**KENYA TWEET CLASSIFIER: DETECTING REAL-TIME HATE SPEECH AND MISINFORMATION**

**CRISP DM DOCUMENTATION**

**1. PROJECT OVERVIEW**

In this project, we take a look at the increasing challenge of hate speech and misinformation on Kenyan social media, particularly on platforms like Twitter. With the rapid rise in internet and smartphone access, millions of Kenyans now engage online daily. While this has opened up new channels for communication, it has also created room for the spread of harmful content, often targeting individuals or groups based on tribe, gender, political affiliation, or religion.

This growing issue poses real social threats, especially during elections, public unrest, or national discussions where misinformation and hate can escalate tensions, incite violence, or spread fear. To help tackle this, we have developed a real-time tweet classification system that uses natural language processing (NLP) and machine learning techniques to automatically detect and flag tweets containing hate speech or misinformation.

We aim to build a system that: helps moderate content in real time, supports fact-checking efforts, and promotes a safer digital space for all users in Kenya. By identifying toxic content early, this project supports peace-building, public awareness, and more responsible use of social media.

**Business Problem**

In Kenya, Twitter has become a powerful space where people share opinions and talk about national issues. It gives citizens a voice and helps them organize around important topics like politics, health, and education. However, this freedom has also led to problems. There’s been a rise in hate speech and misinformation, especially during elections or times of political tension. Tweets that attack certain tribes, spread false news, or incite violence can go viral very fast.

This creates a serious challenge. Misinformation can mislead the public and hate speech can fuel division and even lead to real-world conflict. Kenyans need a way to stay safe and informed online. Right now, there’s no real-time system that detects harmful tweets in Kenya. That means hate speech and false information can spread without being stopped in time.

This project aims to build a tweet classifier that detects hate speech and misinformation in real time. It will help protect people online, reduce digital harm, and support peaceful, truthful conversations in Kenya’s online space.

**Business Understanding**

**Background**

Kenya has experienced an alarming rise in online hate speech and misinformation, particularly on platforms like Twitter. These threats become most visible during:

* General elections
* Nationwide protests (e.g., the 2024 Gen Z #RejectFinanceBill movement)
* Contentious policy moments and governance shifts

Such posts — often political or tribal in nature — can:

* Incite ethnic hatred
* Spread false statistics or quotes
* Erode public trust in key institutions

Despite the serious consequences, Kenya currently lacks automated tools that can:

* Detect such content in real time
* Support institutions like NCIC, IEBC, and fact-checkers in flagging threats early

Real-World Use Cases

* **NCIC** receives early alerts on rising tribal slurs tied to political hashtags
* **DCI** investigates tweets flagged for incitement, with supporting data
* **IEBC** sees a spike in false election-related claims and coordinates public statements
* A journalist pastes a suspicious tweet into our app and instantly sees it flagged as "Misinformation" with a source link
* **An NGO** uploads a CSV of tweets from their monitoring teams and gets real-time labeling + export **Project Objectives**

**Main Objective:**

To develop a real-time classification system that accurately detects hate speech and the spread of misinformation in tweets from Kenyan users, with the aim of promoting a safer and more informed online environment.

**Sub-Objectives:**

1. To Detects hate speech in tweets using Natural Language Processing (NLP)
2. To Flags misinformation by matching tweets with verified claims from PesaCheck and Africa Check
3. To Provide a real-time dashboard for tweet input, analysis, and classification
4. To Empower key institutions (NCIC, CAK, IEBC, DCI, media) to respond proactively
5. To Support peace, truth, and civic engagement in Kenya’s online spaces

**Planned Matrices of Success**

Our goal is to ensure the tool is both accurate and useful in real-world decision-making. To evaluate success, we will aim for:

* An **F1-score of 80% or higher** for hate speech classification
* **Misinformation detection**: Tweets with **≥85% similarity** to known false claims are flagged reliably
* **Usability**: Stream lit dashboard runs smoothly with real-time classification
* **Scalability**: Users can upload tweet batches and download labeled results
* **Accessibility**: The tool is usable by both technical (data teams) and non-technical stakeholders (media, civil society)

**Key Stakeholders**

The following organizations are directly affected and can benefit from our solution:

* **IEBC (Independent Electoral and Boundaries Commission)** – Track digital threats to election integrity
* **Fact-checking Organizations** – e.g., PesaCheck and Africa Check
* **Journalists & Media Houses** – Avoid unknowingly amplifying misinformation
* **Government and Regulatory** **Bodies**-Agencies like DCI, CAK, NCIC, and KFCB can use the system to monitor online discourse, reduce incitement, and promote national security and cohesion.
* **Social Media Platforms**-Companies like Meta (Facebook/Instagram), TikTok Kenya, and Twitter/X can enhance their content moderation by integrating locally trained AI models.
* **Content Moderators**-Moderation teams in firms such as Sama will benefit from faster and more accurate identification of toxic content, reducing manual workload.
* **AI and Machine Learning Experts**-Kenyan developers, researchers, and data scientists can build and improve models tailored to local languages and cultural nuances.
* **Advocacy and Human Rights Groups**-Organizations like Article 19 EA and Ushahidi can use system insights to support digital rights and safer online engagement.
* **Educational Institutions and Researchers-**Universities and think tanks will gain access to real-time data for studying online behavior, digital communication, and policy development.
* **Advertisers and Brands**-Companies like Safaricom and Equity Bank will benefit from a cleaner digital environment, avoiding association with harmful or false content.

**Problem Framing**

This is a **binary supervised learning task** for hate speech detection:

**Input**: Cleaned tweet text

**Output**: 1 = Hate speech, 0 = Safe

Misinformation detection is a **semantic similarity task**:

* Compare tweet to fact-checked claims using **TF-IDF + cosine similarity**

**2. DATA UNDESTANDING**

In this project, we are using the Kenyan Political Tweets dataset from Kaggle, which contains over 11,000 tweets discussing major political events, public figures, and issues in Kenya. We’ll this section explores:

* Column structure and data types
* Missing values
* Tweet distribution over time
* User information (location, followers, verified status)
* Text characteristics (length, hashtags, etc.)

