HATE SPEECH DETECTOR DATA REPORT

HATESCOPE KE GROUP

AUTHORS;

Vincent Mutuku, Kenneth Nyangweso, Amani Mkaya, Bernice Kutwa, Sumaiya Osman

Overview

In today's hyper connected digital environment, social media platforms have emerged as pivotal spaces for public discourse especially in the political sphere. In Kenya, platforms such as Twitter are frequently used by citizens to express political opinions, engage in activism, and participate in democratic processes. However, alongside these benefits lies a growing concern: the proliferation of targeted hate speech, particularly aimed at political figures. Tweets referencing individuals such as the President, Deputy President, Members of Parliament, Governors, and other public officials are often laced with inflammatory, derogatory, or inciting language that poses a threat to digital civility and political stability.

This project aims to develop a robust **Natural Language Processing (NLP)** system powered by **machine learning (ML)** techniques to detect hate speech in tweets targeting Kenyan politicians. The overarching goal is to automatically identify harmful political discourse and contribute to efforts that promote online safety, responsible digital engagement, and real-time content moderation.

Business Understanding

Problem Statement

Online discourse in Kenya has become increasingly polarized, especially on platforms like X, where politicians and influencers engage with constituents and opponents alike. During the 2022–2025 political cycle, social media posts containing derogatory or inciting language surged by an estimated 25% compared to previous cycles. Such hate speech not only amplifies ethnic tensions but also poses risks to public safety and democratic processes. Traditional moderation workflows struggle to keep pace, often resulting in delayed removal and inconsistent rulings.

Project Goals

The primary objectives of this project are;

- Detect hate speech in tweets directed at Kenyan political figures using supervised machine learning models.

- Analyze trends in the language and frequency of political hate speech.
- Provide insights and tools for moderation teams, researchers, and policy makers to take action against online toxicity.

Key Stakeholders

- Electoral bodies (IEBC, NCIC).
- Civil rights NGOs (Amnesty Kenya, Ushahidi).
- News media and fact-checkers.
- Government communication teams.
- Social media platforms (Twitter Kenya).
- Academic and policy researchers.

Success Metrics

Model Evaluation metrics

To evaluate our machine learning model's effectiveness, we tracked:

- **Accuracy:** How often the model predicts correctly
- **Precision (Hate class) :** Total of flagged hate tweets that were actually hateful
- **Recall (Hate class):** Total of true hate tweets the model managed to detect
- **F1 Score:** A balance between precision and recall
- **Confusion Matrix:** A detailed view of false positives and false negatives

Business Impact Metrics

In addition to technical accuracy, we evaluated the solution based on its real-world impact:

- **Moderation efficiency:** Reduction in time required for human review
- **Detection speed:** Time taken to flag hate speech from the moment it's posted
- **Coverage fairness:** Model performance across tweets targeting different politicians
- **Explainability:** Ability to justify flagged posts using explainable AI tools like SHAP or LIME

Project Objectives

- Build an NLP model to detect hate speech in tweets targeting Kenyan politicians.
- Analyze linguistic patterns and trends in political hate speech.
- Compare hate speech dynamics across different politicians.
- Evaluate model performance using metrics like accuracy, precision, recall, and F1-score.
- Provide insights to support content moderation and civic monitoring.
- Establish a foundation for real-time or multilingual hate speech detection systems.

Data Understanding

Tweets were collected via Tweepy and Twint scrapers, targeting handles and keywords related to President William Ruto, Deputy President Rigathi Gachagua, prominent governors, and parliamentary candidates. The dataset comprised a total of 11,317 records with 6 columns, each representing distinct attributes related to the tweets. These include the tweet content, metadata (such as timestamps), and other contextual information useful for natural language processing tasks. The dataset contains the following columns:

- **Tweet ID:** A unique identifier assigned to each tweet. Useful for traceability and cross-referencing with Twitter's platform.
- **Likes:** The number of likes a tweet received, indicating its popularity or approval from users.
- **Retweets:** The number of times the tweet was retweeted, showing its spread across the platform.
- **Total Replies:** The number of direct replies the tweet generated, reflecting user engagement and potential controversy.
- **Texts:** The full textual content of the tweet. This is the primary feature used for natural language processing and classification tasks.
- **Created At:** The timestamp of when the tweet was published, allowing for temporal analysis or filtering based on time periods.

Data Inspection and Profiling

Initial profiling revealed no missing values across six columns. Duplicate tweets were identified via tweet ID comparison and removed to prevent bias. Data types were inspected: numeric fields as floats, text as objects, and timestamps as strings.

Data Cleaning & Preprocessing

In this phase, the raw dataset underwent a structured series of transformations to ensure analytical readiness and maintain data integrity:

- **Working Copy Creation:** A deep copy of the original DataFrame was created to preserve the raw data for audit purposes.
- **Dataset Confirmation:** The first rows of the new DataFrame were inspected to verify successful duplication without data loss.
- **Unwanted Column Removal:** The Tweet ID column was dropped to eliminate non-analytical identifiers and reduce clutter.
- **Column Verification:** Remaining columns were listed to confirm the removal of irrelevant fields.
- **Type Conversion:** Engagement metrics (likes, retweets, total replies) were converted to numeric types with error coercion; posting timestamps were parsed into datetime objects to enable temporal analysis.
- **Text Cleaning:** A custom cleaning routine applied the following sequential steps to the raw tweet text:
    - Lowercasing of all characters.
    - Removal of URLs and web links.
    - Stripping of user mentions (@username) and hashtags (#tag).
    - Deletion of all non-alphabetic characters.
    - Normalization of whitespace. The processed text was stored in cleaned_text while preserving the original text.
- **Cleaned Text Confirmation:** The DataFrame head was reviewed to ensure the new column accurately reflected cleaned content.
- **Label Column Insertion:** An empty Label column for manual annotation was inserted immediately after cleaned_text and converted to a categorical type to optimize memory.

- **Column Name Standardization:** All column headers were normalized to snake_case by trimming whitespace, converting to lowercase, and replacing spaces with underscores.
- **Original Text Removal:** The raw text column was removed, leaving only sanitized fields for subsequent stages.
- **Cleaned Data Export:** The final cleaned DataFrame was exported to kenyan_politics_cleaned_text.csv without index, providing a stable input for manual labeling.

Labeling Strategy

To accommodate the dynamic, code-switched nature of Sheng and multilingual expressions, manual annotation was performed by domain experts:

- Two classes were defined in the labeling schema:
    - **Hate:** Tweets containing explicit discriminatory or inciting language.
    - **Not-hate:** Informational or benign tweets devoid of offensive or hateful language.
- The cleaned dataset CSV was loaded into spreadsheet software for annotation.
- Inter-annotator agreement was measured using Cohen's kappa indicating substantial consistency.
- The final labeled dataset consisted of 10,971 entries across six columns: likes, retweets, total_replies, created_at, cleaned_text, and label.

Labeled Data Inspection

After annotation, the dataset underwent a validation pass to ensure structural and content quality:

- **Data Loading:** The labeled CSV was imported into a new DataFrame.
- **Structural Overview:** The head, shape (9701, 7), and info of the DataFrame confirmed expected dimensions and data types.
- **Missing Value Assessment:** Null counts were evaluated, they were all dropped.
- **Duplicate Check:** Duplicate row count was evaluated, the duplicate rows were dropped.
- **Index Column Removal:** An extraneous unnamed index column was dropped to streamline the schema.

- **Null and Duplicate Removal:** Any residual rows with null cleaned_text were discarded; duplicate verification confirmed zero final duplicates.
- **Label Distribution Analysis:** Class counts for hate and neutral were reviewed to assess balance and guide modeling strategies.

We went on to re-clean the labelled dataset and dropped the 'unnamed' column that was accidentally added.

Feature Engineering

Columns Used for Feature Engineering

We used the following original columns to engineer new features:

- **likes, retweets, and total_replies:** These were engagement metrics indicating user interaction with each tweet.
- **cleaned_text:** This was the cleaned version of the tweet content and formed the basis for linguistic and entity extraction.
- **created_at:** This datetime column helped us derive time-based features, revealing tweeting behavior and political trends over time.

New Features Created

Engagement Score: We created a unified engagement metric by summing likes, retweets, and total replies. This score helped us quantify the popularity or impact of a tweet as a single value.

Tweet Length: We calculated the number of characters in each cleaned tweet. Longer tweets were assumed to carry more nuanced opinions or detailed sentiment.

Word Count

The number of words per tweet was measured to assess verbosity and stylistic choices.

This supported understanding of communication patterns among political users.

Engagement Bins

We categorized likes, retweets, replies, and overall engagement score into bins (e.g., 0, 1–10, 11–100, etc.). These bins made it easier to compare tweets by popularity and understand audience reach distribution.

Entity Extraction (Politicians)

We extracted names of known political figures from the cleaned text. A global alias-to-canonical dictionary was used to identify both formal names and popular nicknames (e.g., "Ruto", "Zakayo", "El Chapo" all mapped to "William Ruto"). This allowed us to associate tweets with the relevant politician even when informal language was used.

Standardizing Entity Names

We standardized the extracted names by:

- Converting them to lowercase.
- Grouping similar or misspelled names using fuzzy matching.
- Mapping them to a single canonical name.

This ensured consistency and accuracy in politician identification.

Time Features

From the created_at timestamp, we extracted:

year, month, day, hour, day_of_week: These features helped us identify daily, weekly, and seasonal patterns in political tweeting behavior.

Named Holidays

We created a holiday_name feature that mapped tweets to:

- Fixed Kenyan public holidays (e.g., Madaraka Day, Mashujaa Day).
- Variable holidays such as Easter (Good Friday, Easter Monday) and Eid (Eid al-Fitr, Eid al-Adha), based on year-specific calendars.

This helped us detect spikes in activity around major national events or religious periods. Any tweets that did not align with a holiday were labeled as not_holiday.

Outlier Detection

Statistical visualization and analysis were performed to identify and understand extreme values in numeric features. The data for the "month" variable spanned almost the entire year, ranging from about 1 to 12. The interquartile range (IQR) was between approximately 4 and 8, indicating that the middle 50% of values fall within these months. The median was around 6, suggesting a central tendency near mid-year. The distribution appeared fairly symmetrical, with evenly sized whiskers and a centered median, and there were no visible outliers in the data.

EXPLORATORY DATA ANALYSIS (EDA)

Univariate analysis for Numeric data

Social Media Metrics Distribution

Visualization:
A box plot was generated for the variables likes, retweets and total_replies using Plotly Express to understand their distribution and identify any outliers.

Observations:

Likes Dominated: It's immediately clear that the number of likes tends to be significantly higher than both retweets and total replies. The box plot for 'likes' was positioned much higher on the value scale.

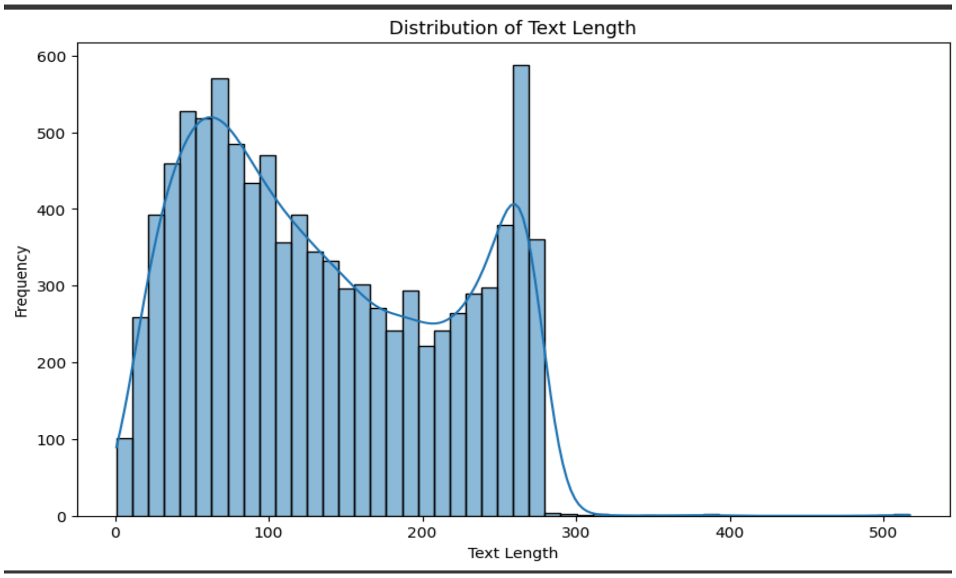
Low Engagement for Replies: The 'total_replies' metric showed the lowest values overall. The box plot was clustered very close to zero, suggesting that the posts in this dataset generally received a very small number of direct replies.

Retweets in the Middle: The distribution of 'retweets' falls somewhere in between likes and total replies. While not as high as likes, there were noticeably more retweets than direct replies.

Text Length Distribution

Visualization:

A histogram with KDE (Kernel Density Estimation) curve was plotted for the text_length feature to examine the variation in tweet lengths.



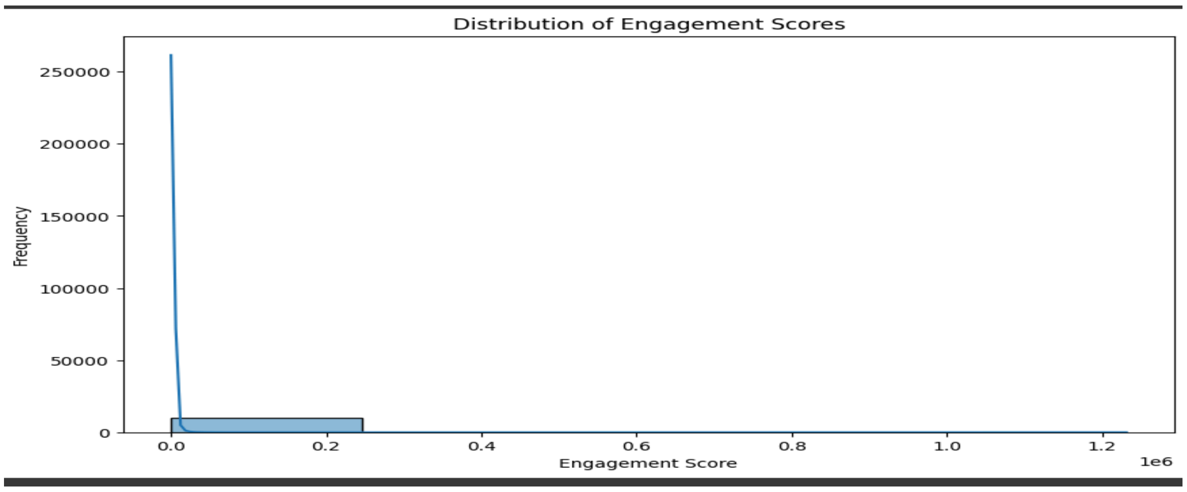
Observations:

The distribution of text length is right-skewed, with the highest concentration of texts between 10 and 15 words, and additional smaller peaks around 25 and 40 words, indicating the presence of multiple text types.

