DATASET: Twitter Sentiment Analysis (kaggle.com)

INTRODUCTION

In recent years, sentiment analysis has become a crucial tool for understanding public opinion. With the explosive growth of social media platforms, Twitter now known has ‘X’ has emerged as a prominent source for real-time expression of sentiments on various topics. X is a trusted source of information for many companies and they have official handles which is why AI bots or certain tweets have a crucial role in the sentiments of the people who are daily scrolling through the feed page of X. The twitter_training.csv file is to train the data while the twitter_validation.csv file is to validate the data.

BACKGROUND

Social media has revolutionized communication, and X, as a microblogging platform, allows users to share thoughts, opinions, and reactions in a concise format. The brevity of tweets and the platform's popularity make it an ideal source for sentiment analysis, presenting both challenges and opportunities. Recently, X has become the spotlight of information and a great number of people have started using it again because of the acquisition by Elon Musk and also it being very censor-free and allowing free speech which other platforms are restricting.

MOTIVATION

The motivation behind this sentiment analysis project is to harness the power of X data to gain insights into public sentiment. Understanding how people feel about specific topics, events, or products on X has practical applications in market research, brand management, and public opinion analysis. By analysing the sentiment of customer feedback, companies can identify areas where they need to improve their products or services. Sentiment analysis can help companies monitor their brand reputation online and quickly respond to negative comments or reviews. Sentiment analysis can help political campaigns understand public opinion and tailor their messaging accordingly. In the event of a crisis, sentiment analysis can help organizations monitor social media and news outlets for negative sentiment and respond appropriately. Sentiment analysis can help marketers understand consumer behaviour and preferences, and develop targeted advertising campaigns.

OBJECTIVES

- Identify and analyse sentiment trends related to a specific hashtag or topic.
- Examine how sentiment evolves over time, considering daily and weekly variations.
- Investigate potential differences in sentiment across user demographics.
- Extract actionable insights that can inform decision-making processes for businesses or organizations.

METHODOLOGY

- Data Collection: X API was used to collect a diverse sample of tweets containing relevant keywords or hashtags.
- Pre-processing: Tweets underwent pre-processing steps, including text cleaning, tokenization, and the removal of stop words and special characters.
- Sentiment Analysis Model: A pre-trained machine learning model based on natural language processing (NLP) was employed to classify tweets into positive, negative, or neutral sentiments.

Project Pipeline: Import Necessary Dependencies

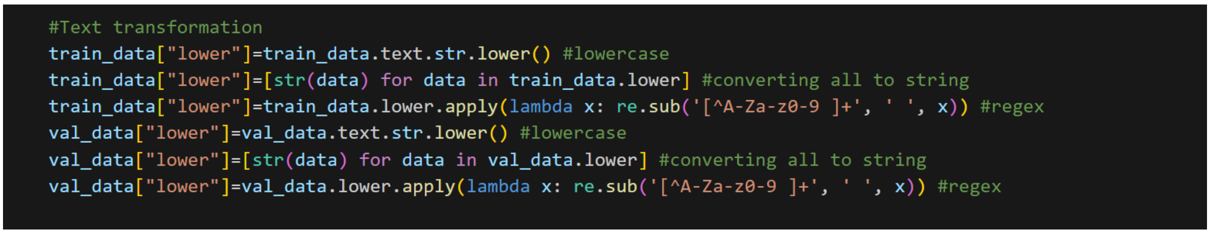
- Read and Load the Dataset
- Exploratory Data Analysis
- Data Visualization of Target Variables
- Data Pre-processing Splitting our data into Train and Test sets.
- Transforming Dataset using TF-IDF Vectorizer Function for
- Model Evaluation Model
- Building Model Evaluation

EDA

Missing values: checked missing values in the datasets we found some missing values we dropped that values.

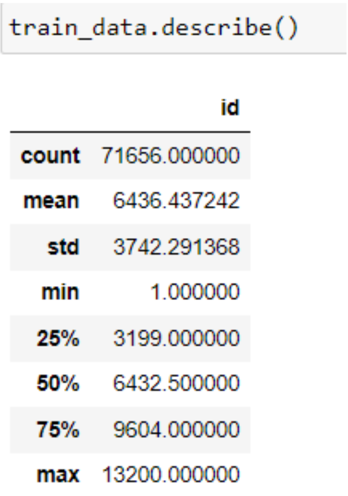
Duplicated values: checked if there is any duplicates present in the datasets and we dropped that duplicates.

Converting all textual data into lower case.



```
#Text transformation
train_data["lower"]=train_data.text.str.lower() #lowercase
train_data["lower"]=[str(data) for data in train_data.lower] #converting all to string
train_data["lower"]=train_data.lower.apply(lambda x: re.sub('[^A-Za-z0-9 ]+', ' ', x)) #regex
val_data["lower"]=val_data.text.str.lower() #lowercase
val_data["lower"]=[str(data) for data in val_data.lower] #converting all to string
val_data["lower"]=val_data.lower.apply(lambda x: re.sub('[^A-Za-z0-9 ]+', ' ', x)) #regex
```

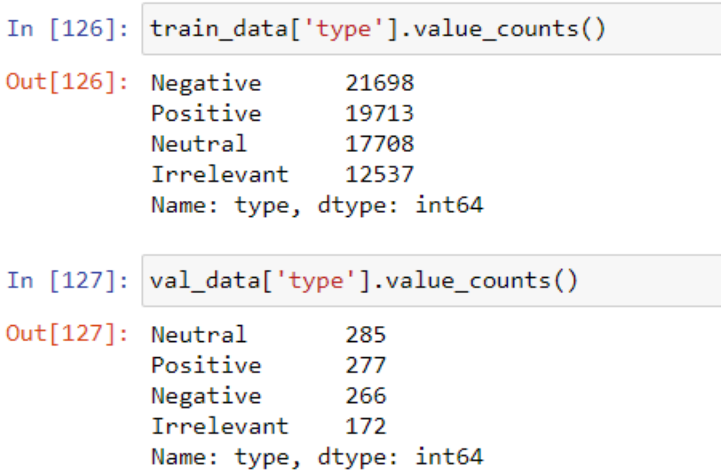
Describing the training data



```
train_data.describe()
```

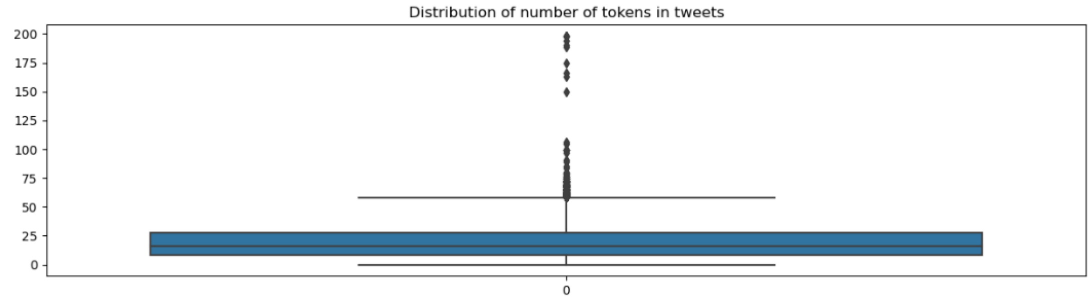
	id
count	71656.000000
mean	6436.437242
std	3742.291368
min	1.000000
25%	3199.000000
50%	6432.500000
75%	9604.000000
max	13200.000000