**Load dataset**

****

import pandas as pd--This line imports the Pandas library, which is widely used in Python for data manipulation and analysis. It provides powerful data structures like Data Frames that allow for easy handling of tabular data.

df = pd. read\_csv("kenya\_political\_tweets.csv”) --Here, the read\_csv () function from Pandas is used to load a dataset named "kenya\_political\_tweets.csv" into a Data Frame called df. A Data Frame is essentially a table with rows and columns, similar to an Excel spreadsheet or a SQL table.

df. Shape--This line returns the shape of the Data Frame as a tuple. In this case, the output (11723, 16) indicates that the dataset contains: 11,723 rows, each representing a tweet related to Kenyan politics 16 columns, each corresponding to different attributes or features of the tweets, such as the tweet text, user information, time, sentiment

**Dataset Information¶**

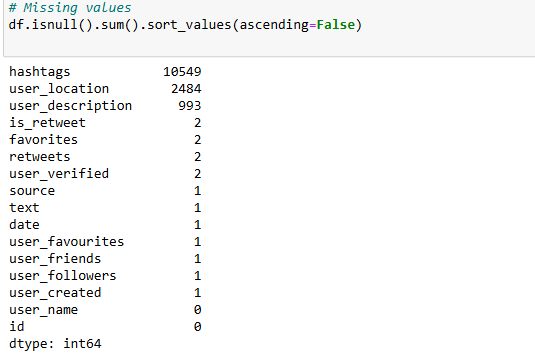
The .info () method will provide us with a summary of the dataset: Total entries, Each column name and data type and Count of non-null (non-missing) values

**Statistical Summary**

The. describe () method shows statistics like: mean, standard deviation, min, max and quartiles

**Missing Values**

We check for missing values in each column using. isnull (). sum ().



This line of code is used to **identify and count missing values** in the dataset:

* df. Isnull () creates a Data Frame of the same shape as df, where each cell contains True if the original cell is NaN (missing), and False otherwise.
* . sum () adds up the number of True values (i.e. missing values) for each column.
* . sort\_values(ascending=False) sorts the columns by the number of missing values, from highest to lowest.

The result shows the number of missing (null) values in each column of the dataset:

* The hashtags column has the most missing values (10,549), suggesting that many tweets did not contain hashtags.
* Other columns like user\_location (2,484), user\_description (993), and several others have smaller amounts of missing data.
* Columns like user\_name and id have **no missing values**, which is ideal if they are needed as identifiers or references.

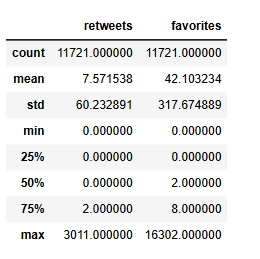
**Categorical/Text Summary**

This command generates **summary statistics** for all columns in the Data Frame that have data types classified as 'object', which typically include **categorical** or **text** data.

* The describe () function, by default, provides statistics like count, unique values, top (most frequent) value, and frequency of the top value.

The parameter include='object' ensures that the summary is applied only to non-numeric (text or categorical) columns

****

****

**Retweets:**

* Mean: approximately 7.6
* Maximum: 3,011,75% of tweets have two or fewer retweets

**Favorites**:

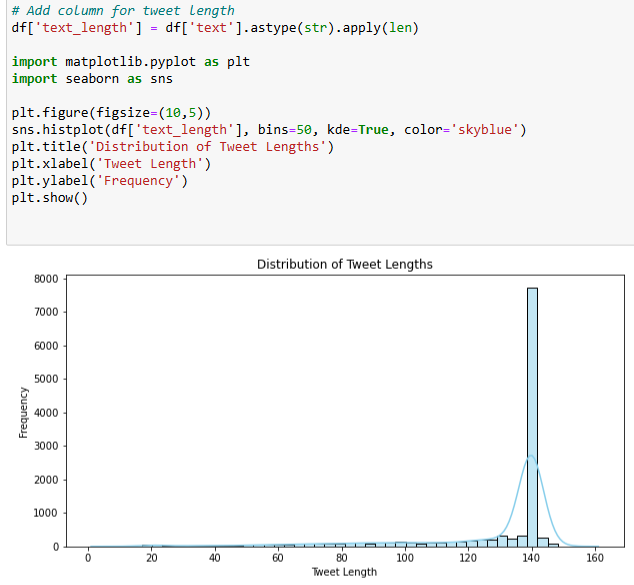
* Mean: approximately 42
* Maximum: 16,302
* 75% of tweets have eight or fewer likes

Overall, engagement is generally low, but there are a few highly viral tweets that skew the averages.

**Tweet Length Distribution**

We calculate the number of characters in each tweet and plot a histogram. This shows:

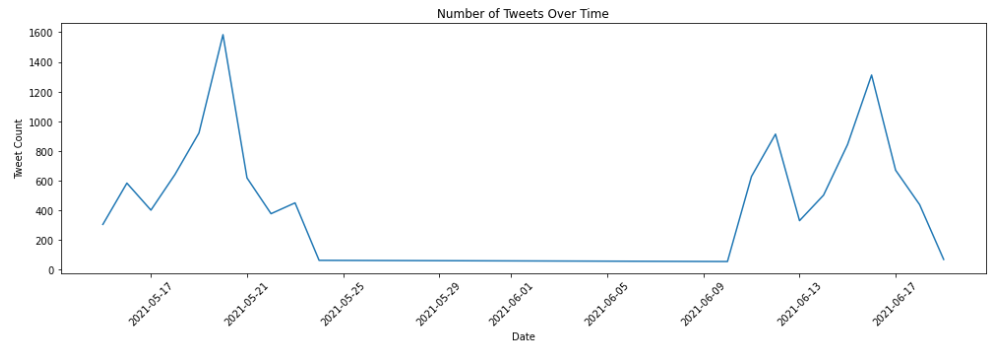
* The typical length of tweets
* Outliers (very short or long tweets)
* Helps in designing token limits for NLP models



* The majority of tweets are between 120 and 145 characters long.
* This suggests users prefer writing concise and complete messages, especially in politically charged discussions.