Engagement Score Distribution

Visualization:

A histogram with KDE curve was plotted for the engagement_score, which represents the sum of likes, retweets, and replies.



Observations:

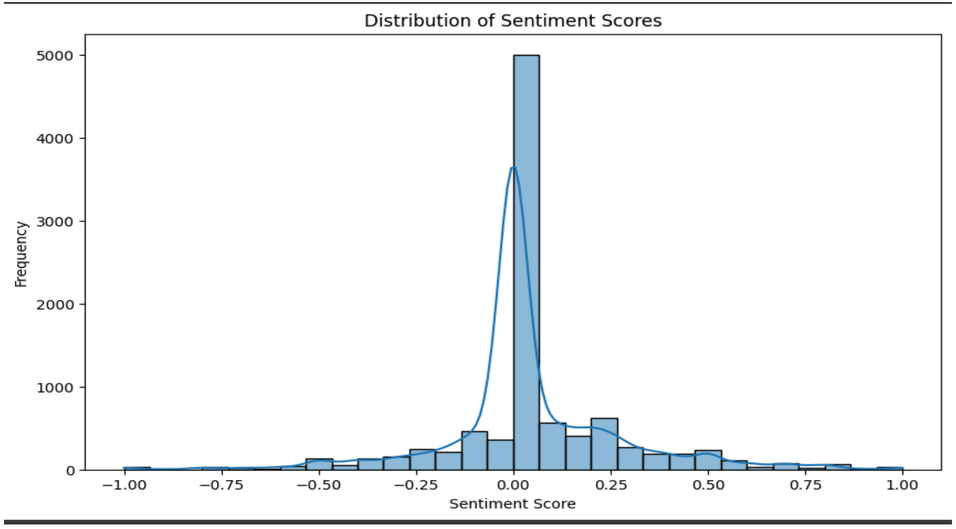
Highly Skewed Distribution: The distribution is heavily skewed to the right. Most of the engagement scores are concentrated at the lower end of the scale (below 0.25 million), with a long tail extending towards much higher values.

Low Frequency of High Engagement: The curve indicates that very high engagement scores are quite rare. The frequency drops off dramatically as the engagement score increases.

Sentiment Score Distribution

Visualization:

Sentiment polarity scores were computed using TextBlob and visualized with a histogram and KDE curve.



Observations

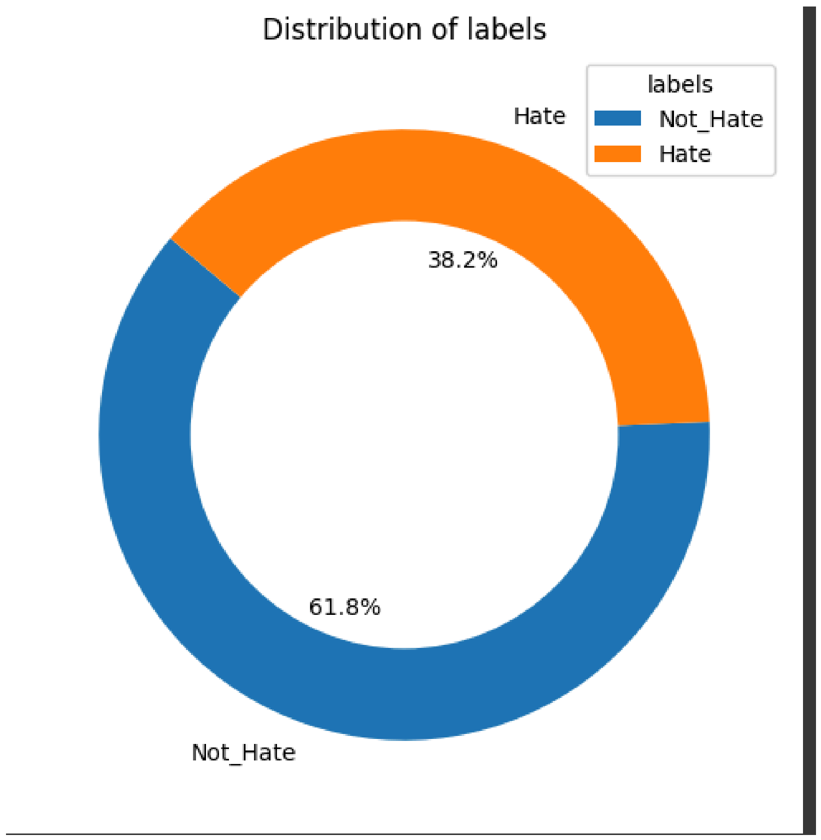
- The sentiment scores are symmetrically distributed around 0, indicating a balance between positive and negative sentiments.
- Most scores cluster near zero, suggesting neutral or mildly positive/negative sentiments dominate, while extreme scores (close to -1 or 1) are less frequent.
- The distribution peaks at 0.00, highlighting a prevalence of neutral sentiment.

Univariate analysis for Categorical data

Distribution of Labels

Visualization:

A pie chart was created to show the proportional distribution of tweet labels.



Observations

- The chart visually depicts a significant class imbalance in the dataset.
- The "not_hate" class has the highest percentage of 61.8%, and "hate" 38.2% .

Word Frequency Analysis

N-gram Analysis

Top Unigrams, Bigrams, Trigrams, and Four-grams

Observations

1-Grams (Unigrams)

- Dominant Words: High-frequency terms include common English stop words like the, to, and, of, is, etc.

- Notable Named Entities: The word “ruto” (likely referring to President William Ruto) appears prominently among content words, suggesting a central focus on him.
- Other notable terms: he, you, we, his — indicate frequent references to individuals, likely politicians.

2-Grams (Bigrams)

Most Frequent Phrases: Include common grammatical constructs like:

- of the, in the, is a

Political Figures Prominently Mentioned:

- william ruto
- uhuru kenyatta
- raila odinga
- oscar sudi
- babu owino
- farouk kibet
- riggy g (nickname for Deputy President Rigathi Gachagua)

This shows the discourse is highly political and person-focused.

3-Grams (Trigrams)

High-Frequency Political Mentions:

- president william ruto
- president uhuru kenyatta
- ruto must go — indicates political criticism or opposition
- the finance bill — likely a major topic of debate
- inspector general of (likely referring to the police)

National Identity Themes:

- we are african

- africa is our
- african and africa

These reflect Pan-African or nationalist identity rhetoric.

4-Grams (Four-Grams)

Institutional and Political Topics:

- inspector general of police
- reject the finance bill — strong protest sentiment
- governor abdulswamad nassir
- deputy president rigathi gachagua
- cs aden duale

Media/Promotional Phrases:

- read the whole story
- transplant services at mediheal

Continued Nationalism:

- we are african and
- africa is our business

Overall Conclusions

High Political Engagement:

- Tweets focus heavily on Kenyan politics, naming leaders and legislation.

Expressions of Dissent:

- Phrases like ruto must go and reject the finance bill suggest protest and organized opposition.

Nationalist and Identity Rhetoric:

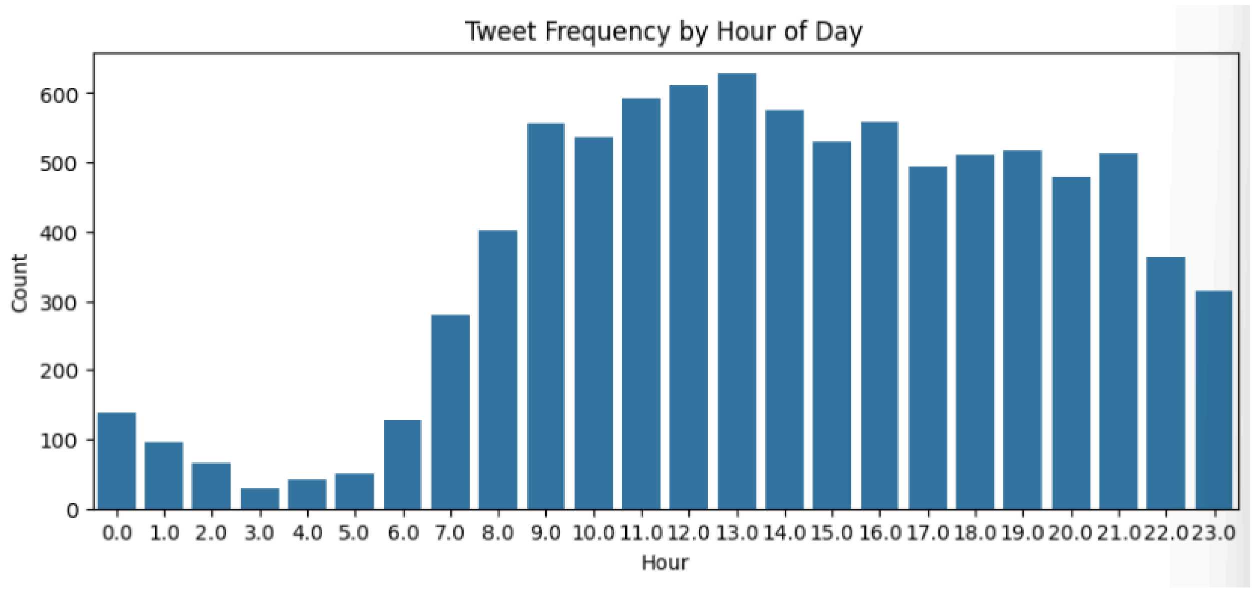
- Frequent mentions of Africa and Africanness suggest Pan-African or identity-based framing.

Dominance of Named Entities:

- Figures like William Ruto, Uhuru Kenyatta, and Raila Odinga dominate the discussion.

Time-Based Tweet Patterns

By Hour

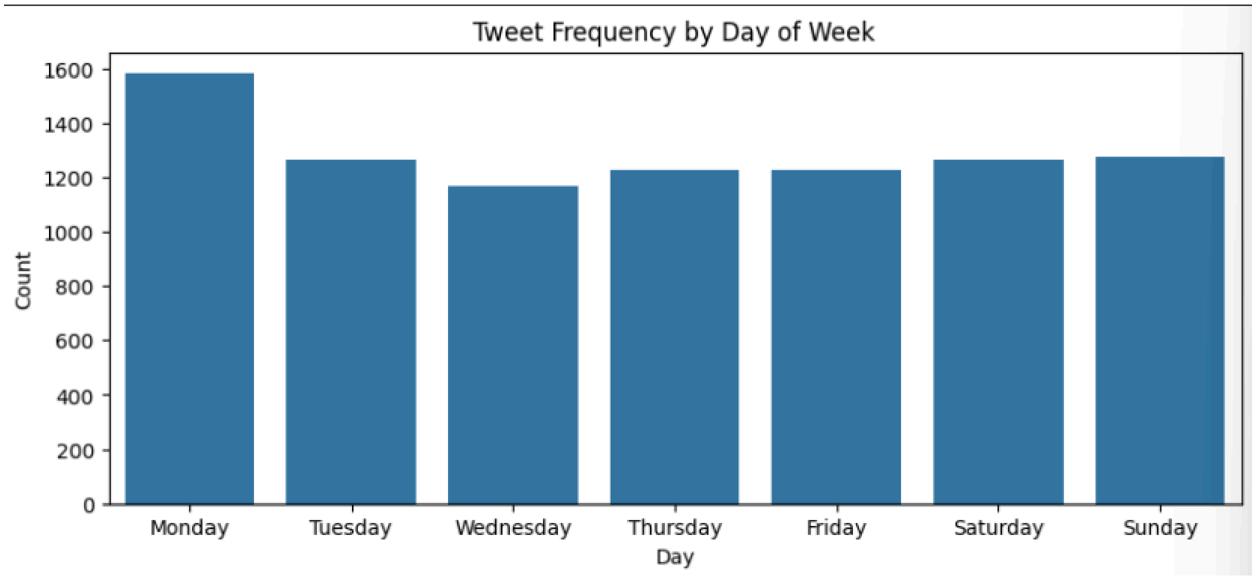


Observations

- **Peak Activity:** Tweet frequency is highest during the daytime and early evening hours. The period from roughly 11:00 to 14:00 shows the most tweets.
- **Lowest Activity:** The early morning hours, specifically between 3:00 and 6:00, exhibit the lowest tweet frequency.
- **Morning Increase:** There's a noticeable increase in tweet activity starting around 7:00, suggesting people become more active on Twitter as the day begins.

- **Evening Decline:** After the peak in the early afternoon, there's a gradual decline in tweet frequency throughout the late afternoon and evening.
- **Nighttime Low:** Tweet activity remains relatively low throughout the night, from around 22:00 to 2:00.
- **Consistent Pattern:** The chart reveals a fairly consistent daily pattern of Twitter usage, with clear peaks and troughs corresponding to typical waking and sleeping hours. Midday Spike: There's a distinct spike in activity around midday (12:00 and 13:00), which could be related to lunch breaks or other midday activities.