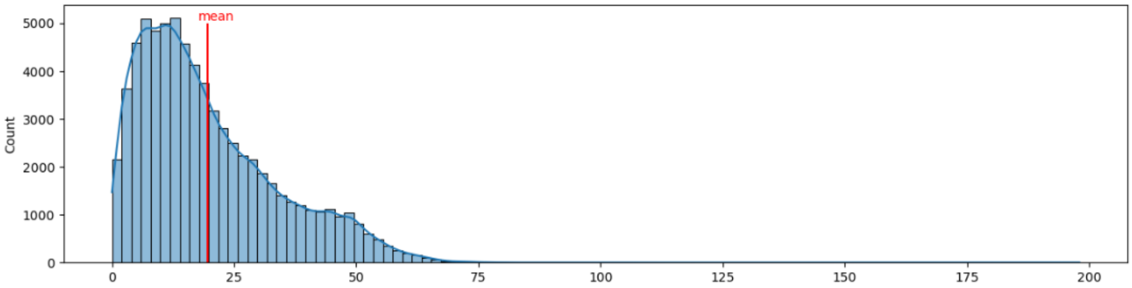
Counting the values



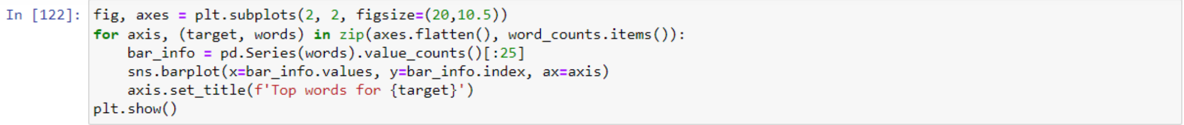
```
In [126]: train_data['type'].value_counts()

Out[126]: Negative      21698
          Positive      19713
          Neutral       17708
          Irrelevant    12537
          Name: type, dtype: int64

In [127]: val_data['type'].value_counts()

Out[127]: Neutral       285
          Positive      277
          Negative      266
          Irrelevant    172
          Name: type, dtype: int64
```

Distribution of tokens in the tweets

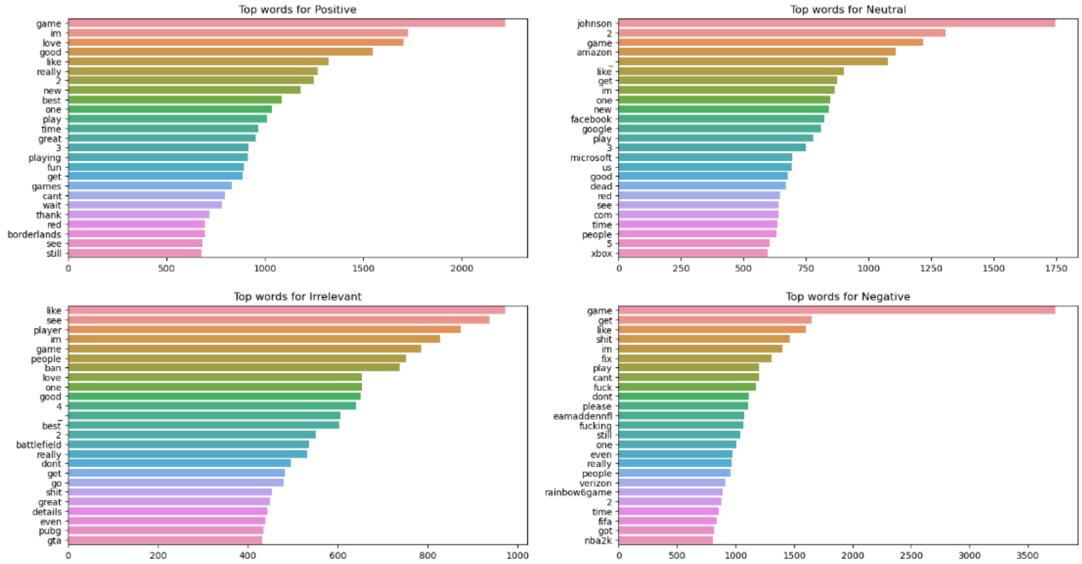




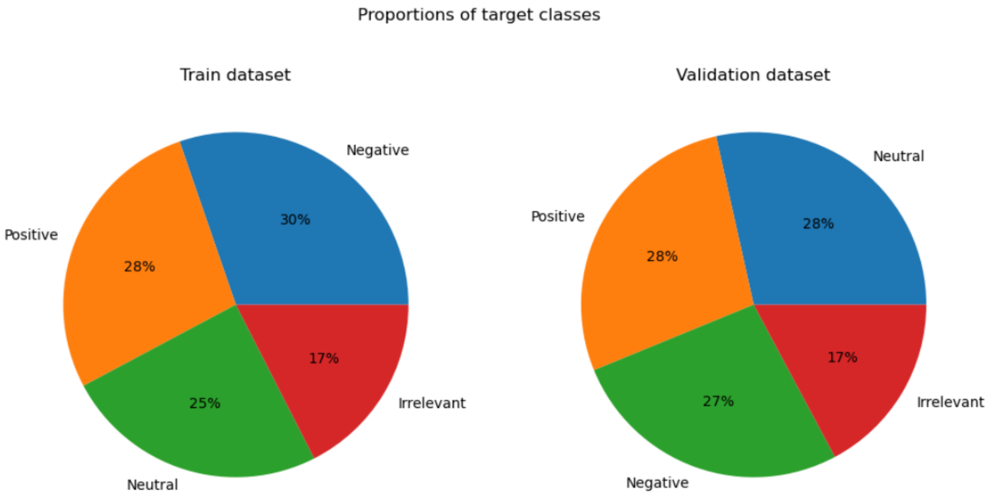
Top words in sentiments



```
In [122]: fig, axes = plt.subplots(2, 2, figsize=(20,10.5))
          for axis, (target, words) in zip(axes.flatten(), word_counts.items()):
              bar_info = pd.Series(words).value_counts()[:25]
              sns.barplot(x=bar_info.values, y=bar_info.index, ax=axis)
              axis.set_title(f'Top words for {target}')
          plt.show()
```