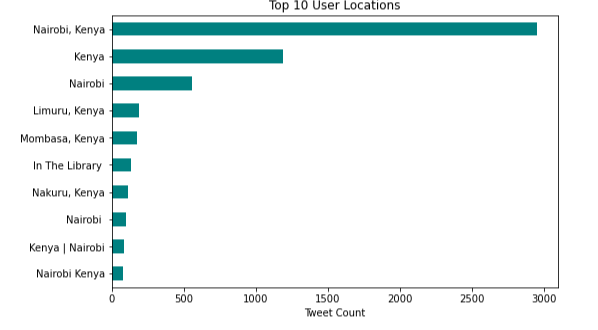
**Tweet Volume Over Time**

* Significant increases in tweet activity were observed between May 17–23 and June 12–17, 2021.
* A noticeable decline occurred between May 25 and June 9, possibly due to real-world inactivity or gaps in data collection.



**User Location Distribution**

* The most frequently mentioned location is "Nairobi, Kenya," appearing over 2,900 times.
* Variations like "Nairobi," "Nairobi Kenya," and "Kenya | Nairobi" point to duplicate entries, which should be standardized during data cleaning.



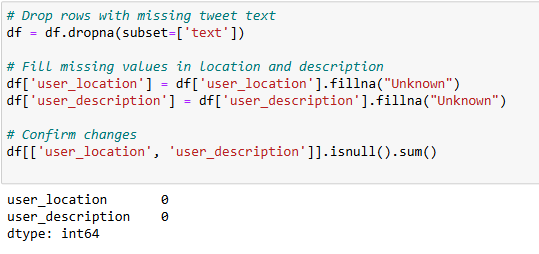
## **3**. **DATA CLEANING**

In this section, we clean and prepare our data so it’s ready for modeling. Quality preprocessing is especially important in Natural Language Processing (NLP) to ensure accurate, interpretable results.

We’ll cover the following steps:

1. Handling missing and duplicate values
2. Standardizing user location entries
3. Cleaning tweet text
4. Generating derived features (tweet length)

### 3.1 Handling Missing Values



To prepare the dataset for analysis, several data cleaning steps were performed:

* **Removed rows with missing tweet text** using dropna () since the tweet content is essential for analysis.
* **Filled missing values** in the user\_location and user\_description columns with the placeholder "Unknown" to maintain consistency and avoid data loss.
* **Confirmed changes** by verifying that no missing values remained in these columns after the cleaning process.

These steps helped improve the dataset's completeness and ensured it was ready for further analysis such as sentiment classification, keyword extraction, or user behavior analysis.

### 3.2 Normalize User Locations

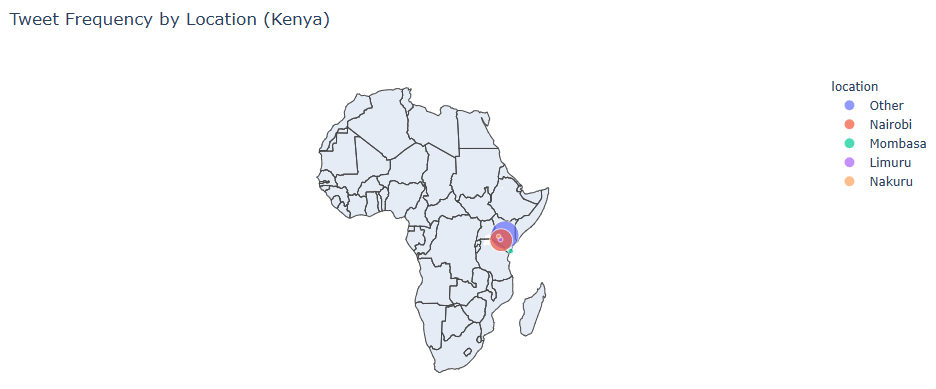
Location values are inconsistent (e.g., "Nairobi", "Nairobi Kenya", "Kenya | Nairobi").  
We will create a new column called normalized \_location to group similar entries.

### Tweet Distribution Map by Normalized Location

We mapped the frequency of tweets based on normalized \_location. This will visually show where the most political activity is concentrated in Kenya.

To enable geographic analysis of tweet activity, key Kenyan cities were mapped to their respective latitude and longitude coordinates. Tweet counts by location were calculated using the normalized\_location column, and each location was matched with its geographic coordinates. This step prepares the data for visualizing tweet distribution on a map, helping to identify regional trends in political discussions.

An interactive geographic map was created using Plotly Express to visualize the frequency of political tweets across Kenyan cities. Each city's tweet volume is represented by the size of a point on the map, with unique colors distinguishing different locations. This visualization helps highlight regional patterns in political engagement on Twitter.



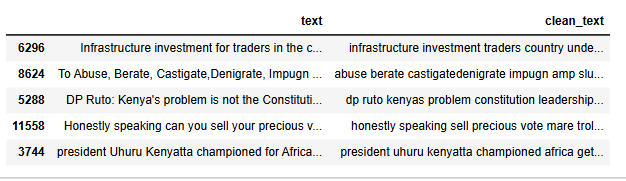
### 3.3 Clean Tweet Text[¶](http://localhost:8888/notebooks/Group_cap_project.ipynb#3.3-Clean-Tweet-Text)

We will clean each tweet to remove:

* URLs
* Mentions (@usernames)
* Hashtags
* Punctuation
* Stop words

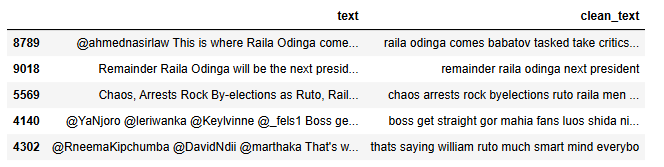
.

