UNIVERSITÀ DI PISA

Department of Information Engineering
MSc in Artificial Intelligence and Data Engineering

Sentiment Analysis

Data Mining and Machine Learning Project

Ludovica Cocchella

Academic Year 2023/2024

Contents

1 Introduction

In the digital era we are immersed in, podcasts have become an increasingly prevalent form of entertainment and information. With millions of episodes available on a wide range of topics, the variety of voices and opinions makes the world of podcasts incredibly rich and diversified.

1.1 Project description

The primary objective of this project is to implement Sentiment Analysis on podcast reviews, utilizing classification techniques to categorize the reviews into positive, neutral, or negative sentiments. To achieve this goal, the project will systematically evaluate and compare various classifiers to identify the most effective one for the analysis.

Following the sentiment analysis phase, the project will embark on a Streaming Analysis—a dynamic approach that tracks the evolution of sentiments toward podcasts over time. The main aim is to continuously assess the efficacy of our classifier in adapting to changing trends and the evolving emotions expressed by listeners. This phase is crucial for assessing whether the classifier maintains its classification accuracy in the face of challenges posed by emerging trends and evolving listener sentiments.

In the event that new trends or substantial variations in user sentiments arise, it may indicate the necessity to retrain the classifier. This dynamic phase serves a multitude of goals, including acquiring valuable feedback on content quality. These insights will function as constructive guidance for content creators, providing them with the means to enhance their content and better align with the preferences of their audience.

In essence, this project goes beyond the mere implementation of Sentiment Analysis; it aims to establish a comprehensive methodology that contributes valuable insights to content creators. By fostering continuous improvement and adaptability, the project addresses the dynamic landscape of podcasting, ensuring creators are well-equipped to navigate and thrive in an ever-evolving podcasting ecosystem.

GitHub Link: https://github.com/LudovicaCi/PodcastSentimentAnalyzer.

2 Dataset

2.1 Dataset description

The dataset selected for the project is available at https://www.kaggle.com/datasets/thoughtvector/podcastreviews and includes 2 million podcast reviews related to 100,000 podcasts, updated monthly.

I focused my attention on the dataset containing the information about the reviews that users left on podcast.

The dataset is composed of the following attributes:

- podcast_id
- title
- content
- rating
- author_id
- created_at

2.2 Dataset cleaning

In this phase, I am addressing missing values by removing them, as their number is significantly lower compared to the total number of reviews. Therefore, their removal will not impact the overall quality of the final results.

Another operation performed is the removal of duplicate reviews. This step is crucial for ensuring the quality of the dataset in sentiment analysis. It guarantees an accurate representation of user opinions, contributes to maintaining consistent and reliable data, and enhances the effectiveness of sentiment prediction. Simultaneously, eliminating duplicates prevents accidental overlaps between training and test sets, preserving the integrity of the analysis.

After obtaining the dataset, several text preprocessing steps were applied to enhance the quality and consistency of the textual data. This process is crucial for cleaning and standardizing the text before further analysis, as in the context of a sentiment analysis project. The preprocessing step are the following:

- Emoticon Removal: Using regular expressions, emoticons present in the text are removed, eliminating elements like :), :(, etc.
- URL Removal: Strings containing URLs or web addresses are removed using regular expressions.

- HTML Tag Removal: Any HTML tags present in the text are removed to ensure a clean textual representation.
- Unicode Normalization: The `unicodedata.normalize` function is used to normalize unicode characters, removing any accents or non-ASCII special characters.
- Removal of Special Characters, Numbers, and Punctuation: Only alphabet letters and spaces are retained, removing special characters, numbers, and punctuation.
- Conversion to Lowercase: The text is entirely converted to lowercase to ensure standardization.
- Trimming Whitespaces: Leading and trailing whitespaces in the text are removed.
- Removal of Extra Spaces: Any multiple whitespaces within the text are reduced to a single space.

2.3 Data reduction

In the context of our project, the phase of data reduction plays a pivotal role. The primary aim is to simplify the complexity of the dataset. It is applied dimensionality reduction to select the most relevant features, reducing the number of variables. Simultaneously, numerosity reduction was employed to handle excessive amounts of data, maintaining a representative subset. In both cases, the goal is to enhance computational efficiency.

2.3.1 Dimensionality reduction

At this stage, attributes that are not necessary for our analysis have been removed. To make our sentiment analysis on the podcast' reviews I consider only two attributes:

- content: this attribute contains the text of the reviews used to build our classifier
- rating: this attribute is used to divide comments into three classes establishing a ground truth.

2.3.2 Numerosity reduction

The number of comments is very high and for this reason I decided to keep just a representation of them.

I analysed the distribution of comments over time, extracting this graph:

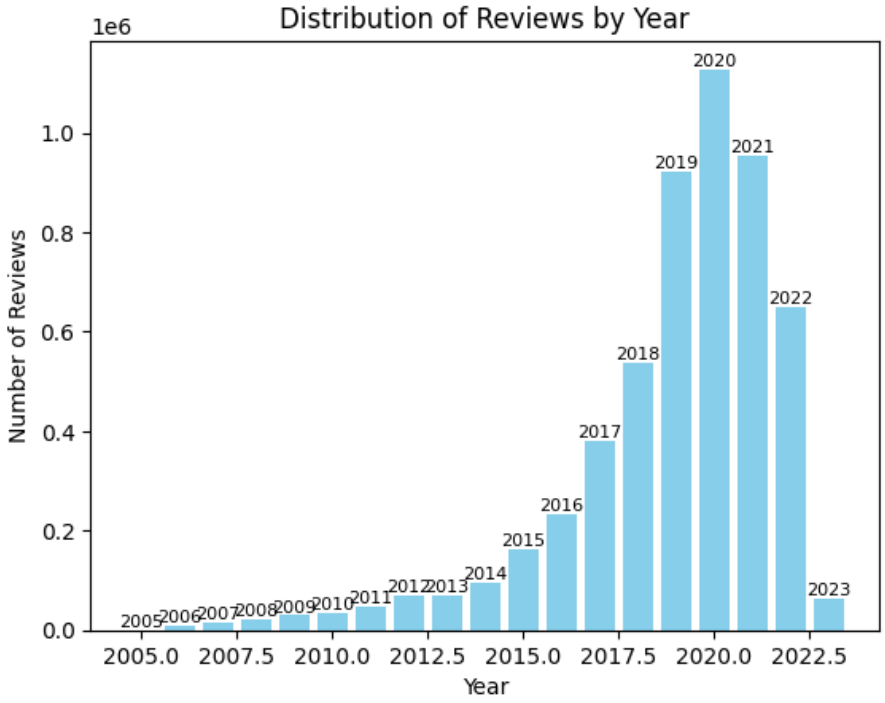

Figure 1: Training set distribution.

The comments range from the year 2005 to 2023. Only the reviews corresponding to the year 2020 are utilized, as it had the highest volume. Initially, I had 5,430,620 reviews, but by considering only those related to the year 2020, I obtain 1,127,656 reviews.

2.4 Training set building

For the training set, a dataset composed of 79,236 reviews was built and labeled with three different classes using a ground truth:

- Positive reviews: their rating is 5.
- Neutral reviews: their rating is 3.
- Negative reviews: their rating is 1.

I decided to have a balanced training set composed of 26.412 reviews from each class. I removed all the comments rated 0 (not rated comments) and comments rated 2 or 4 (they don't have a well-defined class).