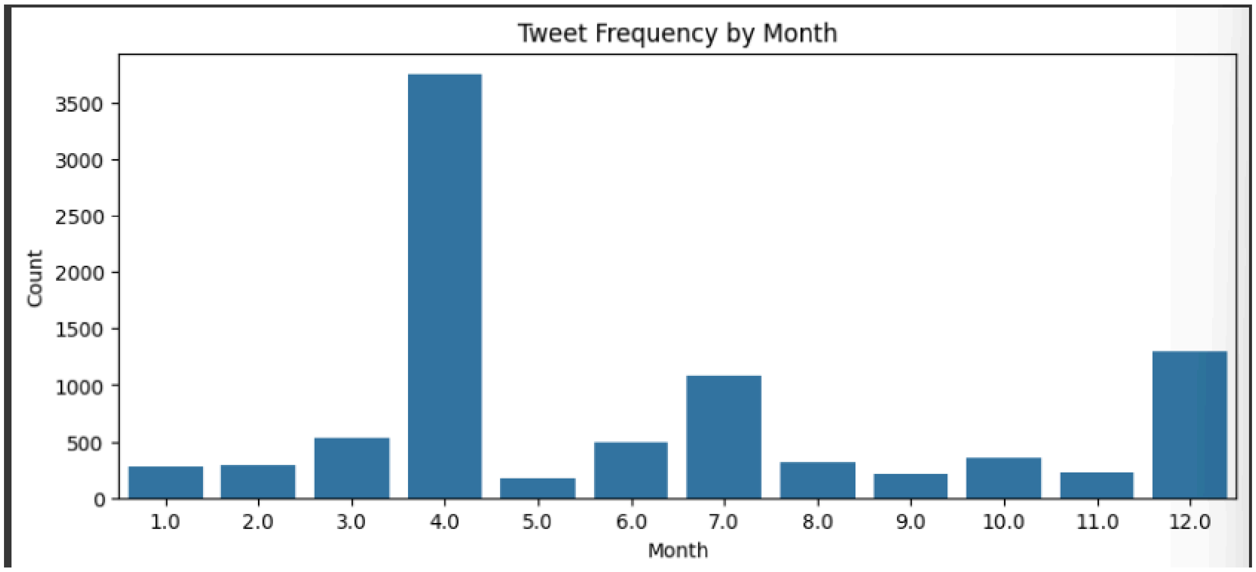
Day of the week



- **Monday Dominance:** Monday shows the highest frequency of tweets compared to all other days of the week. There's a noticeable drop in activity from Monday to Tuesday.
- **Mid-Week Dip:** Tweet frequency appears to be slightly lower during the middle of the week, specifically on Wednesday.
- **Consistent Weekends:** The tweet frequency is quite similar across the weekend days (Saturday and Sunday), and it's higher than the mid-week low but still lower than Monday.
- **Relatively Stable End of Week:** Thursday and Friday show tweet frequencies that are fairly close to each other and are higher than Wednesday but lower than the start of the week.

- **Weekly Pattern:** Overall, there seems to be a pattern of higher tweet activity at the beginning of the work week (Monday), a dip in the middle, and then a slight recovery towards the end and into the weekend, although the weekend activity doesn't reach the level seen on Monday.

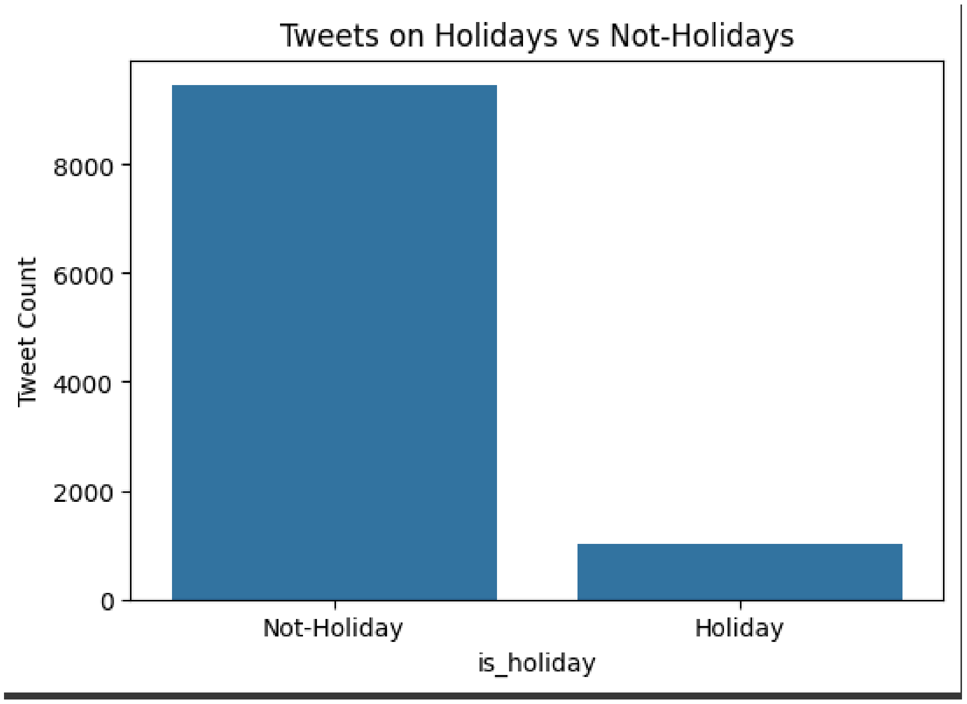
By Month



- **April Surge:** There's a very significant spike in tweet frequency in April (month 4). The number of tweets in April is dramatically higher than in any other month.
- **December Peak (Secondary):** December (month 12) also shows a relatively high tweet frequency, although not as extreme as April. It's the second highest month for tweet activity.
- **Low Mid-Year Activity:** The months from May through September (months 5 through 9) generally exhibit lower tweet frequencies compared to the beginning and end of the year.
- **January and February Similar:** January (month 1) and February (month 2) have fairly similar and relatively low tweet counts.
- **March Increase:** There's a noticeable increase in tweet frequency in March (month 3) compared to January and February, leading up to the April surge.

- **October and November Recovery:** After the mid-year low, there appears to be a gradual increase in tweet frequency in October (month 10) and November (month 11), leading into the high activity in December.
- **Seasonal Variation:** The chart suggests a strong seasonal pattern in tweet frequency, with peaks in the spring (April) and winter (December) and a lull during the summer months. This could be related to various factors like events, holidays, or changes in user behavior throughout the year.

Holiday Tweet Analysis



Significantly More Tweets on Non-Holidays: The bar representing "Not-Holiday" is substantially taller than the bar representing "Holiday." This indicates that there is a much higher volume of tweets on days that are not designated as holidays.

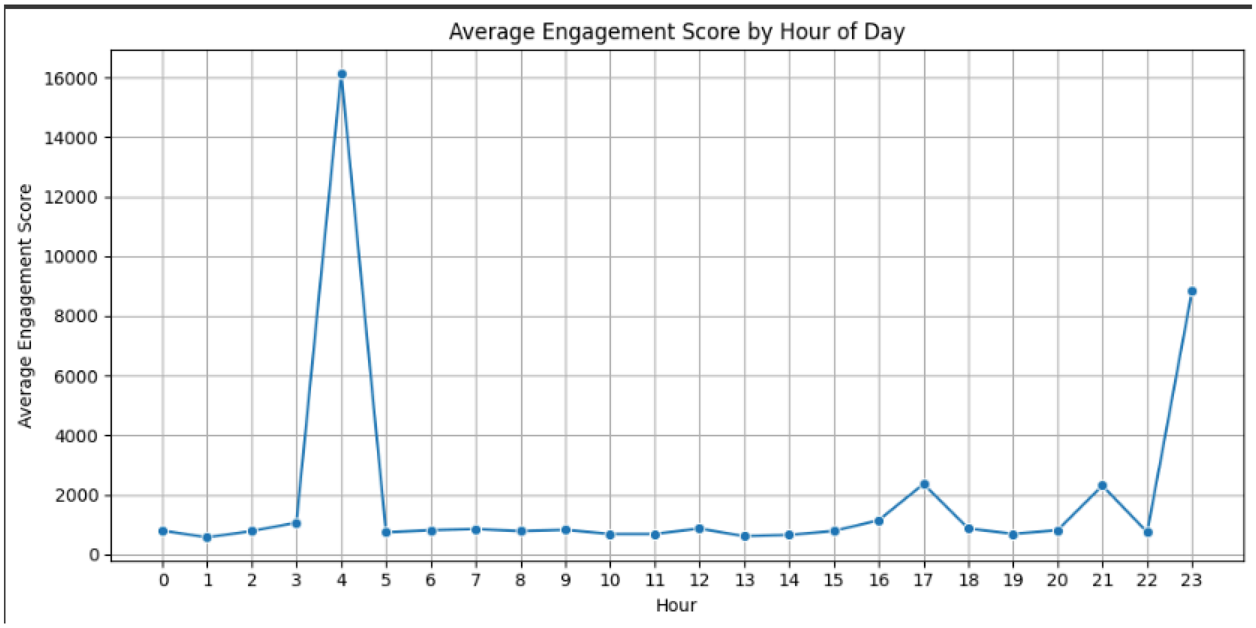
Lower Tweet Volume on Holidays: The "Holiday" bar shows a significantly lower count of tweets compared to non-holiday days.

Imbalance in Activity: There's a clear imbalance in Twitter activity, with users being far more active on regular days than on holidays.

Potential Explanations: This could be due to various reasons, such as people having different routines on holidays (e.g., spending time with family, traveling, being less engaged online), or simply that there are far more non-holiday days in a given period than there are holidays.

Bivariate Analysis

Engagement Score vs Hour of Day



Key Peaks:

Hour 4 (4 AM) shows an exceptionally high spike in engagement score (16,000), far above all other times.

- This could be an outlier or influenced by a few high-performing tweets during that hour.

Hour 23 (11 PM) also shows a secondary spike in engagement (8,800), indicating strong activity late at night.

General Pattern:

Engagement is relatively low and steady throughout the day, especially between 6 AM and 3 PM. There are minor increases around:

- 3 AM (1,200)
- 5 PM (2,300)
- 9 PM (2,300)

Low Engagement Hours

Hours 1, 2, 5–15 show the lowest engagement scores, with values generally below 1,000.

These may represent hours when audience activity or tweet reach is minimal.

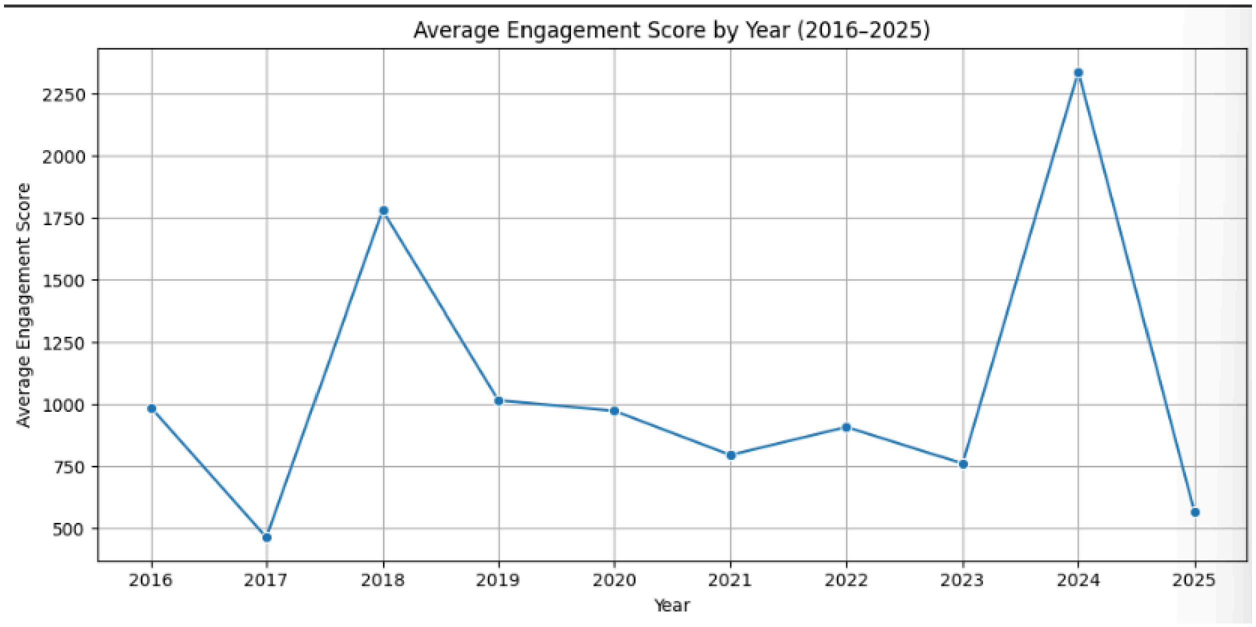
Interpretation

Late night (11 PM) and very early morning (4 AM) tweets may have disproportionately high engagement, possibly due to:

- Viral content shared across time zones
- Scheduled posts by media outlets or influencers
- Fewer tweets during these times increasing visibility

For optimal visibility and impact, posting at 11 PM or 4 AM could be strategic, though the 4 AM spike should be investigated further for anomalies.

Engagement Score vs Year



Key Trends

Engagement spiked in 2018 and 2024:

- 2018: Average engagement score surged to ~1,770 — possibly linked to key political or social events.

- 2024: A dramatic peak of over 2,300, the highest in the 10-year span, indicating an exceptionally engaging year (could correlate with election cycles or major controversies).

Significant drop in 2017 and 2025:

- 2017 saw the lowest engagement (470), likely a cooldown year following prior activity.
- 2025 also dropped sharply (570), following the 2024 spike — suggesting post-event fatigue or lower activity.

Stability in Mid Years (2019–2023):

- Engagement remained relatively stable in this period, ranging between 750 and 1,020, showing no dramatic shifts.

Interpretation

- Engagement appears to follow a cyclical pattern, with highs in select years — possibly aligned with national or global events.
- The peaks may correspond to election periods or major socio-political developments, which tend to amplify tweet interactions.
- Sharp year-over-year drops following peaks (e.g., 2018→2019, 2024→2025) may reflect reduced public discourse or event-driven engagement decline.

EngaEngagement Score vs Holiday Name

- **Easter Monday Dominance:** Easter Monday stands out with the highest overall tweet count, and notably, a significantly higher number of "Not_Hate" tweets compared to "Hate" tweets.
- **Good Friday Contrast:** While Good Friday also shows a substantial number of tweets, the proportion of "Hate" tweets is visibly higher compared to "Not_Hate" tweets, unlike Easter Monday. Lower Activity on Other Holidays: Most other holidays displayed on the chart (Christmas Day, Boxing Day, Eid al-Fitr, Mashujaa Day, New Year's Day, Jamhuri Day, Labour Day, Madaraka Day, and Eid al-Adha) show considerably lower tweet volumes compared to Easter Monday and Good Friday.
- **Varying Hate Speech Proportion:** The proportion of "Hate" versus "Not_Hate" tweets varies across these lower-activity holidays. Some, like Christmas Day and Boxing Day,

appear to have a relatively higher proportion of "Hate" tweets compared to others like Eid al-Fitr and Madaraka Day.
- **Eid al-Fitr and Eid al-Adha:** Both Eid holidays show relatively low overall tweet counts, with a slightly higher number of "Not_Hate" tweets compared to "Hate" tweets.
- **National Holidays:** Mashujaa Day, New Year's Day, Jamhuri Day, Labour Day, and Madaraka Day all exhibit low tweet frequencies, with the "Not_Hate" category generally being slightly higher than "Hate," although the absolute numbers are small.
- **Potential Spikes in Specific Sentiments:** The data suggests that certain holidays might be associated with different levels and types of online sentiment. For instance, Good Friday seems to have a higher prevalence of hate speech in this dataset compared to the generally positive sentiment observed on Easter Monday.