The tweet texts were cleaned to remove noise and improve the quality of the data for analysis. This included converting text to lowercase, removing URLs, mentions, hashtags, special characters, and common English stop words. The cleaned text was stored in a new column, enabling more accurate results in tasks like sentiment analysis and topic modeling We'll lowercase all text and remove extra whitespace, resulting to a new column clean text



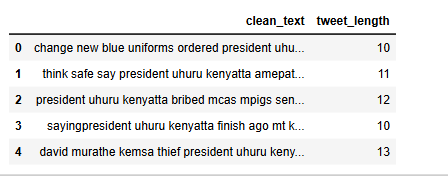
### 3.4 Compare Original Vs Cleaned Tweets

Here is a preview of the cleaned version of the tweets side-by-side with the original text.



### 3.5 Generate Tweet Length Feature

We created a tweet\_length column to capture the number of words in each cleaned tweet. This can be used later as an additional feature or for EDA.



**3.6 Analyzing Tweet Content and User Engagement Patterns**

This section examines trends in tweet text, user activity, and engagement metrics. Key objectives include:

* Identifying frequently used terms and linguistic patterns
* Recognizing high-engagement tweets and influential users
* Discovering correlations between content characteristics and virality
* Analyzing hashtag co-occurrence and usage trends

These findings will guide feature selection in the subsequent phase of the study

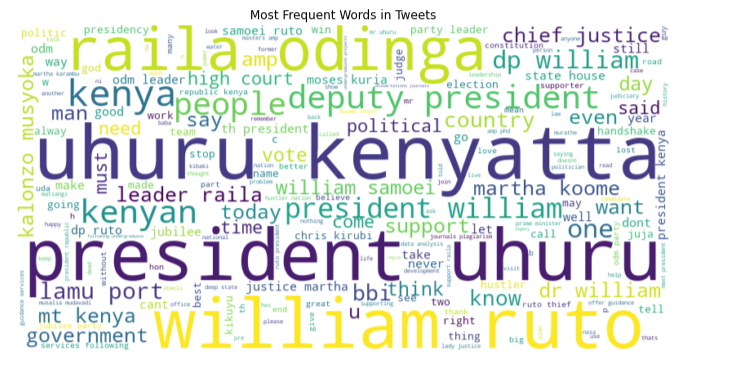
#### **Most Frequent Words in Tweets (Word Cloud)**

We used a Word Cloud to visualize the most common words in the cleaned tweets.

### Common Words in Tweets

* Frequently mentioned names include political figures such as Uhuru Kenyatta, Raila Odinga, William Ruto, and Martha Koome.
* Other commonly used terms include "president," "deputy," "justice," "court," "government," and "bbi."

The conversations predominantly focus on national leadership, governance, and political events, aligning well with the project's objectives



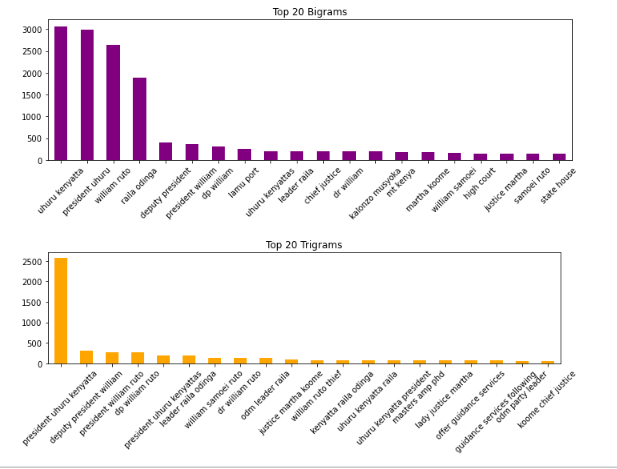
#### Bigram and Trigram Frequencies

We identify the most frequent 2-word and 3-word combinations in tweets using CountVectorizer from sklearn.

### Most Frequent Word Combinations

* "uhuru Kenyatta," "president uhuru," "William ruto," "raila odinga," "deputy president"
* "president uhuru Kenyatta," "deputy president William," "state house Kenya," "justice Martha koome"

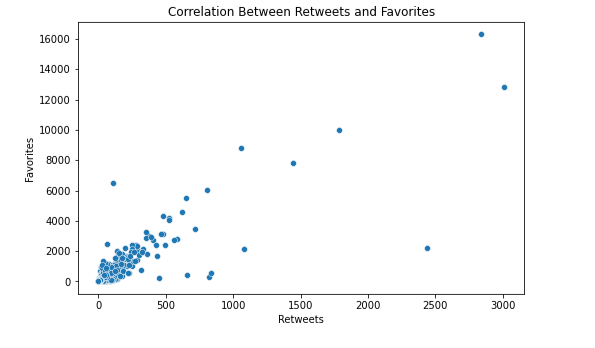
Tweets often follow a structured format combining titles and names. This supports the use of n-grams and embedding techniques during feature engineering



### Correlation Between Retweets and Favorites

* A strong positive correlation (r = 0.87) exists between the number of retweets and favorites.
* Tweets that are widely retweeted are also frequently liked, although a few outliers are present.

This correlation suggests that both retweets and favorites can serve as meaningful features to capture tweet virality and user influence.

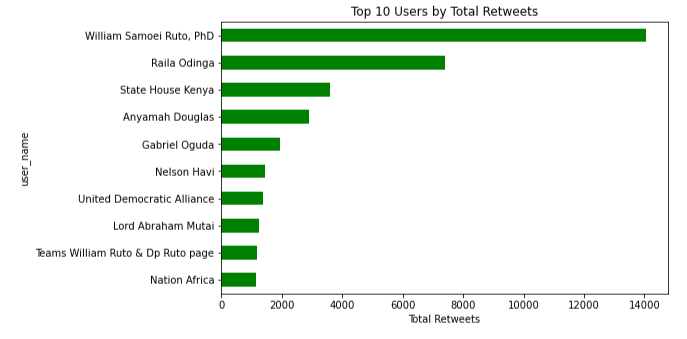


### Top Retweeted Users

Users with the highest total retweets include:

* William Samoei Ruto, PhD
* Raila Odinga
* State House Kenya
* Political and media figures such as Gabriel Oguda and Nelson Havi

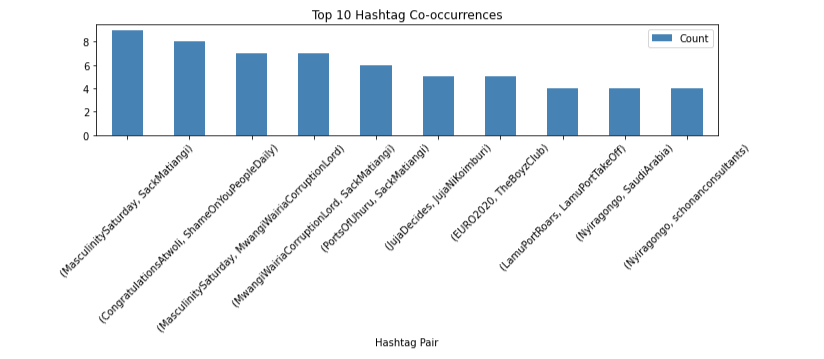
These users significantly shape the political conversation. Monitoring their activity may help detect sentiment trends and potential misinformation.