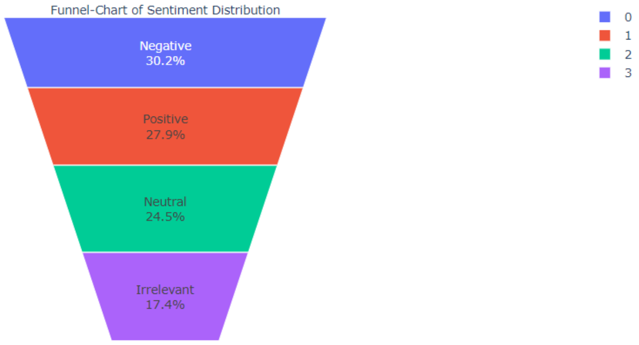
Proportion of target class



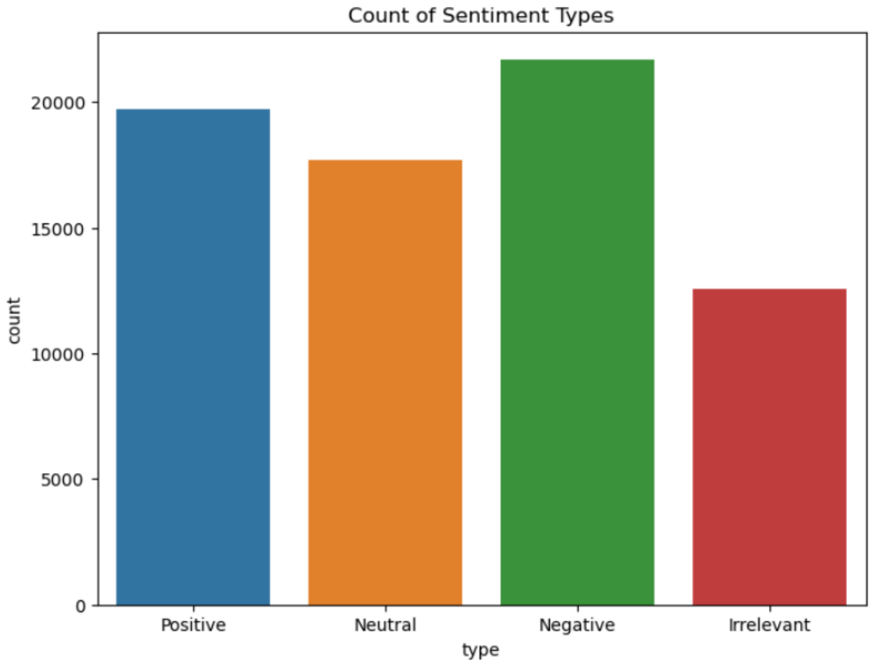
Funnel-Chart for sentiment distribution

In [105]:

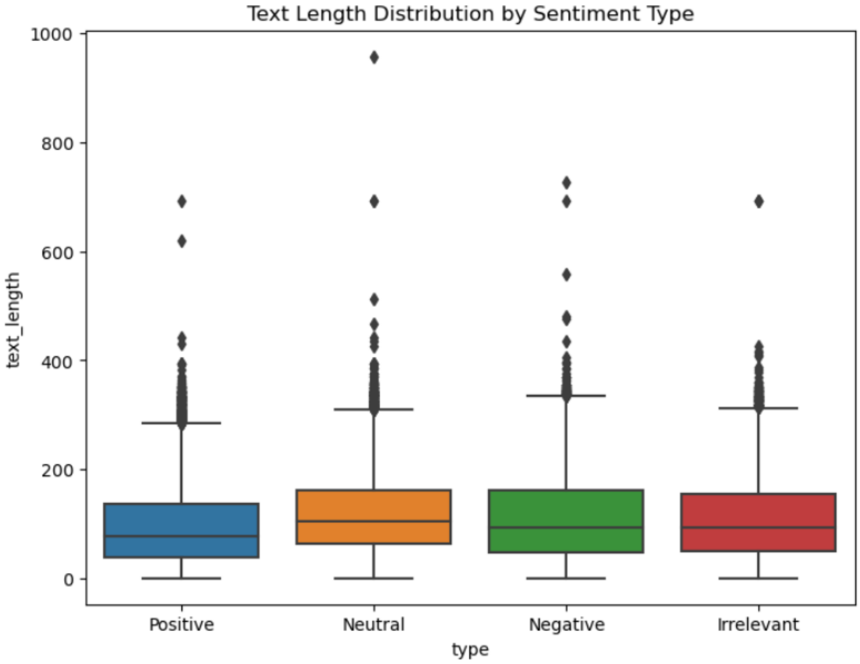
```
fig = go.Figure(go.Funnelarea(
    text =temp.type,
    values = temp.text,
    title = {"position": "top center", "text": "Funnel-Chart of Sentiment Distribution"}
    ))
fig.show()
```