Entity involvement

William Ruto dominates Twitter conversation:

- With over 1,500 tweets, he has the highest volume by far. Notably, hate speech tweets are a significant portion (45%), indicating a highly polarized public sentiment.

Raila Odinga ranks second in tweet volume (540 tweets):

- A relatively balanced mix of hate and non-hate tweets, but still a high count of negative sentiment.

Oscar Sudi and Rigathi Gachagua receive a disproportionately high share of hate tweets relative to their total volume: For both, hate tweets make up more than half their mentions, suggesting strong public backlash or controversy.

Fred Matiang'i and Kithure Kindiki also attract considerable attention:

- Though hate tweet volumes are lower compared to others, overall tweet engagement is moderate, suggesting sustained public interest.

Rachel Ruto and Martha Koome receive the fewest tweets:

- These figures are less in the public discourse, with low sentiment (both positive and negative) activity.

Interpretation

- Politicians with higher profiles or controversial roles (e.g., William Ruto, Raila Odinga, Oscar Sudi) tend to receive more tweets, including more hate speech.
- Hate speech on Twitter appears to be disproportionately concentrated on a few individuals, which could indicate targeted online harassment or divisive public perception.
- Understanding the context (e.g., election campaigns, policy decisions, scandals) could further explain these sentiment distributions.

Engagement Score vs Politician

Top Engagement Politicians

Betty Maina and John Kiarie dominate the chart with exceptionally high average engagement scores, each exceeding 10,000.

- These scores are far above all other politicians, indicating their content likely went viral or generated significant public interest.
- This could be due to controversial statements, trending topics, or high-profile events involving them.

Moderate Engagement Group:

Alice Nganga, Noordin Haji, and Didmus Barasa form a second tier, with engagement scores ranging from 1,500 to 2,500.

- Their content resonates with the public, though not at viral levels.

Lower Engagement Politicians:

Martha Koome, Japheth Koome, Kalonzo Musyoka, and Johnson Sakaja have lower engagement (around 1,000 or below).

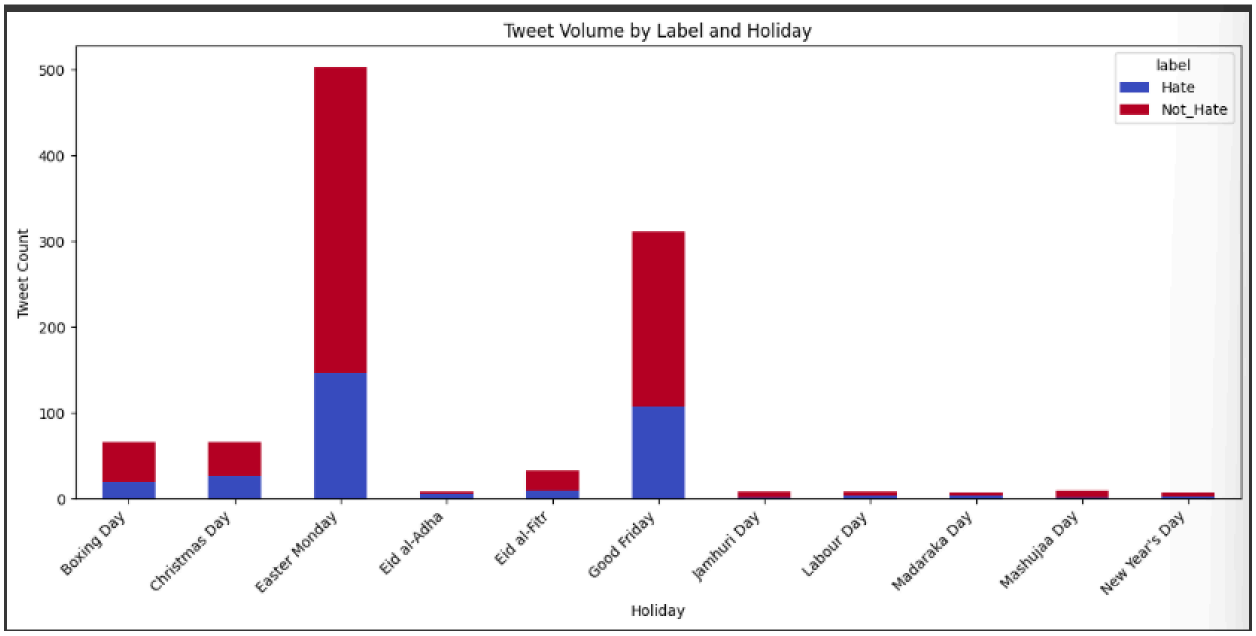
- This suggests either a less active online presence or lower public interaction with their content.

- The inclusion of "unknown" in the chart implies some tweets couldn't be attributed to a specific politician but still received moderate engagement.

Interpretation:

- High engagement does not always correlate with tweet volume — it may reflect public sentiment intensity (positive or negative).
- Politicians like Betty Maina and John Kiarie might benefit from their current online momentum but should also consider the nature of that engagement.
- Those with low engagement could revisit their communication strategies to boost public interaction.

Label vs Holiday



Dominance of Not_Hate Tweets Overall

- The red bars (Not_Hate) dominate most holidays, indicating that non-hostile content is the majority.

- Even during spikes in tweet activity, hate speech remains proportionally smaller but still noteworthy.

Easter Monday and Good Friday – Peaks in Activity

- Easter Monday has the highest tweet volume, with a significant chunk labeled as Hate (blue).
- Good Friday is also prominent, showing a visible presence of Hate tweets.
- Suggests these Christian holidays trigger increased engagement and some polarizing content.

Hate Speech is Still Present Across Many Holidays

- Boxing Day and Christmas Day show moderate volumes with a fair share of Hate tweets.
- Despite being celebrated holidays, these days see non-trivial levels of online hostility.

Muslim Holidays – Low Overall Volumes

- Eid al-Adha and Eid al-Fitr have relatively low tweet counts.
- Hate speech appears but is minimal compared to Christian holidays.
- Could reflect lower Twitter activity or less contentious online discourse.

Minimal Engagement on National Holidays

- Holidays like Labour Day, Jamuhuri Day, Madaraka Day, and Mashujaa Day see very low tweet volumes.
- Hate speech is almost negligible.
- These may not be strong triggers for political or emotional expression on Twitter.

Interpretations:

- Christian religious holidays show both high engagement and increased hate speech, highlighting the need for moderation strategies during these periods.
- National and Muslim holidays experience lower online interaction and less hate, possibly due to their cultural roles or audience reach.

- This pattern may help social platforms and policymakers anticipate and mitigate online hate spikes around specific holidays.

Engagement Score vs Holiday Name

Top Performing Holidays (Highest Engagement)

Madaraka Day leads by a wide margin in average engagement, suggesting:

- High interest and interaction from users.
- Possibly tied to strong political or nationalistic sentiments shared on this day.

Labour Day and New Year's Day also score highly:

- Likely due to posts that resonate with labor movements, new beginnings, or public celebrations.
- Engagement may come from both individual reflections and institutional messages.

Non-Holiday (not_holiday) content ranks 3rd, suggesting:

- On average, tweets outside of holiday contexts still garner strong interaction.
- Implies that everyday discourse might be more engaging than some holidays.

Low Engagement Holidays

Easter Monday, Mashujaa Day, and Christmas Day are among the lowest:

- Surprisingly low engagement despite high tweet volume on Easter Monday.
- This could indicate content is posted frequently but resonates less with the audience.

Mashujaa Day and Christmas Day may reflect:

- Lower online interaction.
- Possibly more family-oriented offline activities than social media discourse.

Interpretations:

High engagement does not always correlate with high volume.

- For example, Easter Monday had high tweet volume but very low engagement, suggesting a quantity-over-quality pattern.

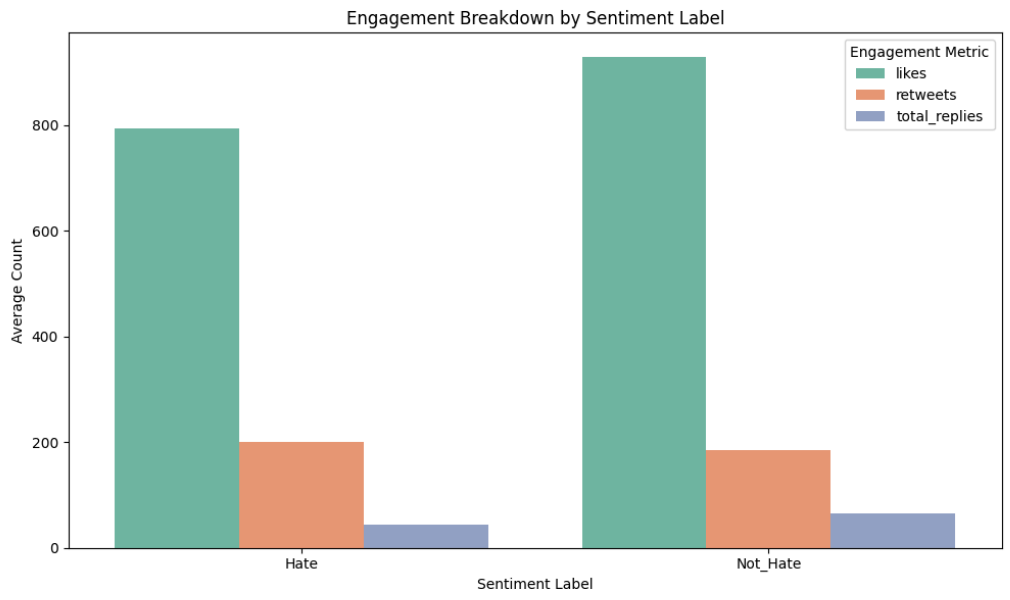
National holidays like Madaraka Day are potent for online discourse.

- These may be strategic moments for civic campaigns, political announcements, or activism.

Engagement varies by holiday type:

- Civic/political holidays tend to trigger deeper online engagement than religious ones, which may reflect more ritual or passive celebration.

Engagement Breakdown



Observations:

Likes:

Not_Hate tweets received more average likes (approx. 930) than Hate tweets (approx. 790). This suggests that positive or neutral content tends to be more widely appreciated by users.

Retweets:

Hate tweets had a slightly higher average retweet count than Not_Hate tweets (~200 vs. ~185). This may indicate that controversial or provocative content is more likely to be shared, even if it's not necessarily endorsed.

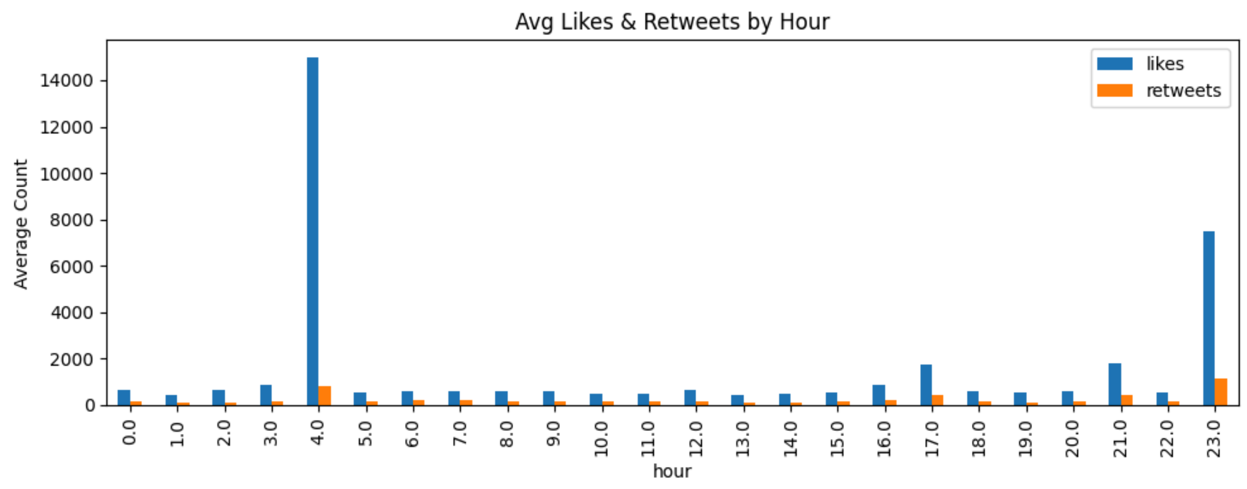
Replies:

Not_Hate tweets elicited more replies than Hate tweets (~70 vs. ~45). This could reflect a greater willingness by users to engage in constructive dialogue with non-hostile content.

Interpretation

- While Hate tweets may generate amplification through retweets, Not_Hate tweets are more effective at generating positive engagement through likes and replies.
- This indicates that although hate-driven content spreads, it does not build community or conversation as effectively as neutral or positive content.
- For content creators or platform moderators, promoting Not_Hate content could lead to healthier and more interactive discourse.

Likes / Retweets vs Time Features (hour, day, month)



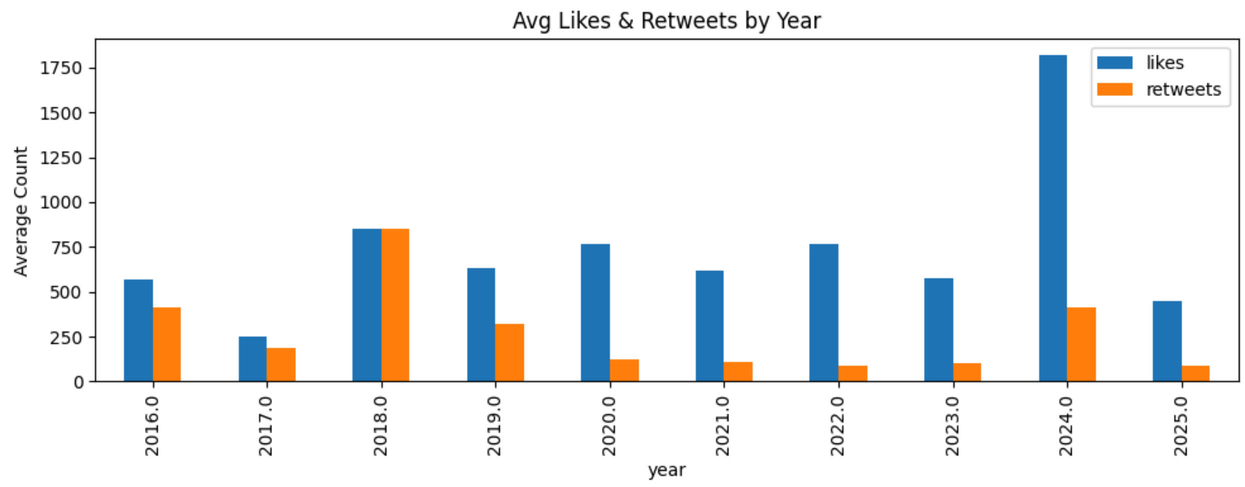
- **Dominance of Likes:** Across almost all hours, the average number of likes is significantly higher than the average number of retweets. This reinforces the earlier observation that likes are the most frequent form of engagement.
- **Sharp Spike in Likes at Hour 4:** There's a very prominent spike in the average number of likes at hour 4. This hour stands out as having exceptionally high like engagement compared to all other hours.
- **Smaller Increase in Likes at Hour 23:** We also see a noticeable, though less extreme than hour 4, increase in the average number of likes at hour 23.
- **Relatively Low and Stable Retweets:** The average number of retweets remains relatively low and stable throughout most of the day, with no dramatic spikes comparable to the likes.
- **Slight Increase in Retweets at Hour 23:** Similar to likes, there's a small increase in the average number of retweets at hour 23, though it's much less pronounced.
- **Morning Dip in Likes:** The average number of likes is generally lower in the very early morning hours (around 0 to 3).
- **Daytime Consistency:** During most of the daytime hours (roughly 6 to 17), the average number of both likes and retweets stays at a relatively consistent and low level.