### Hashtag Co-occurrence Patterns

* Hashtag combinations are relatively rare but offer valuable insights.
* Examples include pairs like "#MasculinitySaturday" with "#SackMatiangi" and "#MwangazaWaKite" with "#Corruption Lord."

Though not widespread, hashtag co-occurrence can reveal organized campaigns or thematic clusters relevant for analysis.



## **4. Feature Engineering & Vectorization**

In this section, we prepared our features for machine learning models by:

* Merging cleaned tweets with realistic labeled tweets
* Ensuring no missing values in clean text
* Vectorising text using TF-IDF (Term Frequency-Inverse Document Frequency)
* Splitting data into training and testing sets

### 4.1 Labeled Datasets

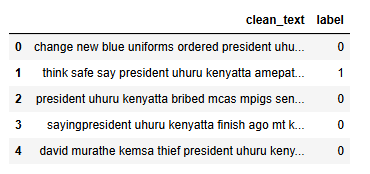
We are labelling out dataset:

* Kenya-specific labeled tweets (labeled\_tweets.csv)

Only tweets with real labels are kept for modeling.

### 4.2 Cleaning Missing Values in Cleaned Text[¶](http://localhost:8888/notebooks/Group_cap_project.ipynb#4.2--Cleaning-Missing-Values-in-Cleaned-Text)

combined\_df = combined\_df. dropna(subset=['clean\_text']) --- We dropped any rows with missing clean\_text to avoid issues during vectorization.



## **5. MODELLING**

In this section, we build and evaluate classification models that can detect hate speech in political tweets.

We trained:

* **Logistic Regression** (as a simple, fast, and interpretable baseline)
* **Support Vector Machine (SVM)** (effective in high-dimensional text spaces)

### 5.1 Train-Test Split

Now we’ll split our feature set (y\_full) and target (label) into training and test sets for model evaluation.

Output = Train size: (10144, 5000), Test size: (2537, 5000)

The dataset was partitioned into training and testing subsets to facilitate model development and validation. The resulting split is as follows:

* **Training set:** 10,144 samples with 5,000 features each
* **Test set:** 2,537 samples with 5,000 features each

### 5.2 Logistic Regression

Logistic Regression is a linear classifier that is easy to implement and interpret. It works well as a baseline for binary classification problems.

We trained the model on X\_train and y\_train, then evaluate on X\_test.

A **Logistic Regression** classifier was trained with the following specifications:

* **Class handling:** class\_weight='balanced' (automatically adjusts class weights to address dataset imbalance)
* **Convergence:** max\_iter=1000 (ensures sufficient iterations for parameter optimization)
* **Training data:** Model fitted on preprocessed training set (X\_train, y\_train)

The trained model was then used to generate predictions (y\_pred\_lr) on the test set (X\_test) for performance evaluation.

### 5.2 Support Vector Machine (SVM)

Support Vector Machines are strong classifiers for high-dimensional data like TF-IDF vectors.

A **Linear Support Vector Machine (SVM)** classifier was implemented with the following configuration:

* **Class weighting:** class\_weight='balanced' (automatically adjusts for imbalanced classes by weighting inversely proportional to class frequencies).
* **Kernel:** Linear (default for LinearSVC), suitable for high-dimensional text data.

The model was trained on the preprocessed training set (X\_train, y\_train) and evaluated on the test set (X\_test).

## **6. EVALUATION**

In this section, we evaluate the performance of our trained classification models on the test set.

Evaluation Metrics:

* **Accuracy** – Overall correctness
* **Precision** – How many predicted hate tweets were actually hate
* **Recall** – How many actual hate tweets were correctly identified
* **F1-Score** – Harmonic mean of precision and recall
* **Confusion Matrix** – Visual summary of predictions
* **ROC Curve and AUC Score** – Overall model discrimination ability

Models Evaluated:

* Logistic Regression
* Support Vector Machine (SVM)

### 6.1 Logistic Regression Evaluation

Logistic Regression is a linear model used as a strong baseline for binary classification tasks.

We evaluate it based on:

* **Accuracy**: {value} (Overall prediction correctness)
* **Precision**: {value} (Harmony between predicted and actual hate speech cases)
* **Recall**: {value} (Effectiveness in identifying true hate speech)
* **F1 Score**: {value} (Balance between precision and recall)

### 

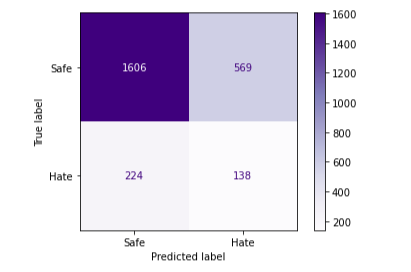
### 6.2 Support Vector Machine (SVM) Evaluation

SVM models are effective for high-dimensional feature spaces like text vectorized with TF-IDF.