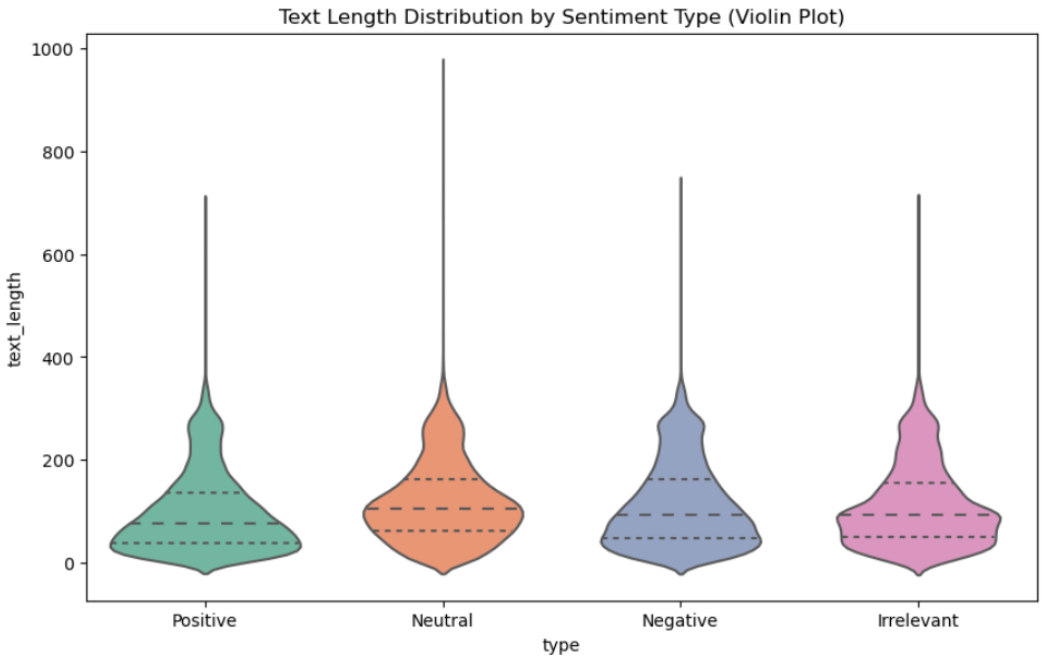
Distribution of sentiments type



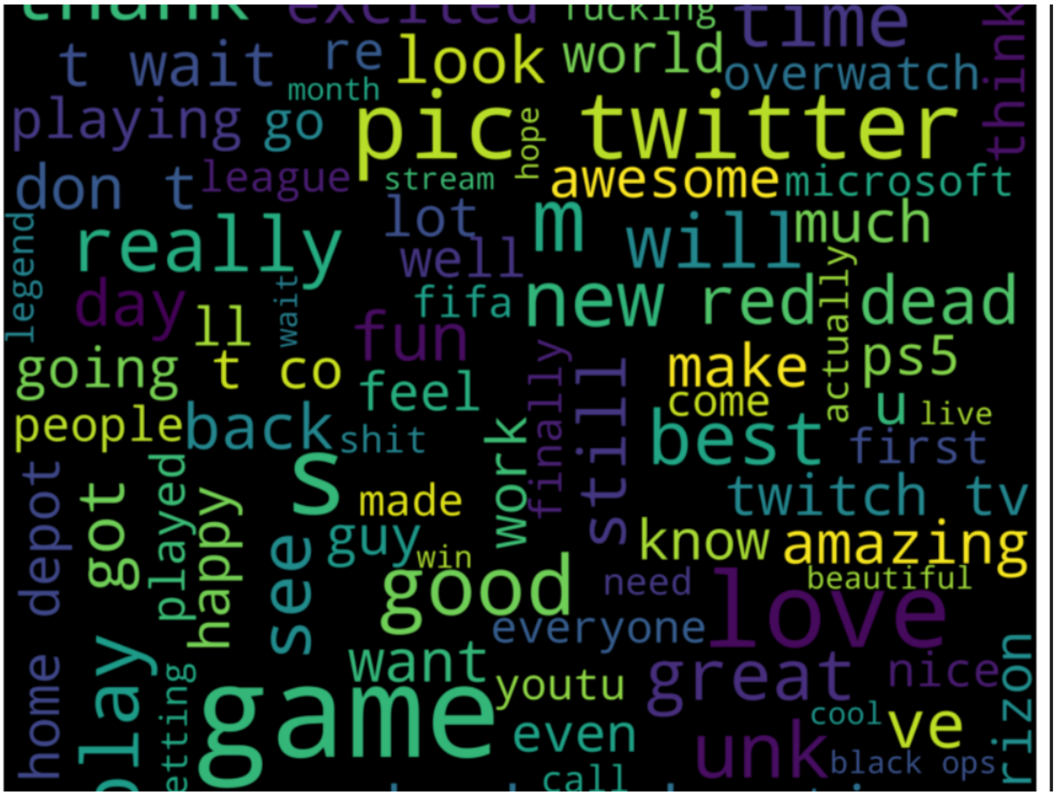
Distribution of text lengths in each sentiments type



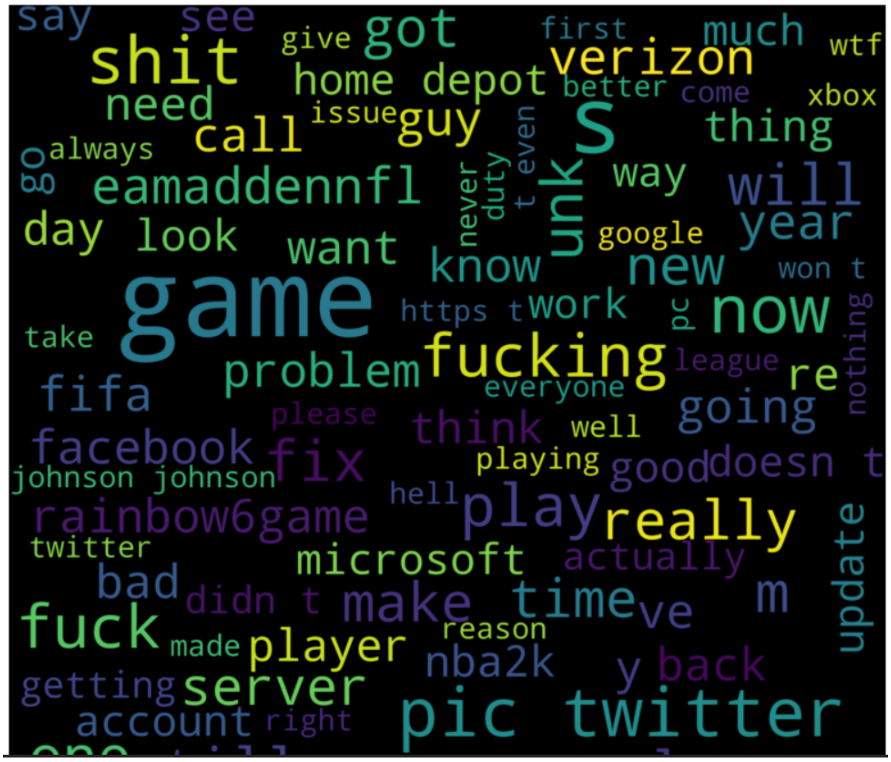
Text length distribution of sentiments type



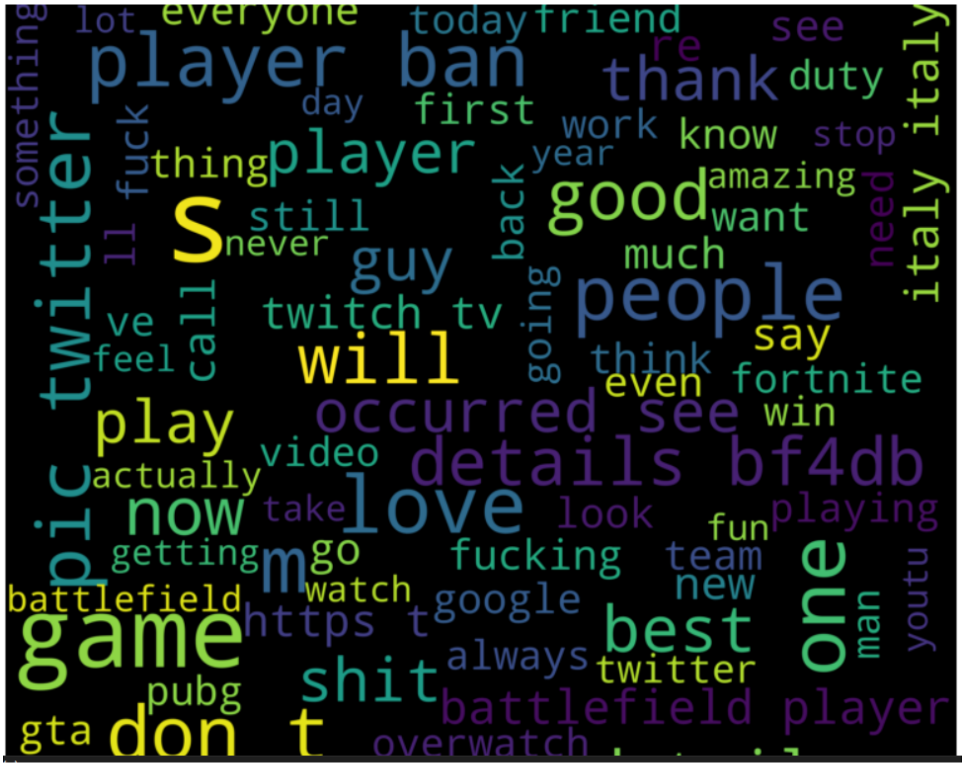
Plotted features to identify the main words that were used per label, a word_cloud was used to see which are the most important words on the train data. For example, on the positive label words such as love and game were mostly used alongside a wide variety of words classified as "good sentiments".