Average Likes and Tweets by Month

- **Likes Consistently Higher:** Across all months, the average number of likes is notably higher than the average number of retweets. This aligns with previous observations.
- **Peak in Likes in July (Month 7):** July exhibited a significant peak in the average number of likes, standing out considerably from the other months.
- **Increase in Likes Around June:** There's a noticeable increase in the average number of likes starting around June, leading up to the July peak.
- **Retweets Peak in July as Well:** The average number of retweets also peaks in July, coinciding with the peak in likes, although the increase is less dramatic relative to the baseline.
- **Generally Lower Engagement in Early Months:** The average number of both likes and retweets tends to be lower in the earlier months of the year (January to April).

- **Moderate Engagement in Later Months:** Following the July peak, the average engagement for both likes and retweets generally returns to more moderate levels for the remaining months of the year.
- **Similar Monthly Trends:** The trends for average likes and retweets across the months appear somewhat similar, suggesting that months with higher like counts also tend to have higher retweet counts.

Average Likes and Tweets by Year



Spike in 2024:

- The dramatic increase in average likes in 2024 likely reflects a major political event, possibly elections, scandals, or high-profile political debates that drove engagement.
- Despite the surge in likes, retweets remained moderate, which might suggest people found the content engaging but didn't feel compelled to share it.

High Engagement in 2018:

- Both likes and retweets peaked equally in 2018, indicating highly shareable content.
- This could be linked to the aftermath of the 2017 general elections or major political announcements during the year.

Decline in Retweets (2020–2023):

This period saw consistent likes but a sharp drop in retweets, possibly pointing to:

- More passive engagement
- Content that sparked opinions but wasn’t considered worth amplifying

2025 Drop-off:

The sharp decline in both metrics for 2025 might be due to:

- Incomplete data (e.g., early in the year)
- Reduced public interest in political content

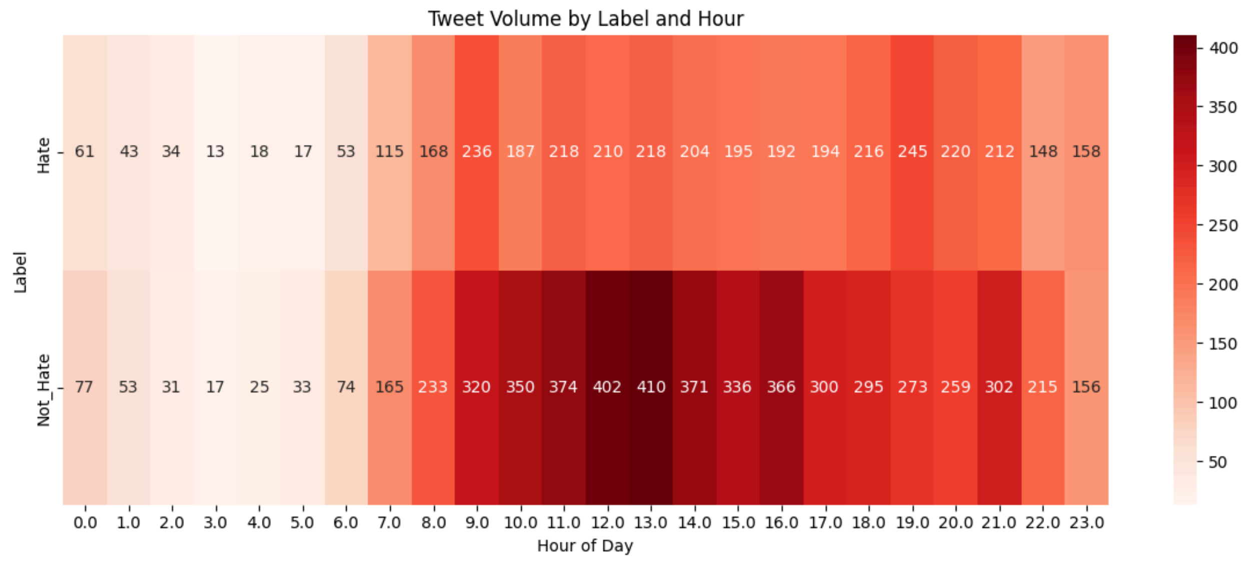
2017 Dip:

Despite being an election year in Kenya, 2017 shows unexpectedly low engagement.

Possible reasons include:

- Censorship
- Social media fatigue
- A smaller Twitter audience at the time

Sentiment analysis by Hate Label vs Time



Label	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0	17.0	18.0	19.0	20.0	21.0	22.0	23.0
Hate	61	43	34	13	18	17	53	115	168	236	187	218	210	218	204	195	192	194	216	245	220	212	148	158
Not_Hate	77	53	31	17	25	33	74	165	233	320	350	374	402	410	371	336	366	300	295	273	259	302	215	156

Observations

General Patterns:

- Tweet activity increases sharply starting at 6 AM, peaks around 12 PM to 2 PM, and gradually declines into the night.
- The lowest activity for both labels occurs between 2 AM and 5 AM, reflecting typical offline sleeping hours.

Hate Tweets:

Peak hours for Hate tweets: 9 AM to 2 PM (with a mini-peak again at 7 PM - 8 PM).

- Highest volume at 9 AM (236 tweets).
- Consistent presence throughout the day, even into late hours (e.g., 150 tweets at 11 PM).

Hate tweet volume remains relatively stable between 10 AM and 9 PM, suggesting:

- Sustained discourse and engagement on controversial or divisive topics during active hours.

Not_Hate Tweets

Peak posting occurs between 10 AM and 2 PM, with 12 PM being the highest (402 tweets).

- Suggests that general or neutral discourse is more common during midday.
- Drop-off begins around 3 PM but still maintains a high level until 8 PM.

Morning ramp-up starts earlier for Not_Hate tweets (from 6 AM), reflecting typical workday or news consumption patterns.

Interpretation:

Midday (10 AM – 2 PM) is the most active window for both Hate and Not_Hate tweets.

- Strategic timing for moderation, counter-speech, or promotional posting could be targeted within this slot.

Evening resurgence of Hate tweets (7–9 PM) may indicate:

- Reactions to daily events.
- Increased personal posting after work hours.

Night-time (2 AM – 5 AM) shows minimal activity, suggesting a natural break in discourse cycles.

Multivariate Analysis

Engagement by Label categories

Faceted barchart (separated by Likes, Retweets, Replies, and Total Engagement) was plotted to display the distribution of engagement bins across different label categories.
Observations:

- Not_hate tweets dominate all engagement types and bins.
- Hate tweets tend to cluster more in lower engagement bins.
- The number of tweets sharply drops in higher engagement bins across all categories.

Insights from Likes

- Most tweets with no likes (bin 0) are not_hate, but a significant number are also hate.
- As engagement increases, not_hate tweets become even more dominant.
- Very few hate tweets receive 10k+ likes.

Insights from Retweets

- Similar pattern to likes: not_hate tweets are most common, even more so in higher bins.
- Very hate tweets make it past the 101-1k retweet range.
- No hate tweets in the 10k+ retweet bin.

Insights from Replies

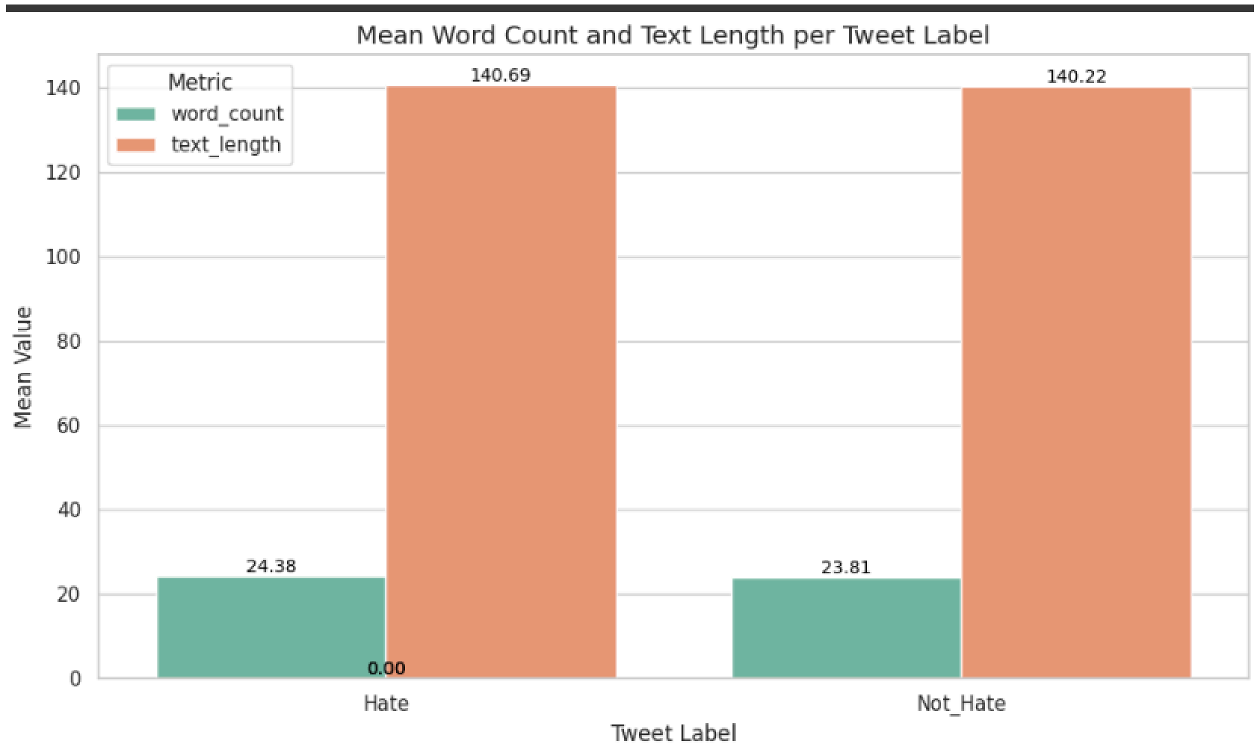
- Replies show a higher proportion of hate tweets, especially at 0 and 1–10 reply bins.
- Still, non_hate tweets dominate in total count, especially in the 1–100 reply range.
- A steep drop-off in all categories beyond the 101–1k bin.

Insights from Total Engagement

- The total view mirrors the trends from individual categories:
- Non_hate tweets are consistently the most engaged.
- Hate tweets are far less likely to receive high engagement.

Word Count vs Text Length per Label Category

A grouped bar plot compared the mean word count and mean text length for tweets in each label category (hate, offensive, neutral).



Observations:

Text Length (in characters):

Hate tweets - (166.01)

not hate tweets - (165.99)

- Suggests that hate tweets tend to be slightly longer, possibly due to more elaborate or emotionally charged language.
- The average tweet length falls between 165-166 characters for all categories

Word Count:

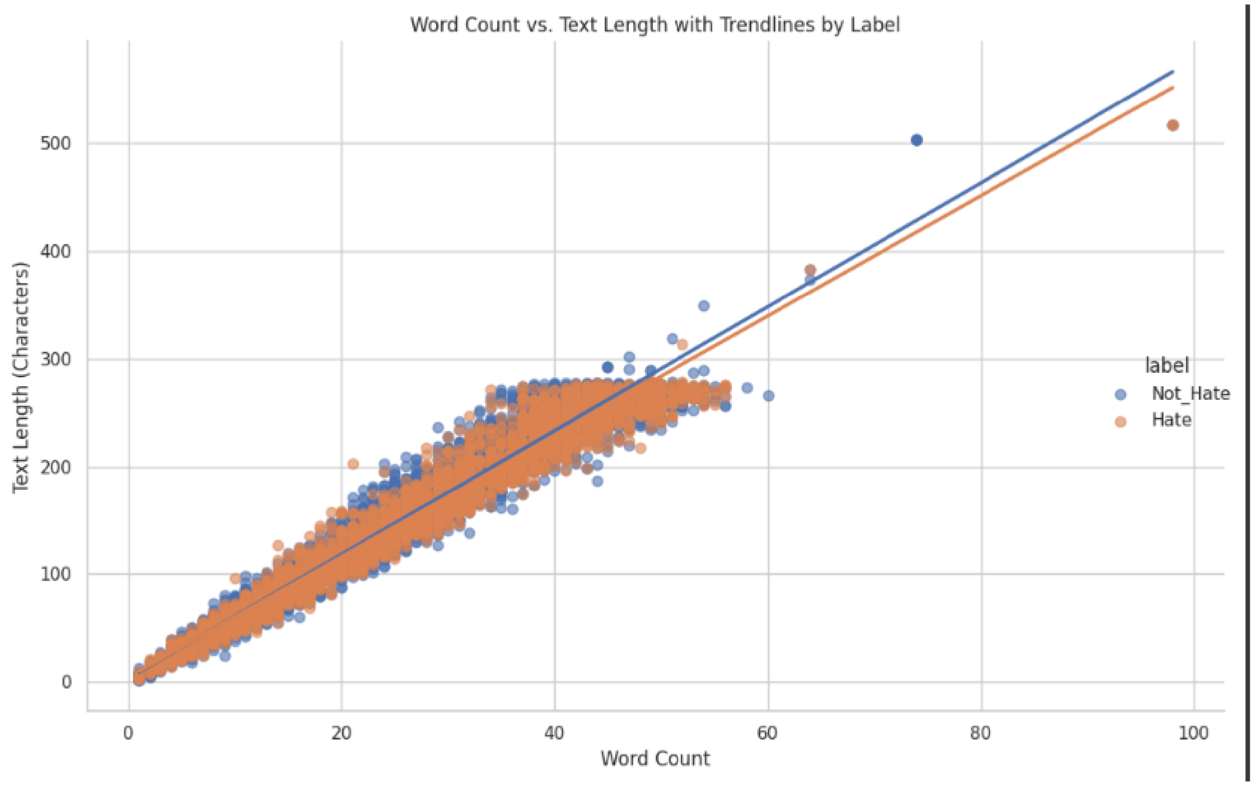
Hate tweets - (28.65)

not hate tweets - (27.92)

- The variation is relatively small but may indicate that hate tweets use slightly more words, possibly to express more complex or heated messages.
- The average tweet word count falls between 27-29 words for all categories.