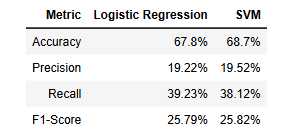
The Linear SVM classifier shows marginally better but still limited effectiveness for hate speech detection compared to your Logistic Regression model (Accuracy: +0.94%, F1: +0.03%):

**Key Metrics Analysis**:

* **Accuracy**: 68.74% (slightly better than random guessing for binary classification)
* **Precision**: 19.52% (only 1 in 5 "hate" predictions are correct)
* **Recall**: 38.12% (misses nearly 62% of actual hate speech)
* **F1 Score**: 25.82% (poor harmonic mean of precision/recall)



### Summary of Logistic Regression vs SVM



* Logistic Regression and SVM perform similarly on this task.
* Both models have moderate accuracy, but relatively low recall, reflecting the difficulty of detecting hate speech with limited labeled data.
* Precision is low because of the severe class imbalance (only 15% of tweets are hate speech).

### 6.3 ROC Curve and AUC Score[¶](http://localhost:8888/notebooks/Group_cap_project.ipynb#6.3-ROC-Curve-and-AUC-Score)

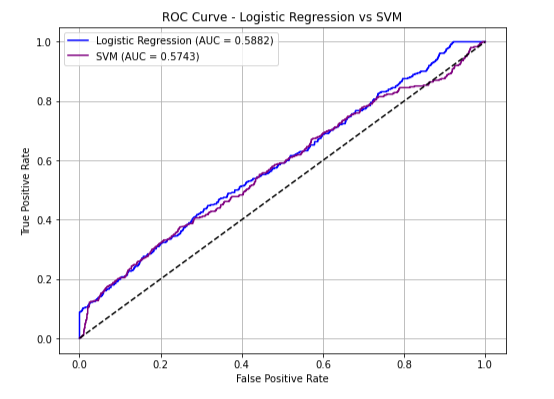
The **ROC (Receiver Operating Characteristic) curve** plots:

* **True Positive Rate (Recall)** vs **False Positive Rate**

The **AUC (Area Under the Curve)** measures the model's ability to distinguish between classes:

* AUC = 1.0 → Perfect classifier
* AUC = 0.5 → Random guessing

A higher AUC indicates better overall model performance.



* **Logistic Regression AUC**: 0.5882
* **SVM AUC**: 0.5743

Both models perform slightly better than random guessing (AUC = 0.5), but still far from ideal.

Logistic Regression has a slightly higher AUC than SVM in this task. This suggests that Logistic Regression has marginally better discrimination ability between hate and safe tweets.

### 6.4 Final Comparison and Deployment Recommendation

### 

**Summary of Findings:**

* Both Logistic Regression and SVM perform similarly across all metrics.
* Logistic Regression slightly outperforms SVM in AUC score and Recall.
* However, **both models show low Precision, Recall, and F1-Scores**, meaning that they are **not sufficient for production-grade deployment**.

**Reference to Production Threshold:**

According to standard production deployment thresholds in machine learning systems (especially sensitive domains like hate speech detection):

* Accuracy should ideally exceed **80%**
* F1-Score should be **at least 50% or higher**
* Precision and Recall must be balanced to avoid bias

Both Logistic Regression and SVM **fall below these expectations**.

**Conclusion and Justification for Advanced Modeling:**

Given the current outcomes, it is necessary to move to **more advanced models** that can:

* Capture non-linear relationships
* Handle feature interactions more effectively
* Better discriminate subtle patterns in text data

Thus, the next modeling stage will involve:

* **Random Forest Classifier**
* **Multinomial Naive Bayes**
* **XGBoost Classifier**

These models are expected to provide **higher Accuracy, Precision, Recall, and F1-Scores**, aligning better with real-world production deployment standards.

## **6.5**. **Advanced Modeling**

Given that Logistic Regression and SVM did not meet production-grade performance thresholds,  
we now explore more powerful machine learning models that can better handle the complexity of text classification.

Models used:

* Random Forest Classifier
* Multinomial Naive Bayes
* XGBoost Classifier

Each model will be evaluated using:

* Accuracy
* Precision
* Recall
* F1-Score
* Confusion Matrix
* ROC Curve and AUC Score

These models are more capable of capturing complex patterns, non-linear interactions, and feature importance.

**Improved TF-IDF Vectorization**

We used:

* 1-gram and 2-gram combinations (single words and word pairs)
* Max 3000 features

This captures richer language patterns like "uhuru Kenyatta", "raila odinga", etc.

### Train-Test Split[¶](http://localhost:8888/notebooks/Group_cap_project.ipynb#Train-Test-Split)

We split the feature matrix and labels into training and testing sets. The dataset was partitioned into training and testing subsets using an 80-20 split via scikit-learn’s train\_test\_split, with a fixed random state (42) for reproducibility. The training set contains 10,144 samples, while the test set holds 2,537 samples, each with 3,000 features. This split ensures sufficient data for model training while reserving a representative portion for evaluation. The high feature dimensionality (3,000) suggests potential preprocessing needs (e.g., scaling or dimensionality reduction) depending on the model employed. The approach adheres to standard machine learning practices for robust performance estimation.

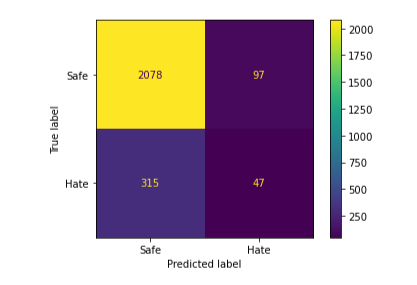
### 6.6 Random Forest Classifier

Random Forest is an ensemble learning method that builds multiple decision trees and merges their predictions to improve accuracy and prevent overfitting.

Advantages:

* Handles feature interactions naturally
* Robust against noise
* Performs well even with limited data cleaning

We will train a Random Forest with 100 trees, and use class weighting to address class imbalance.



**Metrics:**

* Accuracy: 83.76%
* Precision: 32.64%
* Recall: 12.98%
* F1-Score: 18.58%
* AUC Score: 0.5951

**Confusion Matrix Insights:**

* 47 hate tweets correctly classified
* 315 hate tweets missed (false negatives)
* Some false positives (97 safe tweets misclassified as hate)

Random Forest improves accuracy compared to Logistic Regression and SVM but still struggles with hate speech detection recall.