The distribution of the training set is shown below:

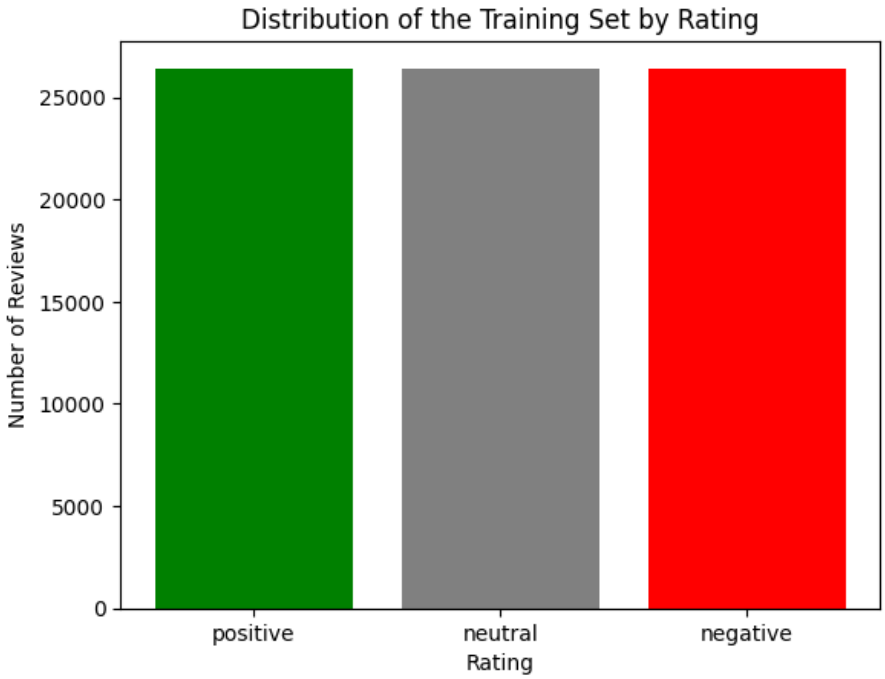

Figure 2: Training set distribution.

3 Classifiers evaluation

In this phase, I have implemented various classifiers with the aim of finding the one that best succeeds in correctly classifying sentiment. This iterative process involves testing and evaluating different algorithms to identify the most effective one for sentiment analysis.

The steps undertaken in this phase are as follows:

1. **Text Preprocessing:** In this initial step, the text of the reviews is preprocessed using techniques such as tokenization, stopwords removal, and stemming to enhance the text representation for subsequent analysis.

2. **Implementation of Classifiers:** The second step involves the implementation of various classifiers, including the Multinomial Naive Bayes Classifier, the Decision Tree Classifier, the Linear Support Vector Classifier, the K-Nearest Neighbors (KNN) Classifier, and the Random Forest Classifier. Each classifier is carefully examined to understand its performance in the task of sentiment classification for podcast reviews.

3. **T-Test for Model Performance Assessment:** In the third step, a t-test is conducted to statistically compare the performance of different classifiers and identify which one excels in sentiment classification. This analysis provides crucial insights for selecting the optimal classifier tailored to the specific requirements of the analysis context.

3.1 Text Preprocessing

In this section, essential data preparation steps are performed before training and evaluating the machine learning models.

In the preprocessing pipeline, a four-step approach was implemented to transform the provided comments into a Bag of Words (BOW) representation. It's important to note that the Bag of Words representation is obtained using the `CountVectorizer` function from the `scikit-learn` (sklearn) library.

To implement the text preprocessing, a custom analyzer has been defined. The custom analyzer specifies how the text should be analyzed, tokenized, and processed before constructing the numerical representation of words.

The custom analyzer encompasses the following steps:

1. **Remove Accents and Punctuation:** The function starts by removing accents from characters and eliminating any punctuation marks. This ensures a cleaner and standardized representation of the text.

2. **Tokenization and Lowercasing:** This is the process of breaking down text into smaller units called tokens and each token is converted to lowercase. This step ensures consistency and simplifies subsequent processing.

3. **Remove Stopwords:** Common English stopwords, such as 'and,' 'the,' and 'is,' are removed from the tokenized words. Stopwords are often irrelevant in the context of certain analyses and can be excluded to focus on more meaningful content.

4. **Stemming:** The remaining words undergo stemming, a process of reducing words to their root or base form. For example, "running", "runs", and "ran" can be stemmed to "run". Stemming helps to reduce the number of words in text and simplify the vocabulary.

When applied to the text, `CountVectorizer` uses `custom_analyzer` to process and tokenize the text before assigning a numerical value to each word based on its frequency of occurrence in the documents. This numerical representation enables machine learning models to work with the text efficiently.

Next, the `TfidfTransformer` is applied, a transformer that converts the count matrix generated by `CountVectorizer` into a TF-IDF representation. TF-IDF (Term Frequency-Inverse Document Frequency) is a metric that weights words based on their frequency in the document and rarity across the entire corpus. This step is useful for assigning greater importance to less common words and reducing the importance of common words.

After these steps, it is possible to proceed with the construction of the classifier.

3.2 Multinomial Naive Bayes

A Multinomial Naive Bayes classifier has been implemented by employing a pipeline that incorporates various text preprocessing steps. The pipeline encompasses a custom text analyzer, CountVectorizer for feature extraction, TfidfTransformer for term frequency-inverse document frequency transformation, and the Multinomial Naive Bayes classifier. The evaluation is performed using cross-validation to ensure the robustness of the results.

A series of experiments were conducted to find the parameters that yielded the best results. The metrics obtained are the following:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.70	0.66	0.68	26270
Neutral	0.65	0.67	0.66	26216
Positive	0.82	0.84	0.83	26265
Accuracy			0.72	78751
Macro Avg	0.72	0.72	0.72	78751
Weighted Avg	0.72	0.72	0.72	78751

Table 1: Classification Metrics

The corresponding confusion matrix is showed below:

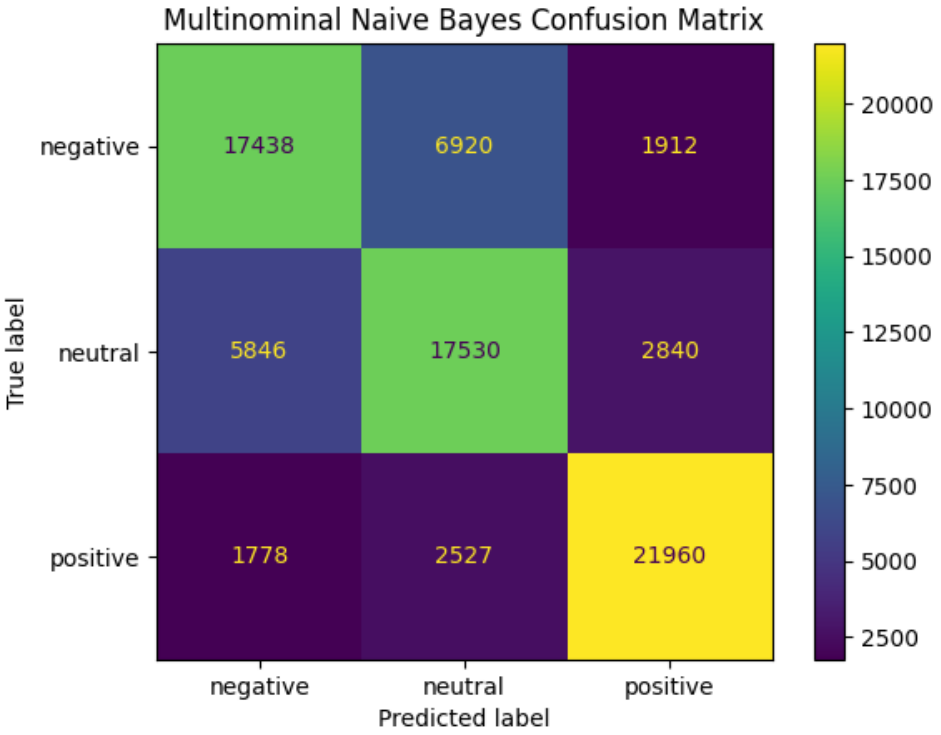

Figure 3: Confusion matrix

3.3 Decision Tree

A Decision Tree classifier was implemented using a pipeline that incorporates various text preprocessing steps. The pipeline includes a custom text analyzer, CountVectorizer for feature extraction (constrained to 3000 features), TfidfTransformer for term frequency-inverse document frequency transformation, feature selection using SelectKBest with a chi-squared test, and a Decision Tree classifier with a maximum number of leaf nodes set to 30. Despite multiple trials exploring different values of k in feature selection, no significant improvements were observed. As a result, the default value of k was retained. Evaluation was conducted through cross-validation to ensure model robustness.

