:

- **Entropy** – how mixed the data is
- **Information Gain** – how much a feature helps in separating the classes
- **Overfitting** – when the model is too complex and learns noise
- **Pruning** – a way to cut down the size of the tree to avoid overfitting

I chose the Decision Tree because it is easy to build, easy to explain, and good for a beginner-level research project.

5. Dataset and Preprocessing

The dataset I used is the **labeled Twitter hate speech dataset** by Davidson et al. It has over 24,000 tweets. Each tweet is labeled as one of the following:

- 0 = Hate Speech
- 1 = Offensive Language
- 2 = Neither

For my project, I only focused on **labels 0 and 1** to keep the task binary (hate or offensive). The data needed some cleaning before it could be used. I did the following steps:

- Lowercased all text
- Removed URLs and special characters
- Removed stop words
- Used **TF-IDF** to turn text into numbers for the model

I split the dataset into 80% for training and 20% for testing.

6. Model Implementation

After preparing the data, I built a **Decision Tree classifier** using Python and the Scikit-learn library. I used a tree with **maximum depth 10** to avoid overfitting.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report, confusion_matrix
```

```
# 1. Load dataset

df = pd.read_csv("labeled_data.csv")  # Replace with your actual filename

# Example: Hate=0, Offensive=1, Neither=2

df = df[df['class'].isin([0, 1])]  # Filter only hate/offensive for binary

df['label'] = df['class']  # Assuming column name is 'class'



# 2. Preprocess

df['text'] = df['tweet'].str.lower().str.replace(r"http\S+|[^a-zA-Z\s]", "",
regex=True)



# 3. Feature extraction

tfidf = TfidfVectorizer(max_features=1000)

X = tfidf.fit_transform(df['text']).toarray()

y = df['label']



# 4. Train/test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)



# 5. Train model

clf = DecisionTreeClassifier(max_depth=10, random_state=42)

clf.fit(X_train, y_train)



# 6. Evaluation

y_pred = clf.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

7. Model Results and Evaluation

After training the model, I tested it on the 20% test set. The model produced the following results:

- **Accuracy:** 93%
- **Precision (Hate Speech):** 0.41
- **Recall (Hate Speech):** 0.20
- **F1-score (Hate Speech):** 0.27
- **Precision (Offensive):** 0.95
- **Recall (Offensive):** 0.98
- **F1-score (Offensive):** 0.97

These numbers show that the model performs very well on offensive language but performs poorly on hate speech. It misses most hate tweets, which is a problem.

This is mainly because of **class imbalance** — the dataset contains far more offensive tweets than hate tweets. So, the model learns to predict the majority class most of the time. Even though the **overall accuracy is high**, the **model is not reliable** for detecting hate speech, which is the main goal of this project.

To fix this, I plan to try **resampling methods** like **SMOTE**, or use **class weights** in the model. I may also compare the Decision Tree with another algorithm like Naive Bayes to see if it gives better results.

8. Challenges Faced

1. Noisy and Short Text

Tweets are often short, full of slang, emojis, and misspellings. It was difficult to clean the data in a way that works well for machine learning.

2. Class Imbalance

Most tweets are not hate speech. This made the model biased toward the offensive class.

3. Model Limitations

A Decision Tree may be too simple for this kind of text classification. It does not understand word order or context, which are important in detecting hate.

9. Next Steps

Here are the remaining tasks for the next 1–2 weeks:

- Try techniques to balance the dataset (SMOTE, class weights)
- Try another model like Naive Bayes or SVM for comparison
- Finalize all evaluations and write the final report
- Push the code and report to GitHub
- Submit final report to Turnitin

10. Conclusion

So far, the project is going according to the original timeline. I have completed the literature review, learned the algorithm, cleaned the dataset, and built a working model. The results show that a simple Decision Tree can give good accuracy but struggles with detecting hate speech due to class imbalance and short text. I plan to improve this by testing new methods and finalizing my work for submission. This project has helped me understand the challenges of NLP in social media and the importance of fair model evaluation.