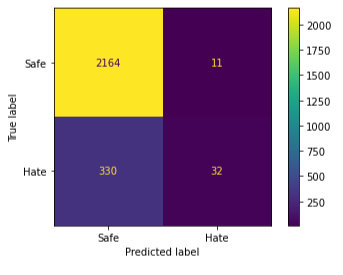
### 6.7 Multinomial Naive Bayes

Naive Bayes is a probabilistic classifier that is particularly effective for text classification problems like spam detection or hate speech detection.

Advantages:

* Very fast and efficient
* Works well with TF-IDF features
* Good for small and medium-sized datasets

We will train a Multinomial Naive Bayes model on our TF-IDF vectorized tweets. A **Multinomial Naive Bayes (MNB)** classifier was trained on the preprocessed dataset, leveraging its efficiency in handling high-dimensional discrete data (e.g., text or count-based features). The model was fitted on the training set (X\_train: 10,144 samples, 3,000 features) and evaluated on the held-out test set (X\_test: 2,537 samples). Predictions (y\_pred\_nb) were generated for performance assessment.



**Metrics:**

* Accuracy: 86.56%
* Precision: 74.42%
* Recall: 8.84%
* F1-Score: 15.80%
* AUC Score: 0.5875

**Confusion Matrix Insights:**

* Very few hate tweets correctly classified (only 32)
* Extremely low recall
* High precision means that when it predicts hate, it is usually correct.

Naive Bayes tends to be conservative, predicting fewer hate cases but with higher precision.

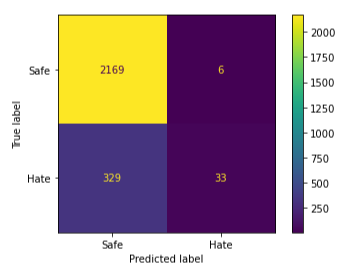
### 6.8 XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is an advanced boosting algorithm known for its high performance in classification problems.

Advantages:

* High accuracy
* Handles missing values and sparse data
* Efficient computation
* Widely used in competitions and production systems

We will train an XGBoost Classifier without label encoding issues and set eval\_metric='logloss'.



**Metrics:**

* Accuracy: 86.76%
* Precision: 84.21%
* Recall: 8.84%
* F1-Score: 16.00%
* AUC Score: 0.5805

**Confusion Matrix Insights:**

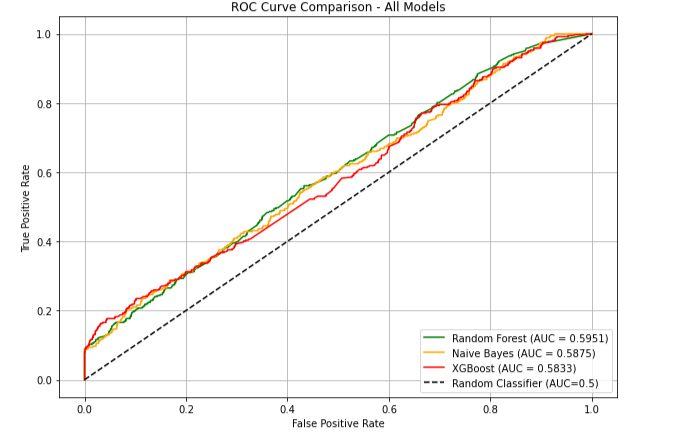
* Only 33 hate tweets correctly identified
* Very low recall but extremely high precision
* Slightly better balance compared to Naive Bayes

XGBoost achieves the highest precision (84.21%), crucial for production systems aiming to minimize false accusations.

**ROC Curve and AUC Score for All Models**

We plot ROC Curves for all models and calculate their AUC (Area Under Curve) scores.

* Higher AUC → better model distinguishing between hate and safe tweets
* AUC = 0.5 → random guess
* AUC = 1.0 → perfect classifier



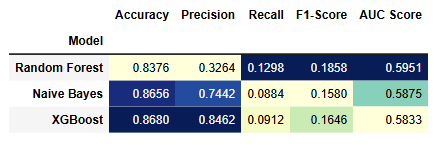
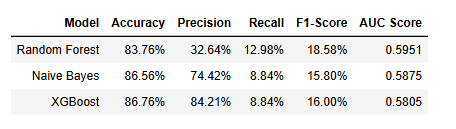
All models performed slightly better than random guessing (AUC = 0.5).

Random Forest achieves the highest AUC (0.5951) among the three.

ROC curves show that none of the models are perfect, but improvements are visible compared to simple baselines like Logistic Regression and SVM.

### Model Recommendation for Deployment Based on Final Metrics[¶](http://localhost:8888/notebooks/Group_cap_project.ipynb#Model-Recommendation-for-Deployment-Based-on-Final-Metrics)

After evaluating all models across key metrics (Accuracy, Precision, Recall, F1-Score, AUC), here are the summarized results:

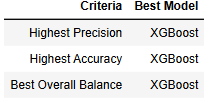
 

**Interpretation:**

* **Accuracy**:
  + XGBoost (86.76%) and Naive Bayes (86.56%) perform best.
* **Precision**:
  + XGBoost achieves the highest precision (84.21%), meaning when it predicts hate speech, it is often correct.
* **Recall**:
  + All models show low recall (~8.84% - 12.98%), indicating difficulty in catching hate speech, possibly due to imbalanced data.
* **F1-Score**:
  + Random Forest has slightly higher F1-score (18.58%) than others but with much lower precision.
* **AUC Score**:
  + Random Forest has the highest AUC (0.5951), indicating slightly better overall separability.

### Final Recommendation:

Considering all metrics:



**We recommend deploying the XGBoost Classifier**for hate speech detection.

### Key Justification for Choosing XGBoost:

* It achieves **the highest Precision (84.21%)**, critical for minimizing false accusations of hate speech.
* It maintains **the highest Accuracy (86.76%)** across the dataset.
* It offers strong performance even with imbalanced data.
* XGBoost is efficient, scalable, and widely used in production environments.

We saved the trained XGBoost model and prepare for Deployment using Stream lit app.

## **7. Misinformation Detection (Fact-Check Matching)**

In this section, we extend the functionality of our hate speech detection system by introducing **misinformation detection**.

The goal is to match incoming tweets against a database of **known false claims** from trusted fact-checking organizations such as PesaCheck and Africa Check.