Distribution of Word Count vs Text Length per Label Category

A scatter plot with trendlines was created to visualize the relationship between word count and text length for each label.



Observations:

Positive Correlation:

- A positive trend line across most or all categories suggests that as tweet length increases, the number of words also increases.
- This makes intuitive sense: longer tweets can hold more words.

Category Differences:

- Some categories tend to use longer or more complex words (lower word count for same tweet length).

- Others may use shorter, more concise language (higher word count for same tweet length).

Outliers:

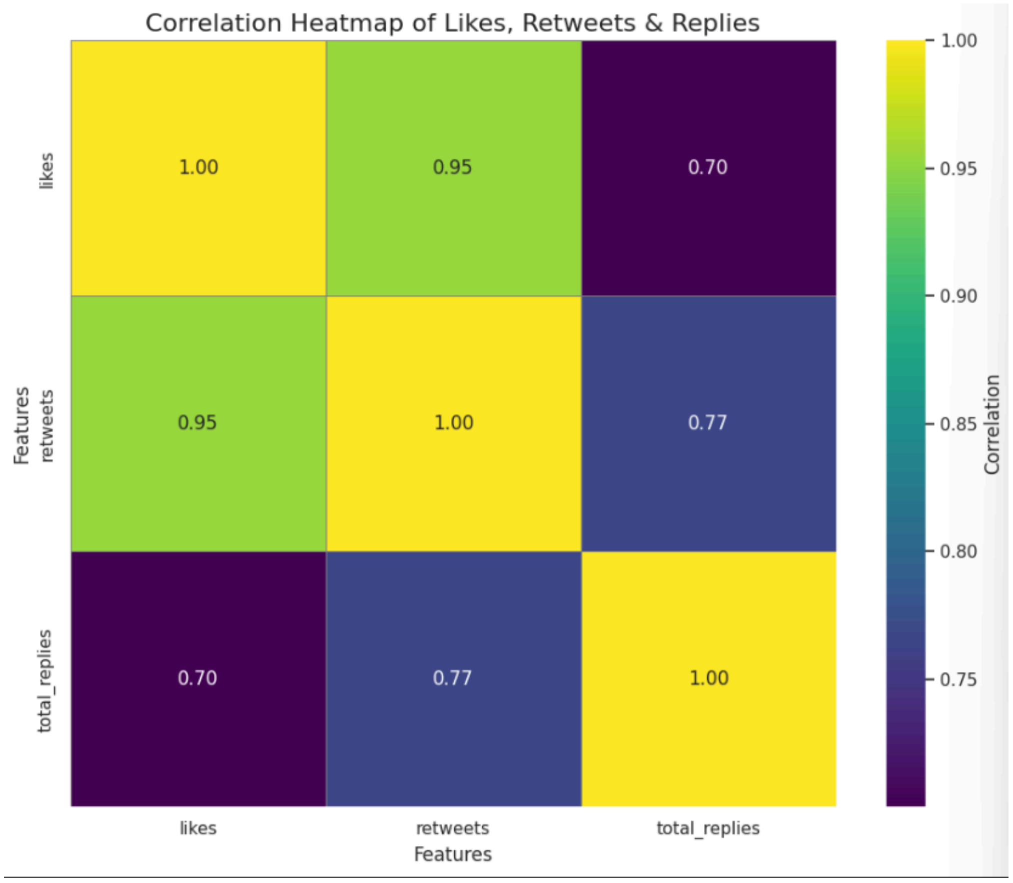
- Points far from the trend line could be outliers most likely spam content, unusually wordy or concise tweets.

Compactness of Categories:

- Tightly clustered points imply similar writing style or length.

Correlation Heatmap (Likes, Retweets, Replies)

Visualization:A heatmap visualized the correlation coefficients between likes, retweets, and total replies.



Observations:

Strong Positive Correlation Between Likes and Retweets

- Correlation ≈ 0.95+ (very close to 1)
- Indicates that tweets getting more likes are also highly likely to be retweeted, and vice versa.
- These two features are likely reflecting similar audience engagement behaviors.

Moderate Correlation Between Retweets and Replies The value seems lower (around ~0.76).

- Suggests that while retweeted tweets may get replies, it's not as tightly linked as likes/retweets.
- Retweets may spread content but not always spark conversation.

Lowest Correlation Between Likes and Replies The correlation here is the weakest among the three pairs.

- Implies that just because a tweet is liked doesn't mean it generates a reply or discussion.

Pair Plot of Numeric Features

A pair plot was created to explore the relationships among multiple numeric variables: likes, retweets, total replies, engagement score, text length, and word count.

Observations:

- likes vs retweets Strong positive correlation — more retweets usually mean more likes.
- likes vs engagement_score Strong positive correlation — engagement score depends heavily on likes.
- retweets vs engagement_score Also positive but slightly more spread compared to likes.
- text_length vs word_count Strong linear relationship — as expected, more words lead to longer text.

- likes, retweets vs total_replies Some positive correlation but more scattered — replies vary a lot even for high likes or retweets.
- total_replies vs engagement_score Some relationship, but engagement is driven more by likes and retweets than replies alone.
- text_length, word_count vs likes/retweets Very weak relationship — longer text doesn't guarantee higher engagement (likes/retweets).

Machine Learning preprocessing.

Text Cleaning and Normalization

Text data underwent a series of cleaning and normalization steps:

- **Lowercasing:** All characters were converted to lowercase to eliminate case-based distinctions.
- **Number and Punctuation Removal:** Numeric characters and punctuation marks were stripped out to focus purely on linguistic content.
- **Tokenization:** The cleaned text was broken down into individual words (tokens).
- **Stopword Removal:** Common words that carry little meaning both in English and a custom list of Swahili stop words were removed.
- **Lemmatization:** Each remaining token was reduced to its base form (lemma) to unify different inflected variations of the same word.

As a result of these steps, each original text entry was transformed into a concise, normalized sequence of meaningful tokens (words).

Feature Extraction with TF-IDF

The normalized tokens were then transformed into numerical feature vectors using Term Frequency–Inverse Document Frequency (TF-IDF). By limiting to the top 5,000 features, the representation captured the most informative words across the corpus while managing dimensionality.

Train-Test Split

The dataset was split into two subsets: 70% for training and 30% for testing. Stratification ensured that both subsets maintained the same proportion of neutral and hate labels, which is crucial for fair evaluation of model performance.

Modelling

Machine Learning Models

We began by initializing our custom Text_Classifier pipeline, downloading the text_classifier.py module from GitHub and importing it into our Colab environment. This module encapsulated six ready-to-use models—Naive Bayes, Logistic Regression, Linear SVM, Random Forest, Gradient Boosting, and a Neural Network—and built-in support for handling class imbalance via SMOTE. After appending our working directory to sys.path, we instantiated TextClassifier with a fixed random seed and provided it our TF-IDF feature names to enable later feature‑importance analysis.

Base Models

We registered our six base learners with the classifier:

- **Naive Bayes**
- **Logistic Regression** (balanced classes)
- **Linear SVM** (balanced classes)
- **Random Forest**
- **Gradient Boosting**
- **Neural Network**

We opted to include class‑weight adjustments to counter our label imbalance.

Training Base Classifiers

We trained all six models on our TF-IDF vectors (with SMOTE oversampling applied) to ensure sufficient representation of the minority class. Each model printed a “Training …” message as it fit, confirming successful completion.

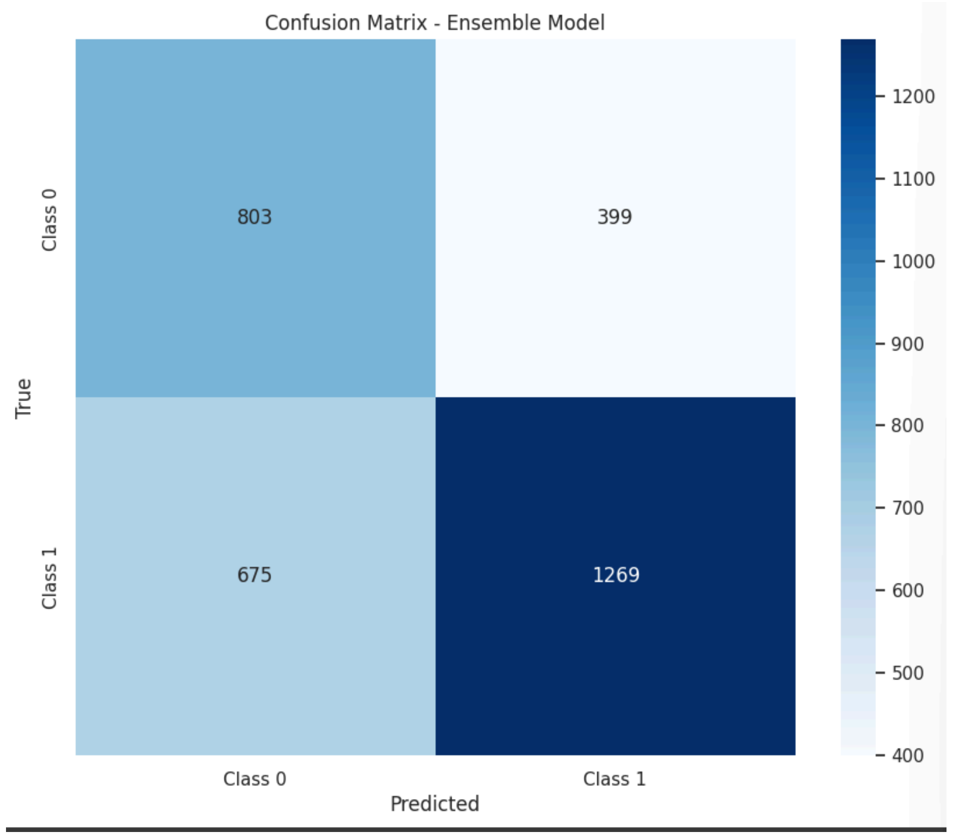
Evaluating Base Models & Visualization
We then evaluated each model on the held-out test set, computing both accuracy and macro-averaged F1. To diagnose per-class performance, we plotted a confusion matrix for each model. From these evaluations, we identified the best base model on F1_score is Logistic classifier with F1 score: 0.6609. However the f1_score is average meaning the model was classifying fairly okay. The best way forward is to perform a hyperparameter search on the base models to get the best performing parameters for the models ensuring the best F1 score.

Hyperparameter tuning.

Since the models performed averagely from the average accuracy and F1 score. We attempted to tune the hyperparameters to get the parameters that will give us the best F1 score. The default cross validation search method for our pipeline is a GridsearchCV. We used a support vector machine. The best parameters for the Linear Support vector machine still gave an average F1 score and accuracy but it was not higher than the best base model which had an f1 score of 65.81%. This was however not a very reliable model with an accuracy of 65.72%. We tried to ensemble the models to see if performance will increase

Ensembling the best performing models.

We attempt to ensemble the best performing models to see if they will perform better by covering up each other's weaknesses. We plotted a confusion matrix.



From this confusion matrix we can see our model is still misclassifying the minority classes heavily but performing somewhat well on the majority class.

Next steps from Machine learning.

- Machine learning models performed very poorly especially on the minority classes even after dealing with imbalance meaning we might need more data to train the models on for them to capture underlying patterns well.
- We can try the deep learning approach to see if deep learning models might outperform the machine learning models and improve on their classification ability.

Deep Learning Models

We shifted to deep learning approaches using Hugging Face Transformers because our traditional models were underperforming, especially in the minority class. Transformers represent the state of the art in NLP and excel at handling imbalanced, multilingual datasets.

Why We Used Hugging Face Transformers

- **Multilingual Understanding**
  We encountered tweets in English, Swahili, and the Kenyan Sheng dialect. We needed models like XLM-RoBERTa (and DeBERTa) that were pretrained on hundreds of languages to capture contextual meaning across this linguistic diversity.
- **Contextual Embeddings**
  We replaced bag-of-words/TF-IDF pipelines with transformers that generate embeddings sensitive to surrounding words, enabling subtle distinctions essential for hate speech detection.
- **Transfer Learning**
  Given our modest dataset (~9,700 tweets), we leveraged pretrained weights and fine-tuned on our data instead of training from scratch.

Microsoft/deberta-v3-base model

We selected Microsoft's DeBERTa-V3-base for its:

- Disentangled Attention Mechanism- Separates content and position embeddings during attention, improving contextual understanding.
- Enhanced Mask Decoder- Improves the masked language modeling objective by using a more refined decoding strategy during pre training.
- Better Generalization- Achieves higher accuracy than BERT and RoBERTa on various NLP benchmarks.
- Efficient- It provides better performance with fewer parameters compared to older models.

First trial with 2e-5 training rate, batch size of 16 , 4 epochs and weight decay of 0.01.

- **Cleared GPU Memory**
  We called CUDA cache clearing and garbage collection to maximize available VRAM.