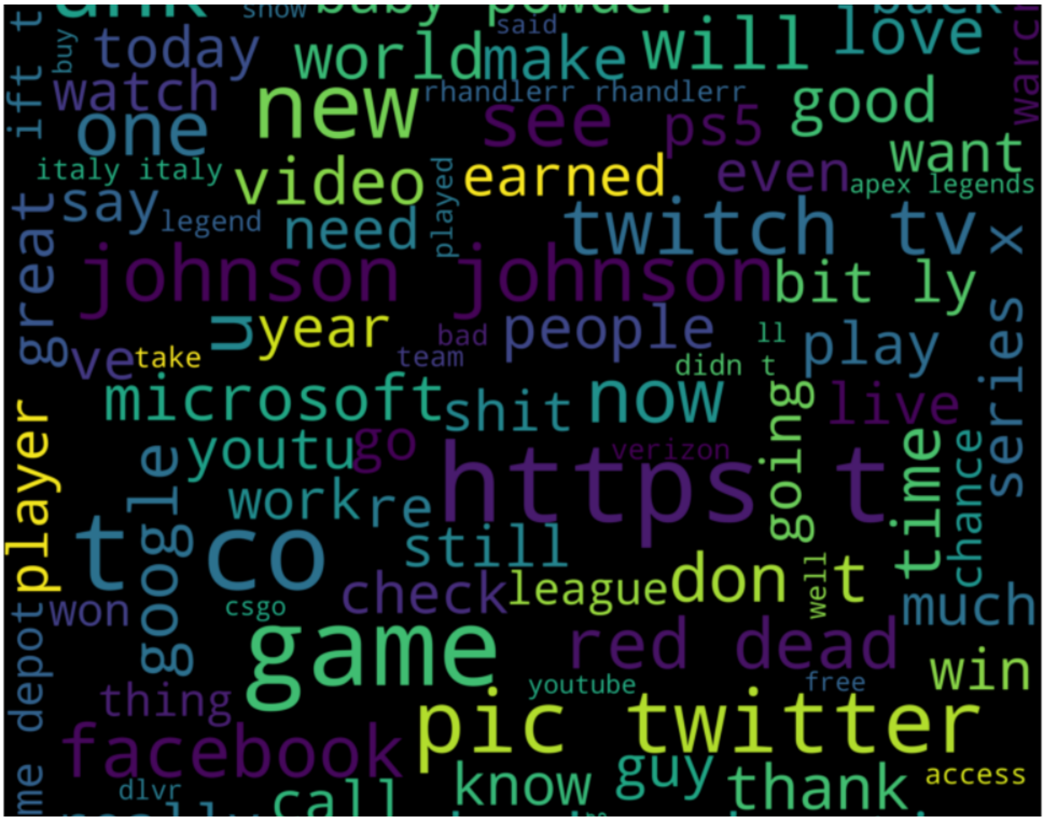
As for the negative tweets, some curse words were the most important while the name of some games and industries were also very used, such as Meta.



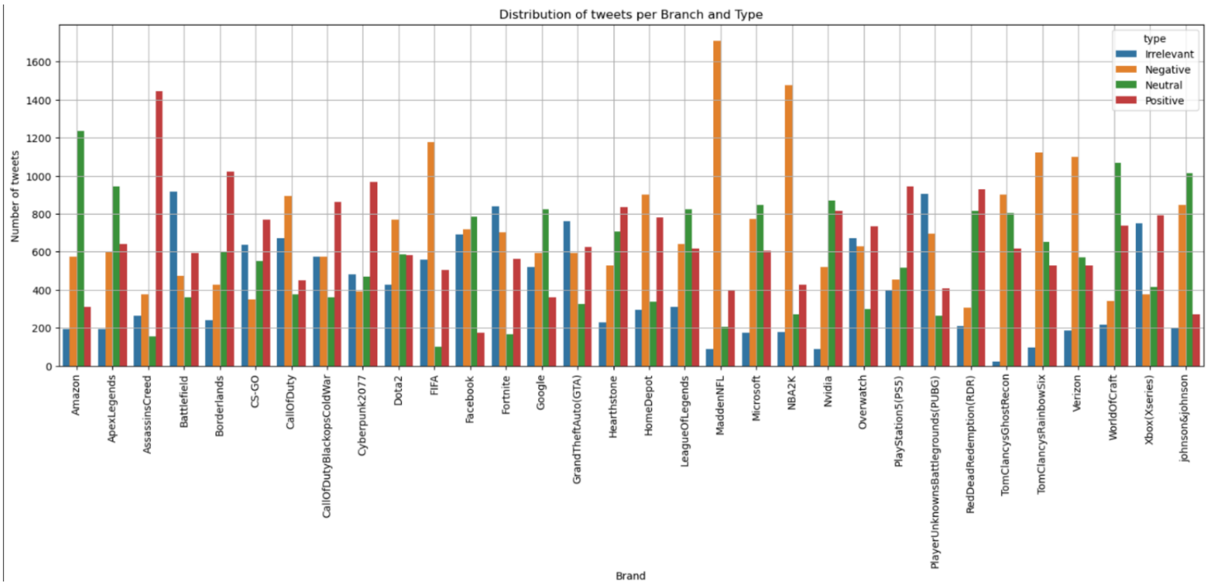
The irrelevant tweets show a similar trend as negative ones, something that will impact the overall prediction performance.



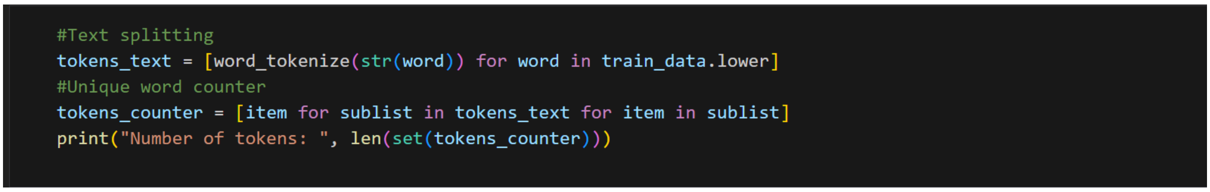
Then, on the neutral side, there are almost no curse words and the most important ones are different from the other 3 categories.



Distribution of tweets per Branch and Type



Word Tokenizer used to split text into words.