The metrics obtained are the following:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.46	0.79	0.58	26270
Neutral	0.51	0.21	0.30	26216
Positive	0.66	0.58	0.62	26265
Accuracy			0.53	78751
Macro Avg	0.55	0.53	0.50	78751
Weighted Avg	0.55	0.53	0.50	78751

Table 2: Classification Metrics

The corresponding confusion matrix is showed below:

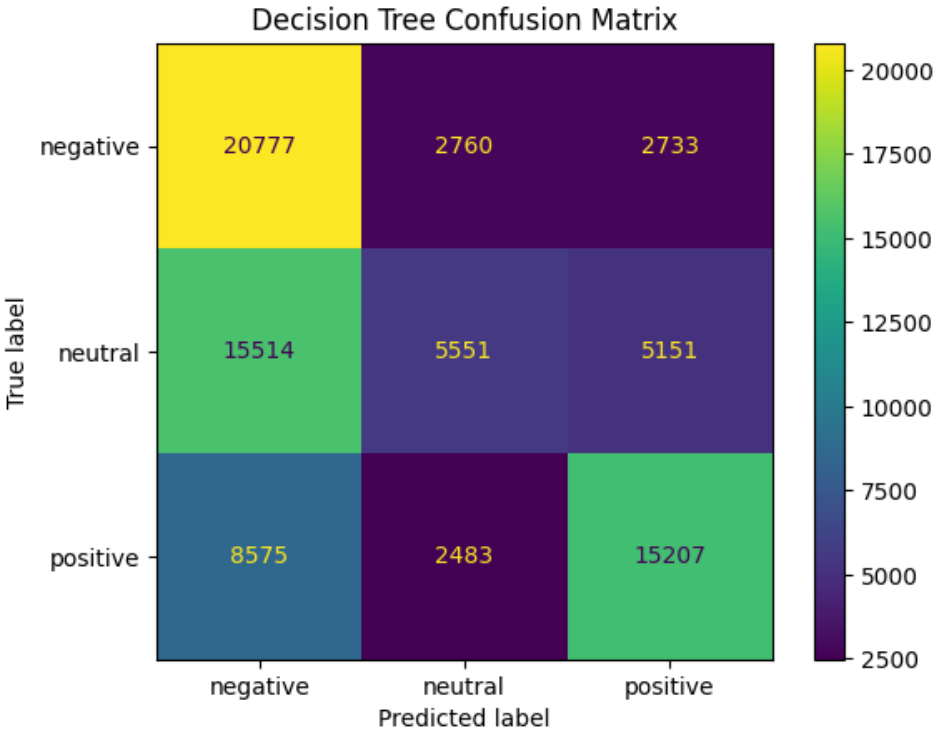

Figure 4: Confusion matrix

3.4 Linear Support Vector Classifier (SVC)

The performance of a Linear Support Vector Classifier (Linear SVC) is assessed through a text preprocessing pipeline. The pipeline, featuring a custom text analyzer, utilizes CountVectorizer for feature extraction (limited to 3000 features), employs TfidfTransformer for TF-IDF transformation, and integrates the Linear SVC classifier with a regularization parameter (C) set to 0.01. Performance evaluation is conducted through the cross-validation technique.

A series of experiments were conducted to find the parameters that yielded the best results. The metrics obtained are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.66	0.68	26270
Neutral	0.65	0.67	0.66	26216
Positive	0.82	0.84	0.83	26265
Accuracy			0.72	78751
Macro Avg	0.72	0.72	0.72	78751
Weighted Avg	0.72	0.72	0.72	78751

Table 3: Classification Metrics

The corresponding confusion matrix is showed below:

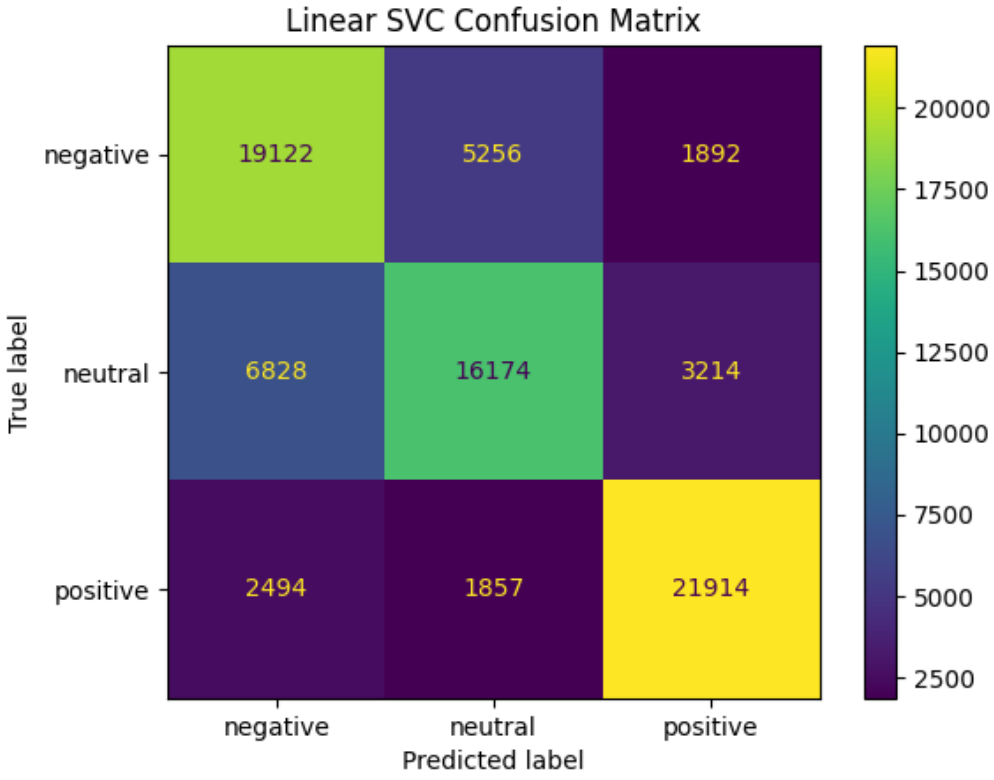

Figure 5: Confusion matrix

3.5 K-Nearest Neighbors (KNN) Classifier

The classifier utilizes the K-Nearest Neighbors (KNN) algorithm within a machine learning pipeline. The pipeline incorporates feature extraction with `CountVectorizer`, transformation with `TfidfTransformer`, and the deployment of the KNN classifier with specific parameters, including the number of neighbors (`n_neighbors`) set to 280 ($k = \sqrt{n}$ where n stands for the number of samples in the training set.), distance-based weighting (`weights='distance'`), and parallel processing utilization (`n_jobs=-1`). To assess the model's performance, a 10-fold cross-validation was conducted.

A series of experiments were conducted to find the parameters that yielded the best results. The metrics obtained are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.02	0.03	26270
Neutral	0.57	0.01	0.01	26216
Positive	0.34	0.99	0.50	26265
Accuracy			0.34	78751
Macro Avg	0.54	0.34	0.18	78751
Weighted Avg	0.54	0.34	0.18	78751

Table 4: Classification Metrics

The corresponding confusion matrix is showed below:

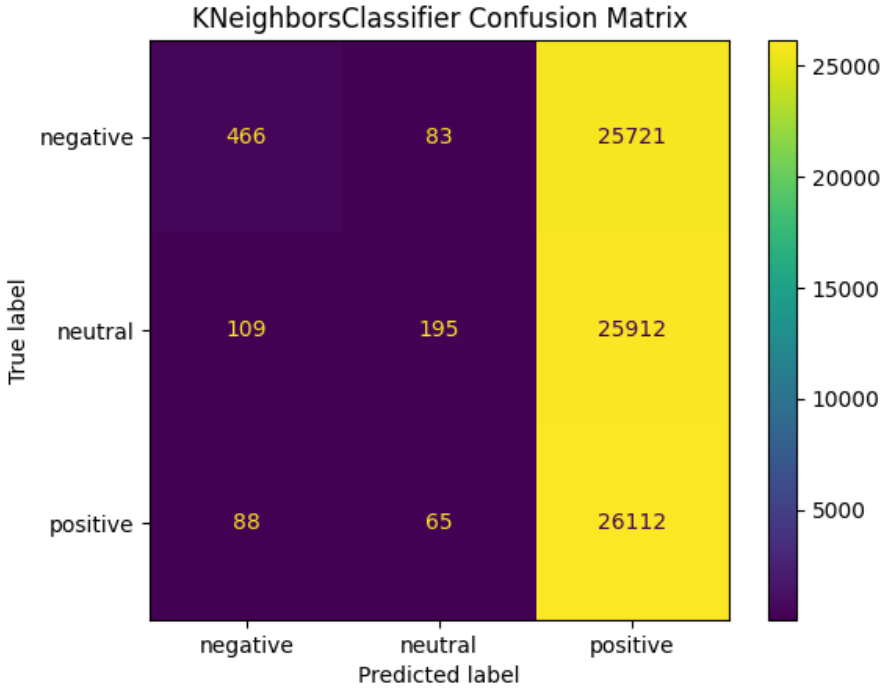

Figure 6: Confusion matrix

3.6 Random Forest Classifier

The classifier under consideration is based on a Random Forest algorithm within a machine learning pipeline. The pipeline includes feature extraction using CountVectorizer, transformation through TfidfTransformer, and the implementation of the Random Forest classifier without specifying additional parameters.