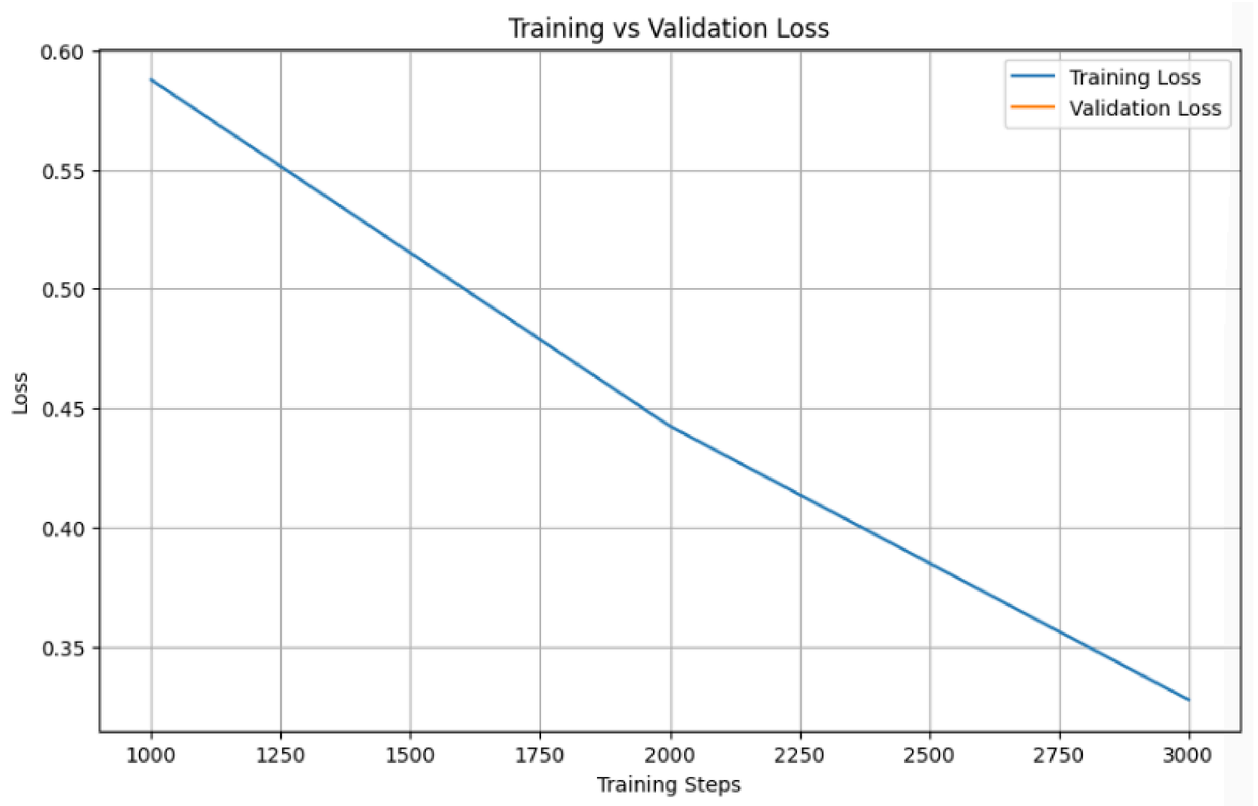
- **Loaded and Cleaned Data**
  We imported the CSV, lowercased labels, and filtered to ‘neutral’, ‘hate’, and ‘offensive’ to maintain consistency.
- **Merged Labels**
  We consolidated ‘hate’ and ‘offensive’ into ‘hatespeech’ and relabeled ‘neutral’ as ‘not_hatespeech’ to define a binary task.
- **Encoded Labels**
  We transformed textual labels into integers via LabelEncoder.
- **Train-Validation Split**
  We performed an 80/20 stratified split (random_state=42) to preserve class ratios.
- **Loaded Model & Tokenizer**
  We instantiated microsoft/deberta-v3-base and its tokenizer for sequence classification with two labels.
- **Prepared Datasets**
  We built Hugging Face Dataset objects from texts and labels for efficient I/O.
- **Tokenization**
  We applied truncation-only tokenization to handle variable-length tweets.
- **PyTorch Formatting**
  We set each dataset to output input_ids, attention_mask, and label as PyTorch tensors.
- **Data Collator**
  We used DataCollatorWithPadding to batch and pad dynamically, boosting throughput.
- **Training Arguments**
  We configured learning rate, batch sizes, weight decay, and evaluation/logging every 500 steps.
- **Metrics Function**
  We computed macro F1 and accuracy to evaluate minority-class performance.

- **Trainer Initialization**
  We instantiated Hugging Face’s Trainer to orchestrate training, evaluation, and checkpointing.

- **Model Training**

  We executed trainer.train(), letting the framework manage optimization.
- **Visualization**

  We plotted “Training vs Validation Loss”.



This model had an accuracy score of 68.52% and an F1 score of 67.48%. This was a fairly good model. It was classyfying quite better than the machine learning models. We could however try reducing the learning rate and increasing batch size to see if it would improve performance.

Second trial with 1e-6 training rate, batch size of 32 , 25 epochs, an early stopping with a patience of 2 to allow 2 logs where it doesn't show significant improvement in the validation accuracy and weight decay of 0.01.

- **Cleared GPU Memory**
  We again freed CUDA memory and collected garbage.
- **Data Loading & Cleaning**
  We replicated the first trial's data preparation pipeline for consistency.
- **Merging & Encoding Labels**
  We maintained the same binary label scheme and encoding.
- **Dataset Splitting**
  We stratified an 80/20 split with identical random state.
- **Model & Tokenizer**
  We reloaded microsoft/deberta-v3-base for direct comparability.
- **Dataset Preparation & Tokenization**
  We added padding to a max length of 256 to standardize input lengths.
- **PyTorch Formatting & Data Collator**
  We retained the same tensor formatting and dynamic padding.
- **Training Arguments**
  We lowered the learning rate, increased batch size, extended epochs, enabled load_best_model_at_end=True, and set early stopping with patience of three evaluation steps.
- **Trainer with Early Stopping**
  We attached an EarlyStoppingCallback to halt training when validation loss ceased improving.
- **Model Training**
  We ran trainer.train() under the new hyperparameter regime.
- **Visualization**
  We replotted "Training vs Validation Loss" to compare convergence behavior.

Third trial with 2e-6 training rate, batch size of 32 , 15 epochs and weight decay of 0.1.

- **Resource Clearing & Data Prep**

  We repeated GPU cleanup and the established data pipeline.
- **Switched to DeBERTa-V3-small**

  We reduced model size to assess performance vs. efficiency trade-off.
- **Tokenization**

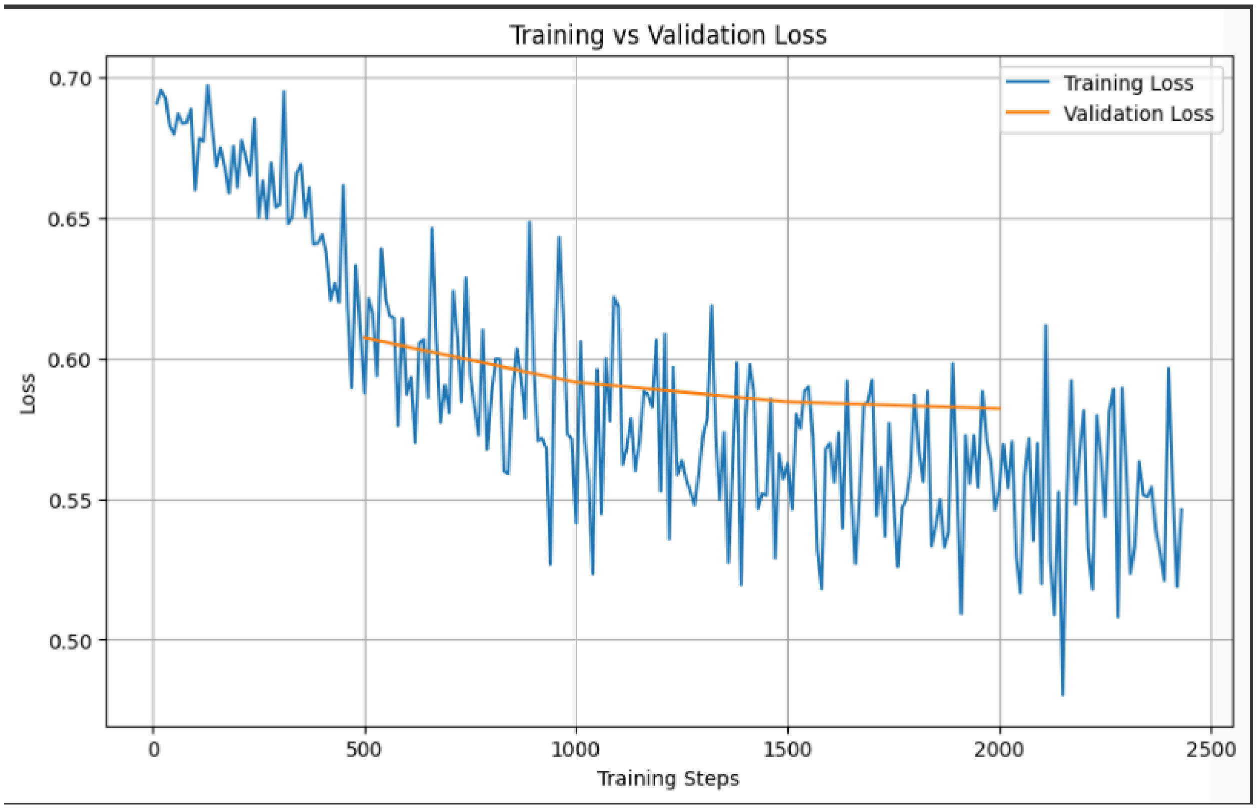
  We reverted to truncation-only tokenization.
- **Formatting, Collating, & Trainer Setup**

  We applied the same tensor formatting, collator, and training callbacks.
- **Training Arguments**

  We maintained batch size, adjusted learning rate, and increased weight decay to 0.1.
- **Training**

  We invoked trainer.train() and monitored through logs.
- **Visualization**

  We plotted the loss curves once more for side-by-side comparison.



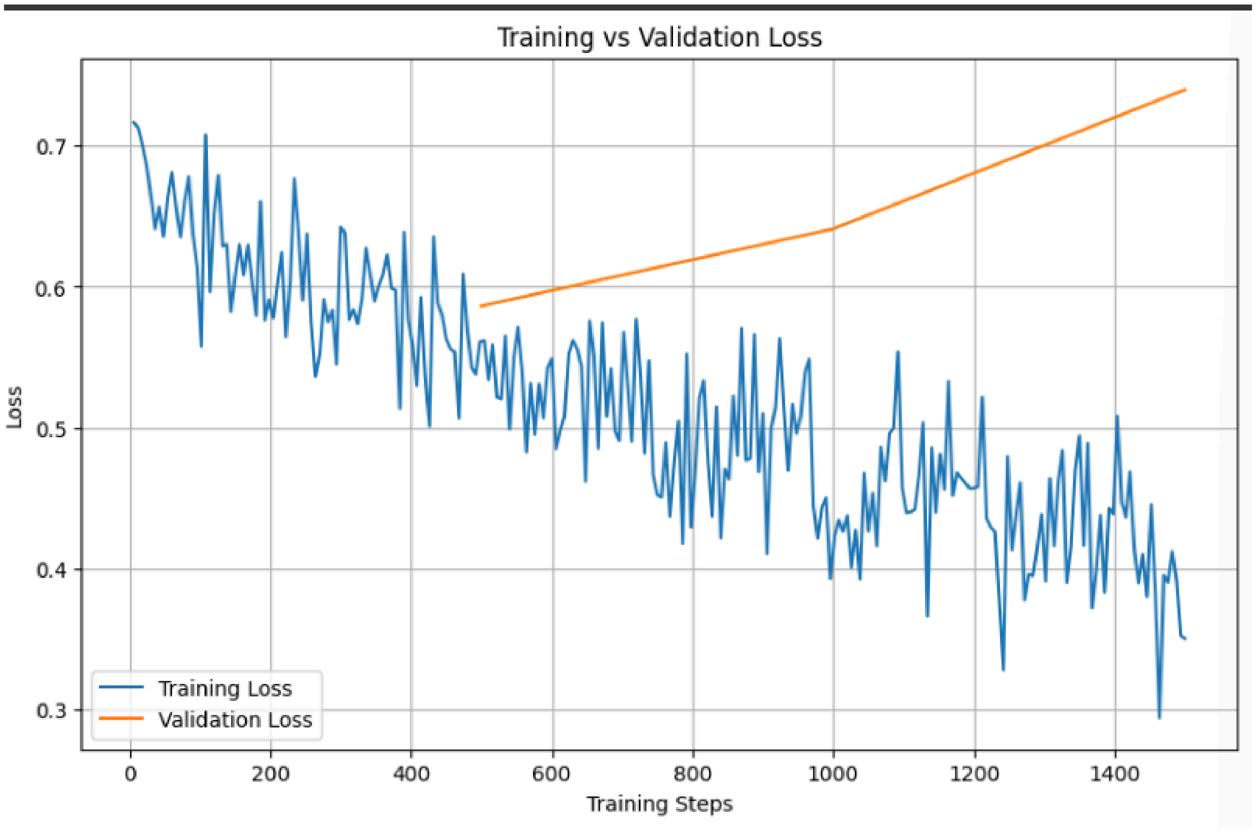
This model had an accuracy score of 68.32% and an F1 score of 66.95%. This was a fairly good model. However it isn't performing better than model 1."

Fourth try with 1e-6 learning rate, batch size of 32, 15 epochs , 0.1 weight decay and an early stopping with a patience of 2 to allow 2 logs where it doesn't show significant improvement in the validation accuracy.

- **Cleared GPU Memory**
  We invoked CUDA cache clearing and garbage collection to maximize available VRAM before model loading.
- **Loaded and Cleaned Data**
  We read the CSV, lowercased labels, and filtered for 'neutral', 'hate', and 'offensive' to maintain consistency.

- **Merged Labels**
  We combined ‘hate’ and ‘offensive’ into ‘hatespeech’ and relabeled ‘neutral’ as ‘not_hatespeech’ to define a binary classification.
- **Encoded Labels**
  We applied LabelEncoder to convert textual labels into integer form.
- **Train-Validation Split**
  We performed an 80/20 stratified split (random_state=42) to preserve class proportions.
- **Loaded Model & Tokenizer**
  We initialized the microsoft/deberta-v3-base sequence-classification model and its tokenizer.
- **Prepared Datasets**
  We created Hugging Face Dataset objects from our texts and labels for efficient batch processing.
- **Tokenization**
  We used padding to a maximum length of 256 tokens with truncation to standardize input dimensions.
- **Formatted for PyTorch**
  We set each dataset to output input_ids, attention_mask, and label as PyTorch tensors.
- **Configured Data Collator**
  We employed DataCollatorWithPadding to dynamically pad batches at runtime.
- **Defined Training Arguments**
  We set learning_rate=1e-5 (intended 1e-6), batch_size=32, epochs=6 (intended 15), weight_decay=0.1, logging every 6 steps, evaluation and checkpointing every 500 steps, and load_best_model_at_end=True based on validation loss.
- **Built Metrics Function**
  We computed macro-averaged F1 and accuracy to fairly evaluate minority-class performance.
- **Initialized Trainer with Early Stopping**
  We attached an EarlyStoppingCallback with patience=2 to halt training when validation loss failed to improve for two consecutive evaluation points.