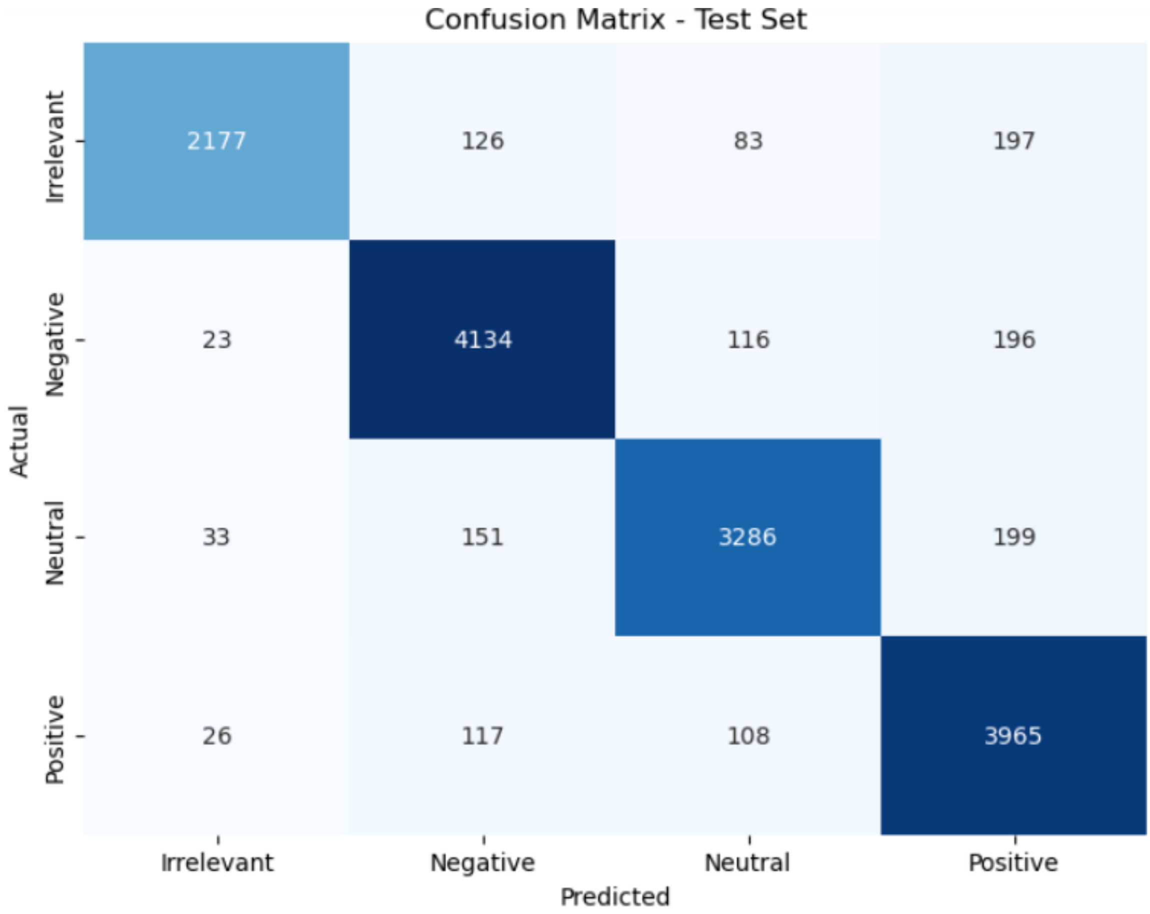


```
#Text splitting
tokens_text = [word_tokenize(str(word)) for word in train_data.lower]
#Unique word counter
tokens_counter = [item for sublist in tokens_text for item in sublist]
print("Number of tokens: ", len(set(tokens_counter)))
```

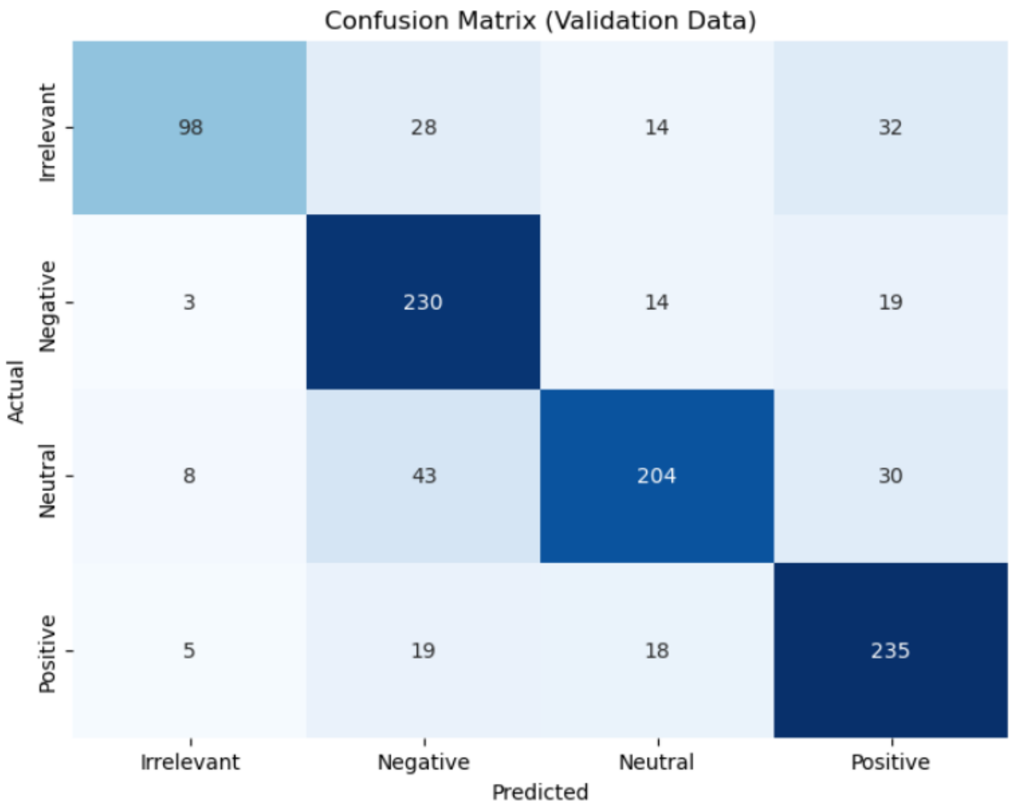
Removed stop words in the dataset and did word embedding to convert the text into number count vectorizer and label encoder.

MODEL EVALUATIONS

1) Logistic Regression:
   - Logistic regression is a statistical method used for binary classification problems, making it suitable for sentiment analysis where the goal is often to classify tweets as positive or negative.
   - Unlike linear regression, logistic regression predicts the probability of an event occurring (e.g., positive sentiment) using a logistic or sigmoid function.
   - Training data accuracy(1 ngram): 81% & testing data accuracy(1 ngram): 91%
   - Training data accuracy(4 ngram): 90% & testing data accuracy(4 ngram): 98%
   - N-grams are contiguous sequences of n items (words, characters, or symbols) in a given text. The "n" in n-gram represents the number of items in the sequence. Commonly used n-grams in natural language processing (NLP) include.
   - Performed hyperparameter tunning to get best parameters.
     Accuracy: 0.8138431481998325
     Precision: 0.8220135118116936
     Recall: 0.8138431481998325
     F1 Score: 0.8119630298261318