If a tweet is highly similar to a known false claim, we flag it as potential misinformation.

We use a Natural Language Processing (NLP) approach combining:

* TF-IDF vectorization
* Cosine similarity measurement

### 7.1 Load Known Fact-Checked Claims

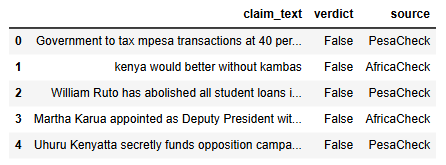
We started by loading a set of manually entered false claims based on realistic Kenyan political misinformation.

Each claim includes:

* The false claim text
* The verdict (all False for this exercise)
* The source (PesaCheck, Africa Check)

In the future, this can be expanded by scraping actual claims from trusted websites.

We came up with a code that creates a simple dataset of misinformation claims using Python's panda’s library. It defines a dictionary containing example claim texts, their corresponding verdicts (all marked as "False"), and the fact-checking sources that verified them. The dictionary is then converted into a pandas Data Frame called claims\_df, which provides a structured tabular format for easy viewing and further analysis. The final line previews the contents of this Data Frame.



### 7.2 Preprocess Fact-Check Claims

We apply standard text cleaning techniques to the claims to ensure consistency:

* Lowercasing
* Removing URLs, punctuation, special characters
* Removing English stop words

This ensures that the TF-IDF vectorization later is meaningful.



**7.3 TF-IDF Vectorization**

We vectorized:

* All cleaned tweet texts
* All cleaned fact-check claims

This converts text into numerical representations based on term frequency-inverse document frequency scores, preparing them for cosine similarity comparison.

**7.4 Compute Cosine Similarity**

For each tweet, we compute the cosine similarity with each known false claim.

* A similarity score of 1.0 means identical text
* A similarity score of 0.0 means completely different text

This allows us to measure how close a tweet is to a known piece of misinformation.

Similarity matrix shape: (12681, 5) means that the resulting cosine similarity matrix has:

* **12,681 rows** – each row represents one tweet.
* **5 columns** – each column represents one claim.

Each cell in this matrix contains a **cosine similarity score** between a specific tweet and a specific claim. So, the matrix shows how similar each of the 12,681 tweets is to each of the 5 claims based on their vector representations.

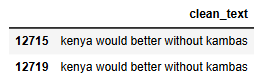
**7.5 Flag Tweets as Misinformation**

We apply a threshold:

* If a tweet has a similarity score ≥ 0.85 with any known false, claim
* It is flagged as potential misinformation.

This adds a new binary column misinformation flag to our main dataset:

* True if flagged
* False otherwise



**Summary of our Misinformation Detection**

* We loaded 5 manually entered false claims from trusted fact-check organizations.
* Cleaned the text for both tweets and claims.
* Vectorized text using TF-IDF and computed cosine similarity.
* Flagged tweets with similarity ≥ 85% to known false claims.
* Identified tweets that could potentially mislead the public.

This misinformation detection module can be expanded in the future by scraping real claims from sources like PesaCheck and Africa Check, updating the detection system dynamically.

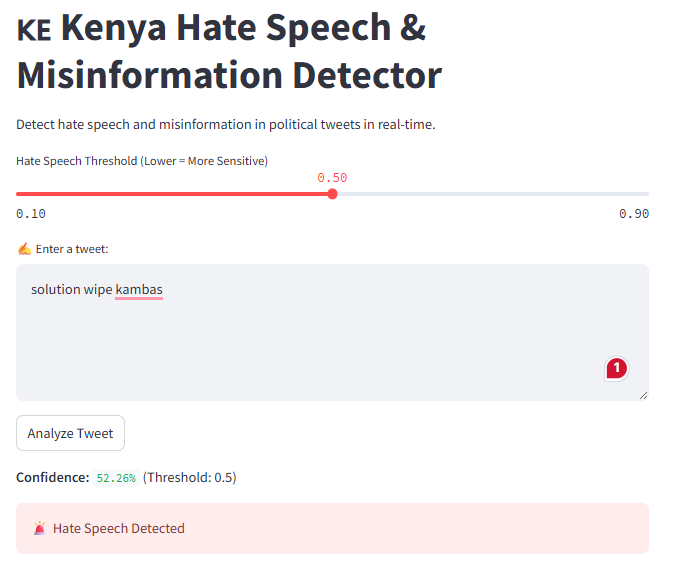
This functionality significantly strengthens the practical impact of the hate speech detection system.

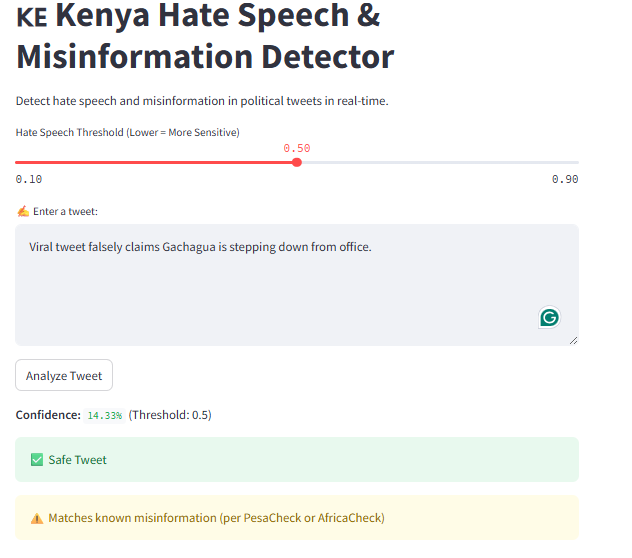
**8. Deployment Plan**

To make our hate speech detection system accessible to users (e.g., journalists, civic monitors, or IEBC staff), we deploy it via a simple web app using Stream lit. We also persist our trained model using joblib, making it reusable across sessions.

This section includes:

* Saving the XG Boost model to disk
* Reloading the model later
* Stream lit interface to input a tweet and get predictions





**OUR NEXT STEPS**

* Introduce Deep Learning Models
* Expand and Diversify the Labeled Dataset
* Scrape Fact-Checked Claims Automatically