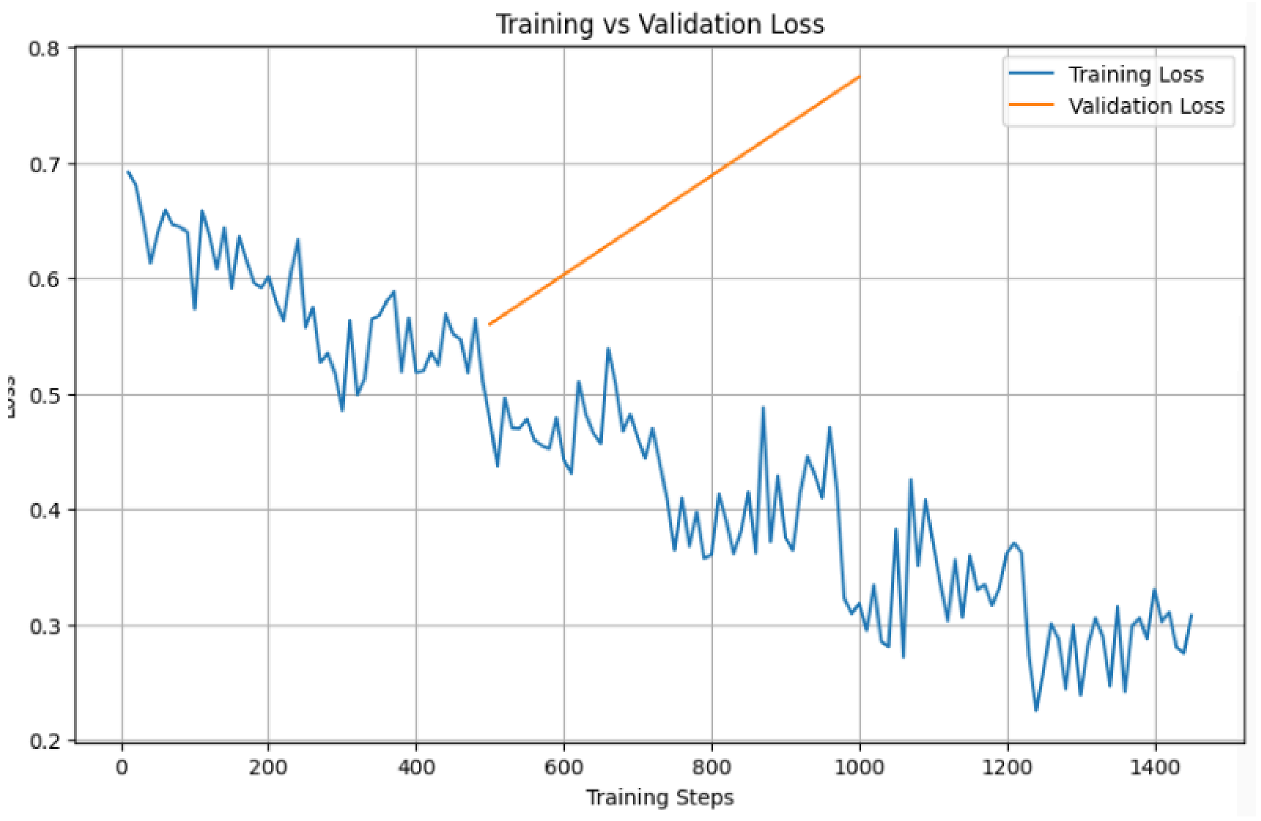
- **Trained the Model**
  We executed trainer.train(), letting the framework handle optimization and checkpointing.
- **Visualization**
  We plotted **Training vs Validation Loss**, observing convergence behavior across steps.



This model had an accuracy score of 69.19% and an F1 score of 66.43. This was a fairly good model. However it was not performing better than model 1 on F1 score.

Fifth try with 2e-6 learning rate, batch size of 32, 15 epochs , 0.1 weight decay and an early stopping with a patience of 2 to allow 2 logs where it doesn't show significant improvement in the validation accuracy.

- **Cleared GPU Memory**
  We again freed CUDA caches and ran garbage collection for reproducible GPU availability.
- **Loaded & Cleaned Data; Merged & Encoded Labels**
  We repeated our established pipeline, renaming ‘neutral’ to ‘non_hatespeech’ for this trial.
- **Train-Validation Split**
  We used the same 80/20 stratified approach to ensure consistent class ratios.
- **Loaded Model & Tokenizer**
  We re-instantiated microsoft/deberta-v3-base and its tokenizer for direct comparability.
- **Prepared & Tokenized Datasets**
  We applied max_length=256 padding and truncation for uniform input sizes.
- **PyTorch Formatting & Data Collator**
  We retained the same formatting and dynamic padding strategy.
- **Training Arguments**
  We increased the learning rate to 3e-5, reduced epochs to 6, maintained batch_size=32, weight_decay=0.1, and early stopping on validation loss.
- **Trainer Initialization**
  We used the EarlyStoppingCallback (patience=2) to avoid over-training.
- **Model Training**
  We ran trainer.train() under these hyperparameters.
- **Visualization**
  We replotted Training vs Validation Loss to compare against prior runs.



This model had an accuracy score of 70.48% and an F1 score of 69.41%. This was a good model. It was classifying quite better than the first model due to a higher F1 score and the higher accuracy. We saved this model in preparation for deployment.

Sixth try with 3e-6 learning rate, batch size of 32, 10 epochs , 0.1 weight decay and an early stopping with a patience of 2 to allow 2 logs where it doesn't show significant improvement in the validation accuracy.

- **Cleared GPU Memory & Data Prep**
  We repeated our trusted pipeline for consistency across experiments.
- **Loaded Model & Tokenizer**
  We once again initialized the base DeBERTa-V3 model and tokenizer.

- **Tokenization, Formatting & Collation**

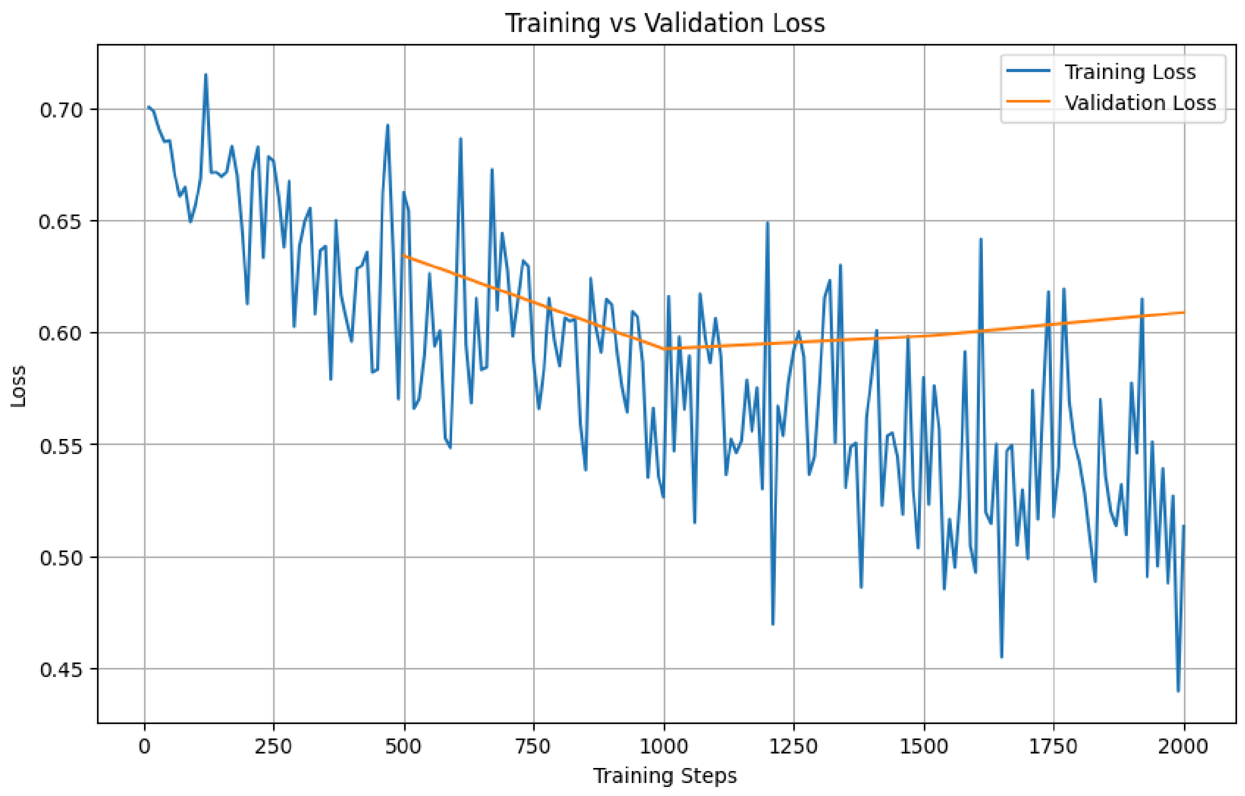
  We maintained padding/truncation to length 256, tensor formatting, and dynamic padding.
- **Training Arguments**

  We set learning_rate=3e-6, reduced batch_size to 16, increased epochs to 15, and kept weight_decay=0.1 with early stopping (patience=2).
- **Trainer Initialization**

  We applied the same early-stopping callback to guard against overfitting.
- **Model Training**

  We executed trainer.train() and monitored progress.
- **Visualization**

  We produced a final **Training vs Validation Loss** plot to verify stability at convergence.



This model did not perform better than the previous one hence we assume it was the best (at the global minima).

Modeling conclusion; deberta_base_hatespeech_5 was the best model hence it is saved for the next step ie Deployment

Conclusions

DeBERTa is the Best Performing Model

- Among all models tested, DeBERTa demonstrated the best performance across key evaluation metrics.
- It achieved higher accuracy and F1 score on the validation set, showing strong generalization capabilities.
- The confusion matrix for DeBERTa showed fewer misclassifications, especially in correctly identifying hate speech instances.
- The model exhibited a lower risk of overfitting, as the training and validation scores were closely aligned.
- In contrast, the Logistic Classifier underperformed, especially in detecting hate tweets, with a high number of false negatives, indicating possible underfitting or lack of optimization for complex patterns in the data.

Time-of-Day Tweet Patterns

- The analysis revealed that 10 AM to 2 PM is the most active period for both Hate and Not_Hate tweets.
- This midday window can be strategically targeted for content moderation, awareness campaigns, or counter-speech efforts.
- There is also a notable resurgence of hate tweets during evening hours (7–9 PM), which may be linked to:
- Reactions to daily news or events.
- Increased user activity after work hours, leading to more emotionally charged content.

Yearly Tweet Trends

- Tweet volume and engagement appear to follow a cyclical trend, with clear spikes in certain years.
- Peaks in activity likely correspond to major socio-political events, such as elections or national controversies.

- For instance, a sharp declines from 2018→2019 and 2024→2025 suggest that high engagement is often event-driven, and public discourse tapers off after major incidents.

Politician Engagement Scores

- High Engagement Politicians: Betty Maina and John Kiarie stand out with average engagement scores above 10,000, suggesting their tweets gained significant traction. This may be due to:
- Viral content.
- Involvement in trending issues.
- Public reactions to controversial statements.
- Moderate Engagement Group: Alice Nganga, Noordin Haji, and Didmus Barasa consistently engaged users, with scores between 1,500–2,500, indicating steady but less viral presence.
- Lower Engagement Politicians: Figures like Martha Koome, Japheth Koome, Kalonzo Musyoka, and Johnson Sakaja saw lower interaction levels, possibly due to a less active online presence or lower public interest in their tweets.
- The presence of tweets labeled as "unknown" suggests that some posts could not be reliably linked to a specific politician but still garnered moderate engagement.

Most Discussed Politicians

- William Ruto leads by a large margin with over 1,500 tweets, of which 45% contain hate speech, pointing to a highly polarized public sentiment.
- Raila Odinga ranks second in volume (540 tweets), showing a more balanced mix of hate and neutral content, though negative sentiment remains notable.
- Oscar Sudi and Rigathi Gachagua have a high proportion of hate tweets (over 50%) despite fewer total mentions, indicating a strong backlash or controversial public image.
- Fred Matiang'i and Kithure Kindiki receive moderate attention, with sustained engagement though not as much negativity as others.
- Rachel Ruto and Martha Koome are among the least mentioned, reflecting lower public focus or reduced visibility in the current discourse.

Recommendations

Enhance Content Moderation During Peak Hours

- Deploy automated moderation tools and increase human oversight between 10 AM–2 PM and 7–9 PM, when hate speech activity peaks.
- Platforms can prioritize review queues and flag high-risk content during these windows for faster intervention.

Invest in Advanced NLP Models for Detection

- Models like DeBERTa, which showed strong performance, should be adopted or integrated into moderation pipelines.
- Regular retraining on updated datasets is recommended to keep up with evolving language and hate speech patterns.

Improve Labeling and Data Quality Processes

- Address inconsistencies found in the labeled dataset by introducing automated language detection, better annotation guidelines, and manual review steps.
- Tools like Excel should be used cautiously for labeling; prefer annotation platforms designed for NLP tasks to avoid structural errors.

Implement Targeted Counter-Speech Campaigns

- Design and launch awareness or counter-hate campaigns during high-traffic hours to combat negative narratives in real-time.
- Leverage influential figures and community leaders to participate during these critical hours.

Monitor Politician-Specific Engagement Trends

- For politicians with high hate speech ratios (e.g., William Ruto, Oscar Sudi, Rigathi Gachagua), public relations teams should consider:
- Conducting sentiment audits.

- Addressing misinformation or controversial narratives through official communication.
- Politicians with moderate engagement can benefit from more active public interaction to boost positive visibility.

Utilize Yearly Trends for Strategic Planning

- Anticipate tweet volume spikes during election years or major national events, and scale moderation infrastructure accordingly.
- Preemptively prepare for discourse shifts by conducting sentiment and keyword analysis ahead of key political dates.

Strengthen User Education and Reporting Tools

- Educate users on what constitutes hate speech and how to report it effectively.
- Improve platform reporting tools to allow faster and more accurate flagging of harmful content, especially during peak periods.

Deployment

We structured our app.py as the central entry point for deployment by importing FastAPI's core classes (FastAPI, UploadFile, File) and response utilities (JSONResponse), configuring CORS via CORSMiddleware to allow cross‑domain clients, and loading our Hugging Face components (AutoTokenizer, AutoModelForSequenceClassification) alongside pandas and torch for preprocessing and inference. By instantiating app = FastAPI() at startup and initializing the tokenizer and model once, we ensured minimal per‑request latency and a clean foundation for defining RESTful endpoints that accept text or file inputs and return structured JSON classification results, ready to be hosted with Uvicorn.