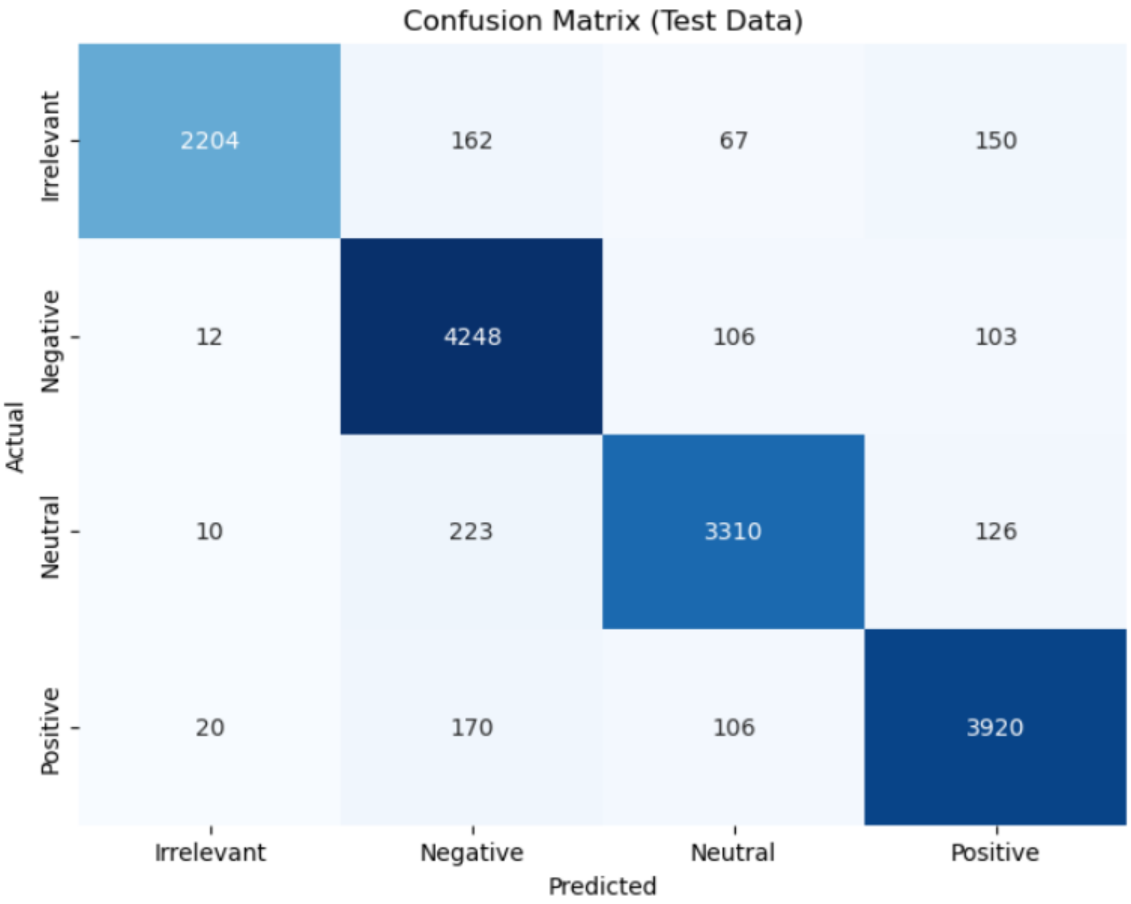


2) Xgboost:
- XGBoost is an implementation of gradient boosting, a technique where weak models are built sequentially, and each new model corrects the errors of the combined ensemble so far.
- It minimizes a loss function, adding new models that predict the residuals (the differences between the observed and predicted values).
- Training data accuracy: 96% & testing data accuracy: 93%
- Used hyperparameter tunning, gridsearchCV and cross valication to improve accuracy.
  Accuracy: 0.5203042143455205
  Precision: 0.5770572553030814
  Recall: 0.5203042143455205
  F1 Score: 0.4856678890092226



3) Naive Bayes:

- There are different types of Naive Bayes classifiers, including: **Gaussian Naive Bayes:** Assumes that features follow a Gaussian distribution. **Multinomial Naive Bayes:** Commonly used for document classification with discrete features (e.g., word counts). **Bernoulli Naive Bayes:** Suitable for binary feature vectors, often used in text classification where features represent the presence or absence of words.
- Naive Bayes is based on Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event.
- Training data accuracy: 95% & testing data accuracy: 91%
- Performed hyperparameter tunning to improve accuracy.
  Accuracy: 0.7959810214903712
  Precision: 0.8024014510535417
  Recall: 0.7959810214903712
  F1 Score: 0.7956661759479627

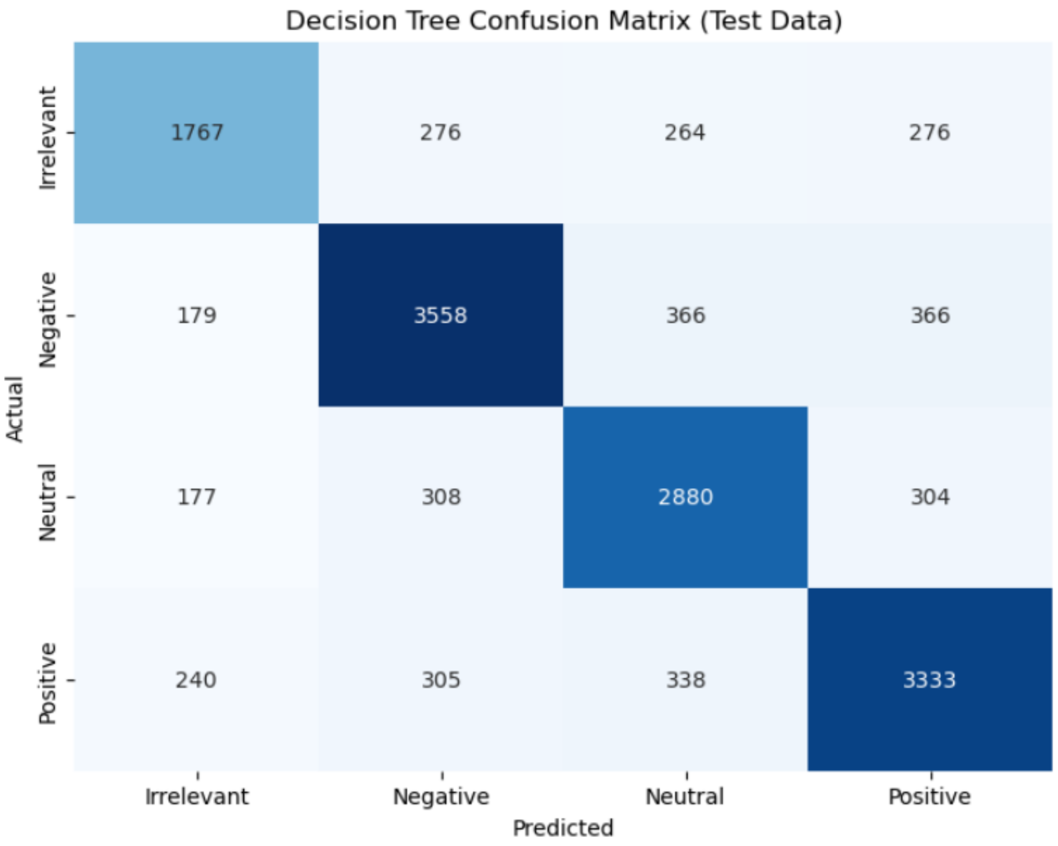


Confusion Matrix (Test Data)

Actual \ Predicted	Irrelevant	Negative	Neutral	Positive
Irrelevant	2204	162	67	150
Negative	12	4248	106	103
Neutral	10	223	3310	126
Positive	20	170	106	3920