To evaluate the model's performance, a 10-fold cross-validation was executed. The metrics obtained are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.66	0.64	0.65	26270
Neutral	0.64	0.63	0.63	26216
Positive	0.77	0.80	0.79	26265
Accuracy			0.69	78751
Macro Avg	0.69	0.69	0.69	78751
Weighted Avg	0.69	0.69	0.69	78751

Table 5: Classification Metrics

The corresponding confusion matrix is showed below:

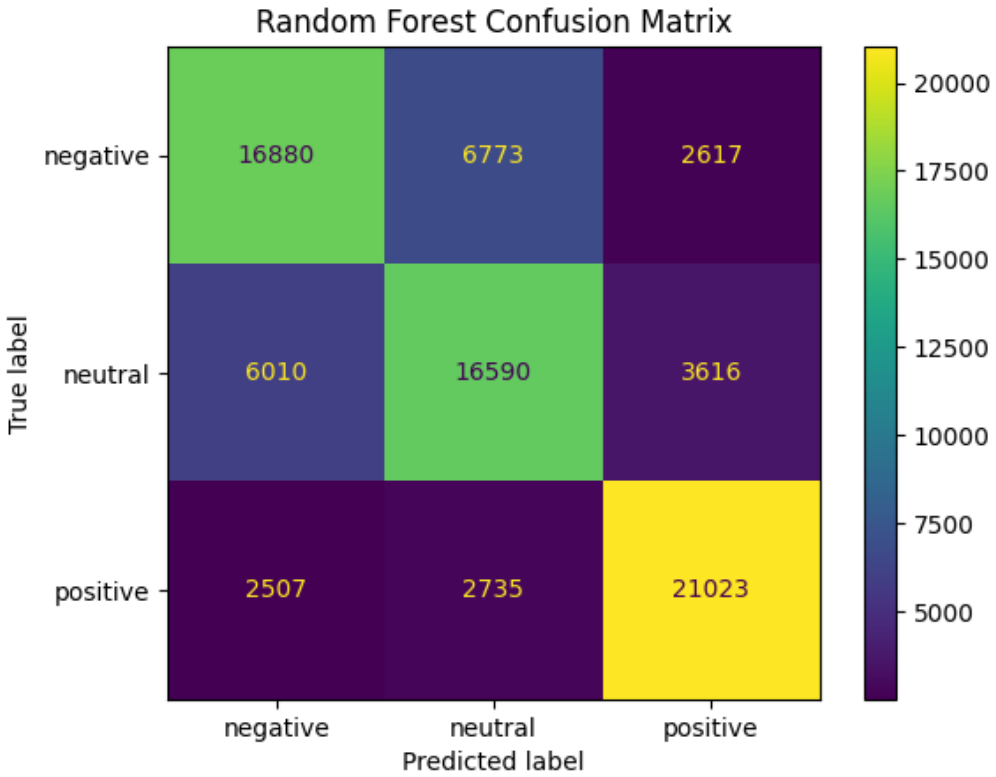

Figure 7: Confusion matrix

3.7 Classifier Comparison

To determine the most suitable model for sentiment classification, a paired t-test was conducted on the two best classifiers: the Multinomial Naive Bayes Classifier and the Linear Support Vector Classifier (SVC). The obtained p-value from the test is 0.00116, which is less than the significance level $\alpha = 0.05$. Consequently, the null hypothesis is rejected, providing the basis for selecting the model based on accuracy. So, the Linear Support Vector Classifier (SVC) is selected for our purpose, as it exhibits a lower error rate compared to the Multinomial Naive Bayes Classifier.

4 Streaming Analysis

The goal of this analysis is to explore whether changes have occurred over time in the sentiment of listeners towards specific themes or new trends discussed by various podcasts. A crucial aspect to consider is the impact of such changes on the predictive ability of our classifier. When a significant shift in sentiment occurs, it negatively affects the accuracy of our model's predictions.

It is important to emphasize that a decline in prediction accuracy may indicate a challenge in keeping the model updated and adapted to new audience dynamics. In other words, if our classifier was initially trained on data that has become outdated due to shifts in listener sentiment, the model may struggle to effectively capture new nuances of opinion.

This phenomenon can be particularly problematic when the model is used in dynamic contexts, such as the podcast landscape, where opinions and trends evolve rapidly. The need to adapt the model to these swiftly changing nuances becomes crucial to maintain its predictive effectiveness and relevance over time.

A graph was extracted to monitor the number of comments written in 2021, as it represents the year following 2020 (the year of the comments extracted for the previous analysis).

The comments distribution is shown below:

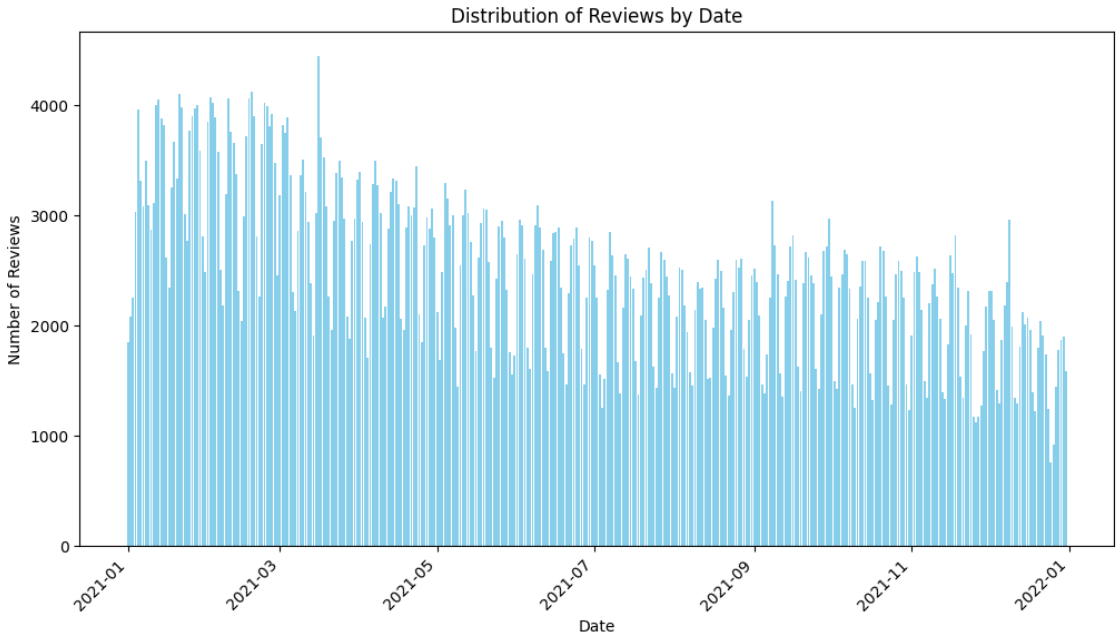

Figure 8: Distribution reviews 2021

In the plot above, some key events related to peaks of comments were identified. Below, the selected events are shown:

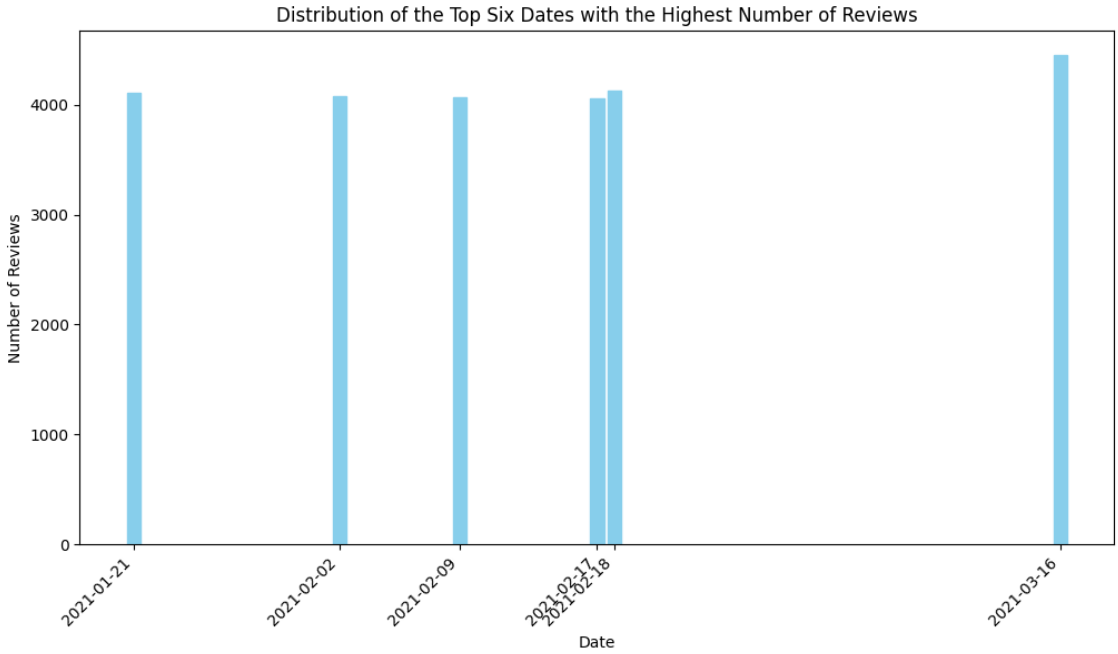

Figure 9: six top events

For each event, 60 new comments were selected for each class and labeled. These comments were then used as a test set for three different learning settings:

1. **Static Model:**
    - The initial model was constructed using the first training set.
2. **Sliding Model:**
    - This model was retrained periodically with the most recent comments of the training set. In each iteration, the oldest 60 comments were removed, and the newest 60 comments were added.
3. **Incremental Model:**
    - Trained with the initial training set adding the labelled data of all previous events.

This approach allowed the evaluation of the models under different conditions, considering both a static initial model and models that adapt to changing data over time.

4.1 Event 1 - 21/01/2021

This is the first event and for this reason I obtained the same result for all three models because they worked only with the initial dataset.

The results obtained are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.72	0.71	60
Neutral	0.72	0.60	0.65	60
Positive	0.74	0.85	0.79	60
Accuracy			0.72	180
Macro Avg	0.72	0.72	0.72	180
Weighted Avg	0.72	0.72	0.72	180

Table 6: Static Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.72	0.71	60
Neutral	0.72	0.60	0.65	60
Positive	0.74	0.85	0.79	60
Accuracy			0.72	180
Macro Avg	0.72	0.72	0.72	180
Weighted Avg	0.72	0.72	0.72	180

Table 7: Sliding Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.72	0.71	60
Neutral	0.72	0.60	0.65	60
Positive	0.74	0.85	0.79	60
Accuracy			0.72	180
Macro Avg	0.72	0.72	0.72	180
Weighted Avg	0.72	0.72	0.72	180

Table 8: Incremental Model Metrics

The confusion matrices are shown below:

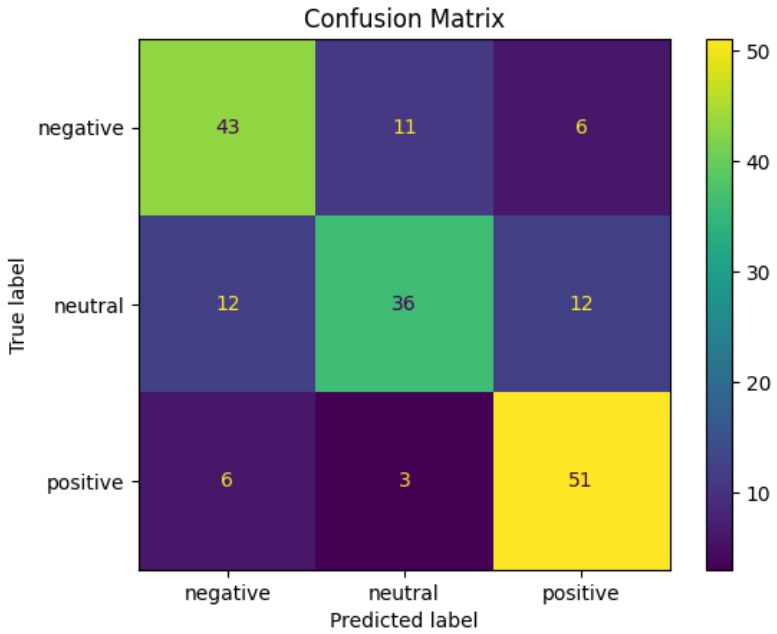

Figure 10: Static Model Confusion Matrix

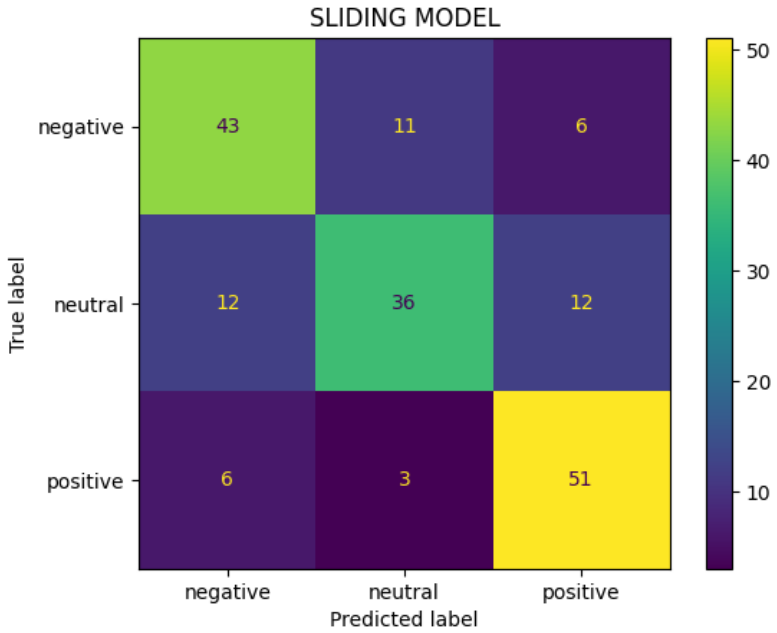

Figure 11: Sliding Model Confusion Matrix

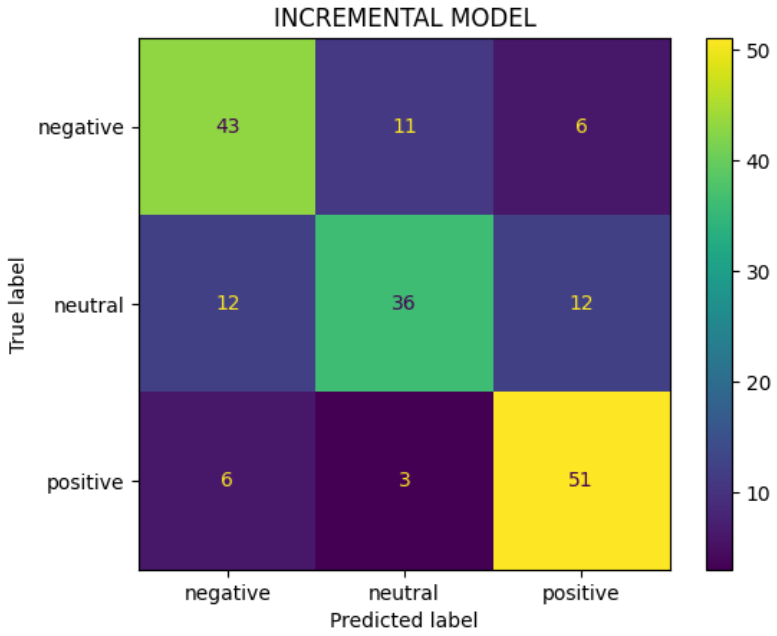

Figure 12: Incremental Model Confusion Matrix

4.2 Event 2 - 02/02/2021

The results obtained for this event are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.69	0.75	0.72	60
Neutral	0.74	0.67	0.70	60
Positive	0.82	0.83	0.83	60
Accuracy			0.75	180
Macro Avg	0.75	0.75	0.75	180
Weighted Avg	0.75	0.75	0.75	180