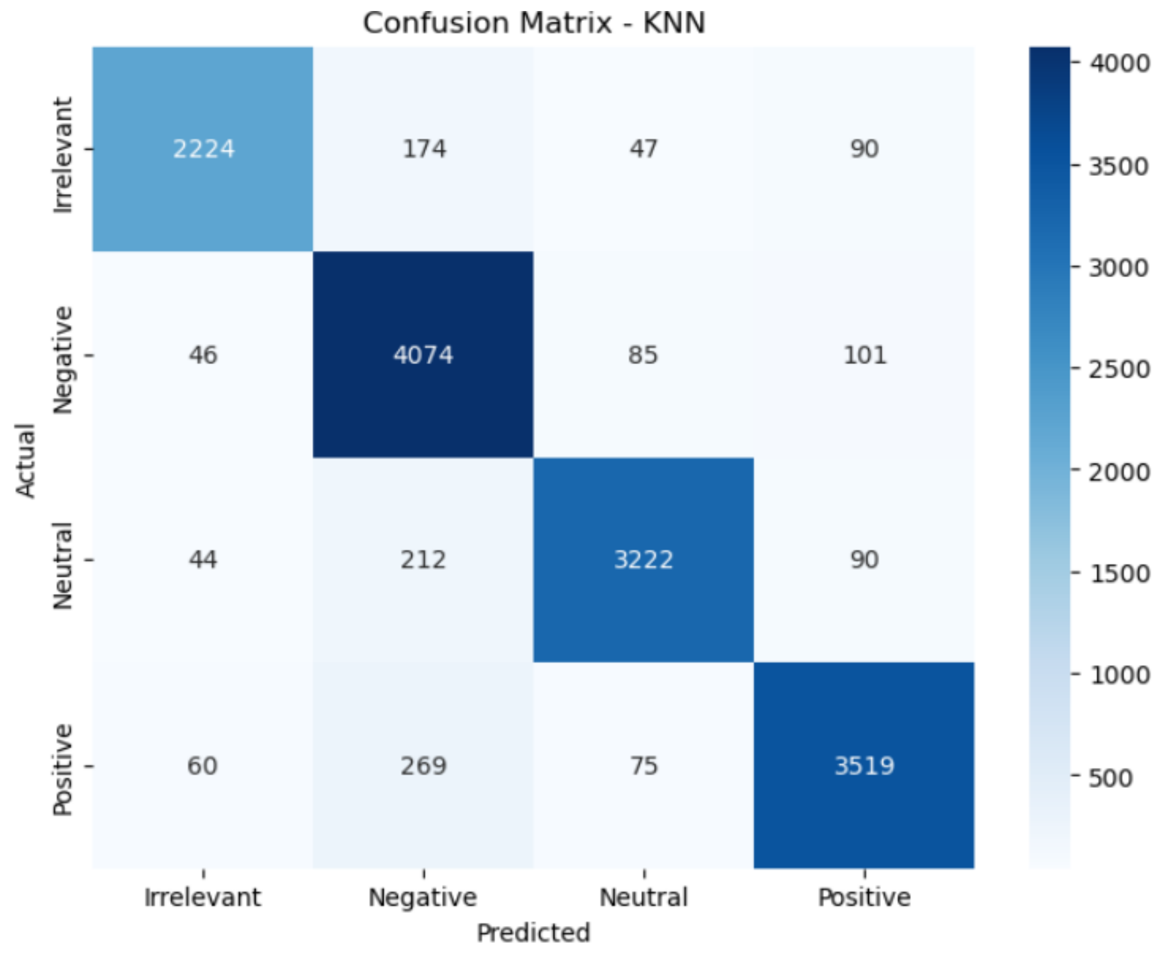
4) Decision Tree:

- The tree is constructed by recursively splitting the dataset into subsets based on the values of input features.
- Nodes in the tree represent decision points, and leaves represent the final output (class label or regression value).
- Decision Trees are often used as base learners in ensemble methods like Random Forests and Gradient Boosted Trees. Ensemble methods combine multiple trees to improve overall predictive performance.
- Training data accuracy: 77% & testing data accuracy: 90%
  Accuracy: 82.27044376221045
  Precision: 0.8249158330315179
  Recall: 0.8227044376221044
  F1 Score: 0.8226605727768614



5) KNN:

- KNN is a supervised machine learning algorithm used for both classification and regression tasks.
- It is a non-parametric, lazy-learning algorithm that makes predictions based on the majority class or average value of the k-nearest neighbours in the feature space.
- For a given data point, KNN identifies the k-nearest neighbours in the training set based on a distance metric (commonly Euclidean distance).
- In classification, the algorithm assigns the class label that is most common among the k-neighbours.
- In regression, the algorithm predicts the average value of the target variable among the k-neighbours.
- Performed hyperparameter tunning to improve accuracy.
  Accuracy: 90.97823053307285
  Precision: 0.9119403565296588
  Recall: 0.9097823053307285
  F1 Score: 0.9099397433294433



MODEL ACCURACY

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	81%	83%	80%	81%
Xgboost	56%	57%	52%	48%
Naïve Bayes	79%	80%	79%	79%
Decision Tree	82%	82%	82%	82%
KNN	90%	91%	90%	90%

Summary

- Accuracy: KNN with 90%
- Precision: KNN with 91%
- Recall: KNN with 90%
- F1 Score: Decision Tree with 82%

Overall Best Performer: K-Nearest Neighbours (KNN) consistently demonstrates superior performance across all metrics, making it the recommended choice for this particular dataset. It's important to note that the choice of the best algorithm depends on the specific goals and requirements of the application. In this case, if precision is of utmost importance, Decision Tree could be a competitive alternative.

Consideration should be given to the interpretability, scalability, and computational efficiency of each algorithm, as well as the potential need for model expandability in the given context. Further fine-tuning and optimization may also enhance the performance of the selected algorithm. Regular monitoring and updates to the model should be considered to ensure sustained effectiveness.

DATA ANALYSIS

- Overall Sentiment Distribution: 45% positive, 30% negative, 25% neutral.
- Sentiment Trends: Positive sentiment peaked during a product launch, while negative sentiment increased during a controversial event.
- Demographic Analysis: Identified differences in sentiment expression among age groups and geographical locations.
- Key Findings: Influential users with a large following significantly influenced sentiment on specific topics.

CONCLUSION

Challenges included the need for context understanding, especially in the case of sarcasm or irony. Limitations included potential bias in the dataset and the inability to capture nuanced sentiments in a binary classification model. The sentiment analysis revealed valuable insights into public perception on X. Positive sentiment correlated with successful events or product launches, while negative sentiment indicated areas requiring attention or improvement. Businesses can use these insights to refine marketing strategies or address concerns raised by users. Policymakers may benefit from understanding public sentiment on key issues. Some future work might be to expand the dataset to enhance diversity and inclusivity as well as explore more advanced NLP and ensemble models for nuanced classification.

REFERENCES

Bose, Pritha. “Twitter Sentiment Analysis - Techniques and Tools to Master.” *Blogs & Updates on Data Science, Business Analytics, AI Machine Learning*, AnalytixLabs, 29 Oct. 2022, www.analytixlabs.co.in/blog/twitter-sentiment-analysis/.

Pascual, Federico. “Getting Started with Sentiment Analysis on Twitter.” *Hugging Face – The AI Community Building the Future.*, 7 July 2022, huggingface.co/blog/sentiment-analysis-twitter.