Table 9: Static Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.75	0.73	60
Neutral	0.74	0.67	0.70	60
Positive	0.82	0.85	0.84	60
Accuracy			0.76	180
Macro Avg	0.76	0.76	0.75	180
Weighted Avg	0.76	0.76	0.75	180

Table 10: Sliding Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.75	0.73	60
Neutral	0.74	0.67	0.70	60
Positive	0.82	0.85	0.84	60
Accuracy			0.76	180
Macro Avg	0.76	0.76	0.75	180
Weighted Avg	0.76	0.76	0.75	180

Table 11: Incremental Model Metrics

The confusion matrices are shown below:

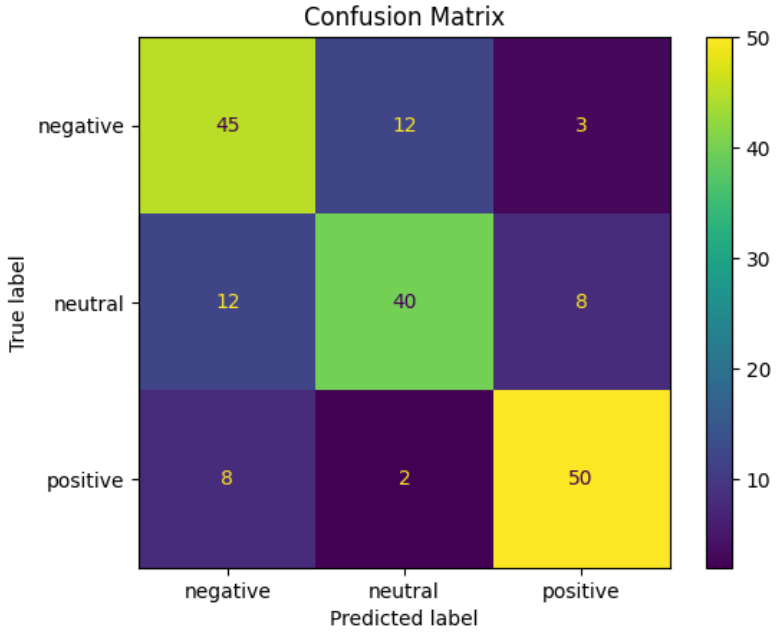

Figure 13: Static Model Confusion Matrix

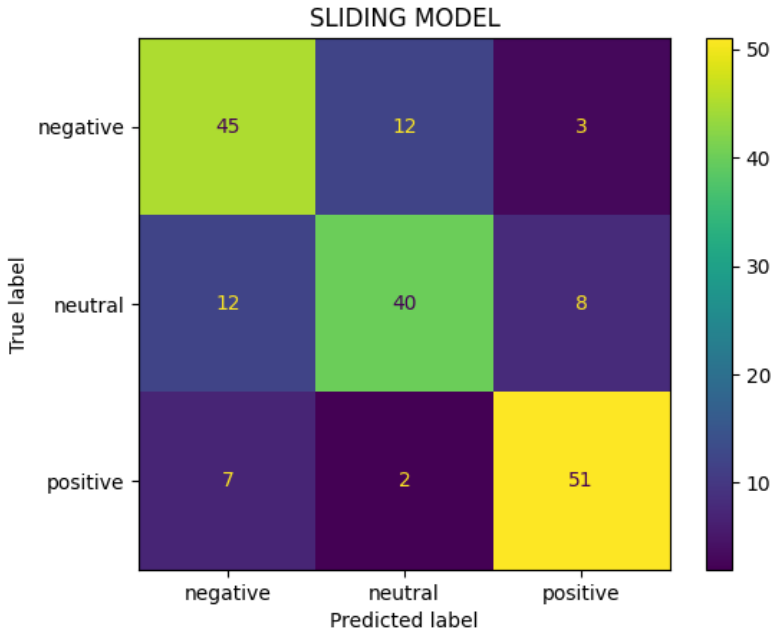

Figure 14: Sliding Model Confusion Matrix

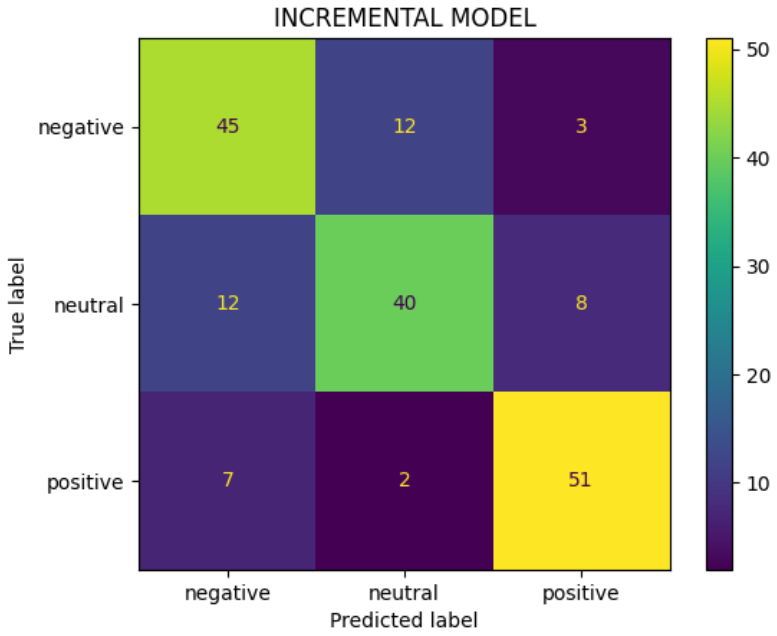

Figure 15: Incremental Model Confusion Matrix

4.3 Event 3 - 09/02/2021

The results obtained for this event are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.69	0.75	0.72	60
Neutral	0.74	0.67	0.70	60
Positive	0.82	0.83	0.83	60
Accuracy			0.75	180
Macro Avg	0.75	0.75	0.75	180
Weighted Avg	0.75	0.75	0.75	180

Table 12: Static Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.67	0.65	0.66	60
Neutral	0.66	0.70	0.68	60
Positive	0.79	0.77	0.78	60
Accuracy			0.71	180
Macro Avg	0.71	0.71	0.71	180
Weighted Avg	0.71	0.71	0.71	180

Table 13: Sliding Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.67	0.65	0.66	60
Neutral	0.67	0.70	0.68	60
Positive	0.80	0.78	0.79	60
Accuracy			0.71	180
Macro Avg	0.71	0.71	0.71	180
Weighted Avg	0.71	0.71	0.71	180

Table 14: Incremental Model Metrics

The confusion matrices are shown below:

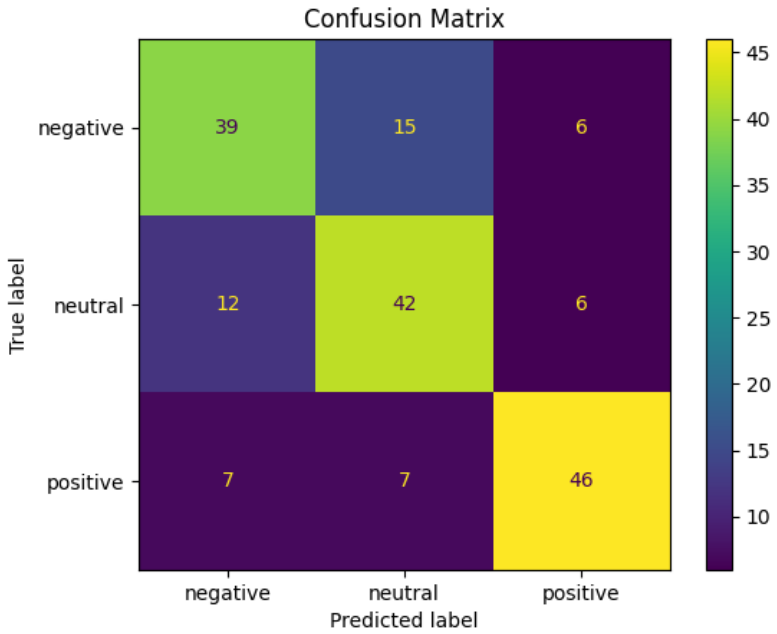

Figure 16: Static Model Confusion Matrix

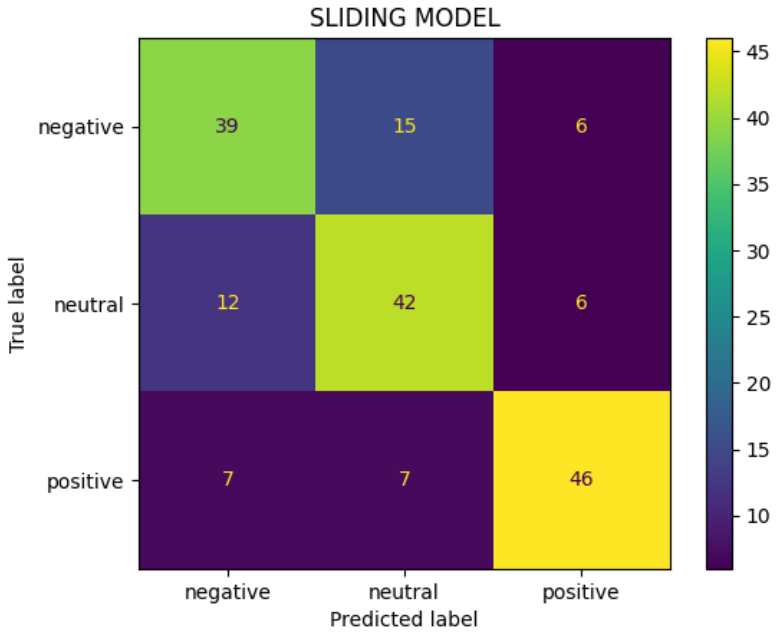

Figure 17: Sliding Model Confusion Matrix

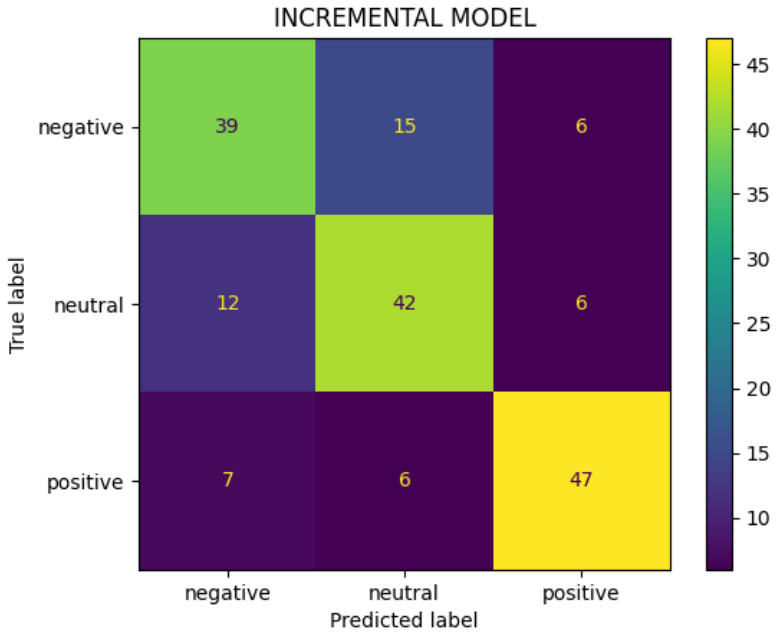

Figure 18: Incremental Model Confusion Matrix

4.4 Event 4 - 17/02/2021

The results obtained for this event are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.67	0.75	0.71	60
Neutral	0.70	0.63	0.67	60
Positive	0.86	0.85	0.86	60
Accuracy			0.74	180
Macro Avg	0.75	0.74	0.74	180
Weighted Avg	0.75	0.74	0.74	180

Table 15: Static Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.68	0.77	0.72	60
Neutral	0.72	0.63	0.67	60
Positive	0.86	0.85	0.86	60
Accuracy			0.75	180
Macro Avg	0.75	0.75	0.75	180
Weighted Avg	0.75	0.75	0.75	180

Table 16: Sliding Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.68	0.77	0.72	60
Neutral	0.72	0.63	0.67	60
Positive	0.86	0.85	0.86	60
Accuracy			0.75	180
Macro Avg	0.75	0.75	0.75	180
Weighted Avg	0.75	0.75	0.75	180

Table 17: Incremental Model Metrics

The confusion matrices are shown below:

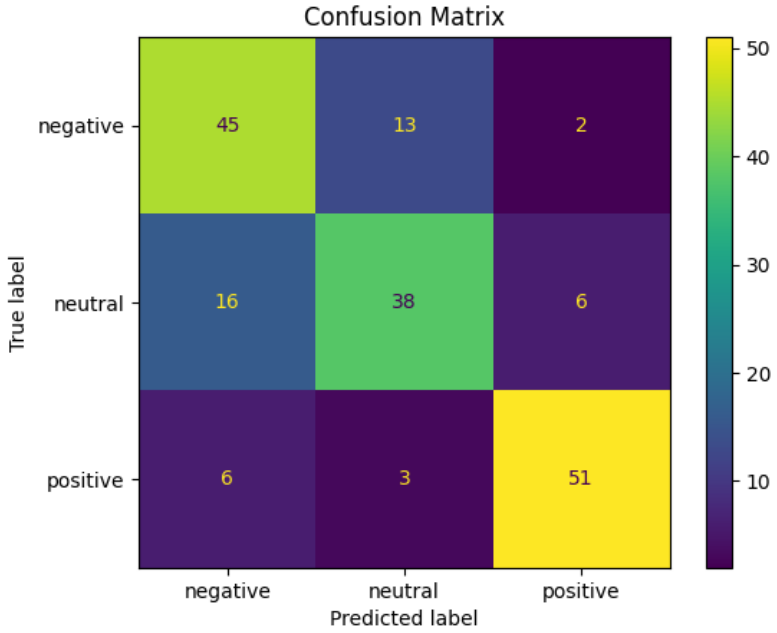

Figure 19: Static Model Confusion Matrix

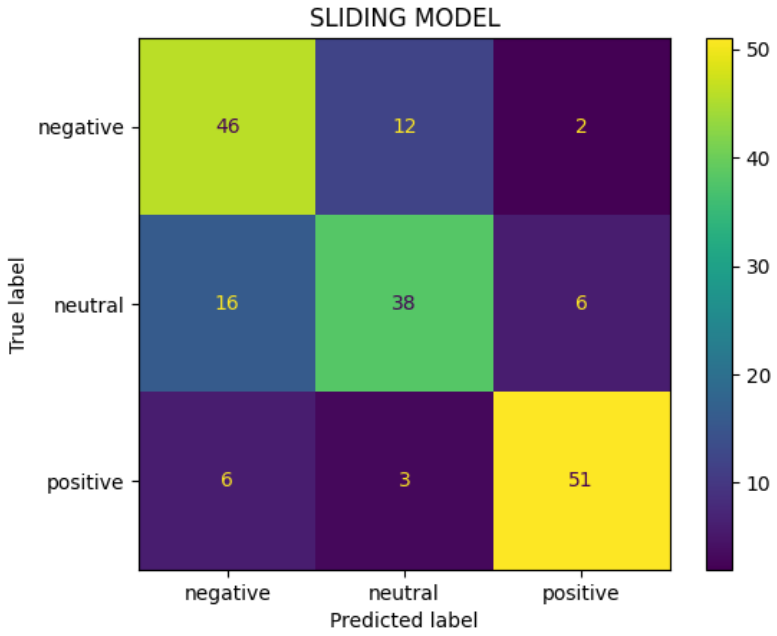

Figure 20: Sliding Model Confusion Matrix

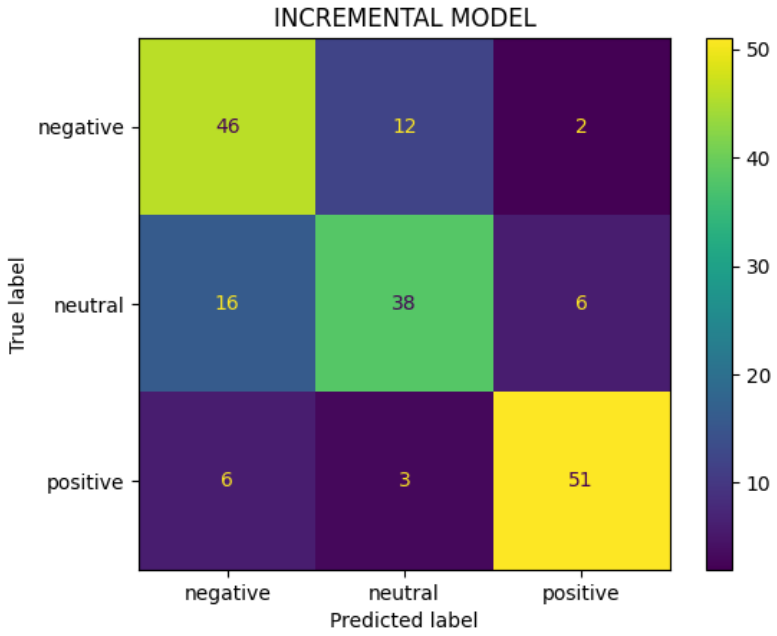

Figure 21: Incremental Model Confusion Matrix

4.5 Event 5 - 18/02/2021

The results obtained for this event are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.69	0.77	0.72	60
Neutral	0.77	0.62	0.69	60
Positive	0.85	0.92	0.88	60
Accuracy			0.77	180
Macro Avg	0.77	0.77	0.76	180
Weighted Avg	0.77	0.77	0.76	180

Table 18: Static Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.69	0.77	0.72	60
Neutral	0.77	0.62	0.69	60
Positive	0.85	0.92	0.88	60
Accuracy			0.77	180
Macro Avg	0.77	0.77	0.76	180
Weighted Avg	0.77	0.77	0.76	180

Table 19: Sliding Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.77	0.73	60
Neutral	0.77	0.62	0.69	60
Positive	0.85	0.93	0.89	60
Accuracy			0.77	180
Macro Avg	0.77	0.77	0.77	180
Weighted Avg	0.77	0.77	0.77	180

Table 20: Incremental Model Metrics

The confusion matrices are shown below:

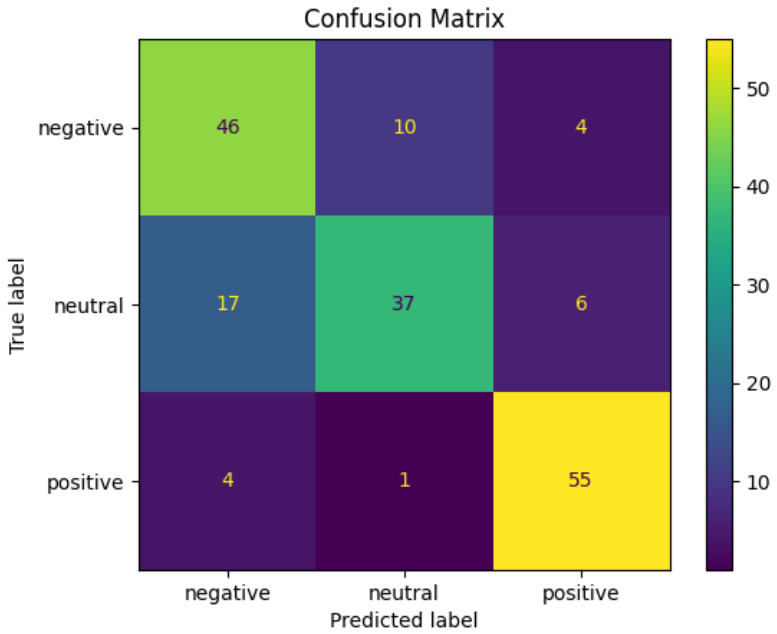

Figure 22: Static Model Confusion Matrix

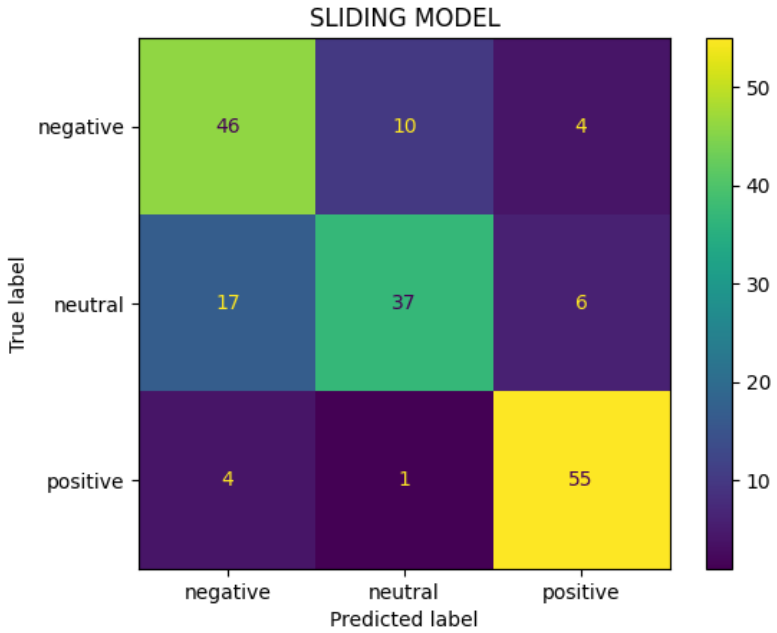

Figure 23: Sliding Model Confusion Matrix

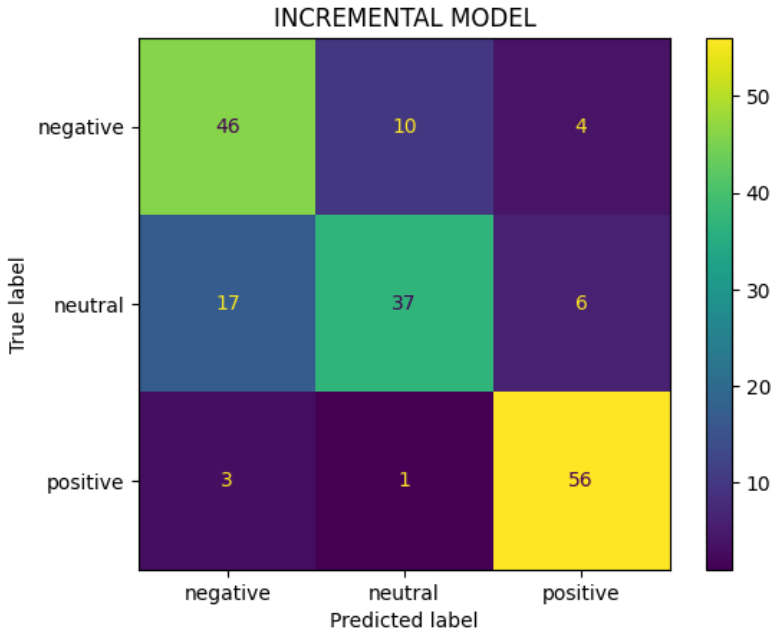

Figure 24: Incremental Model Confusion Matrix

4.6 Event 6 - 16/03/2021

The results obtained for this event are the following:

Class	Precision	Recall	F1-Score	Support
Negative	0.72	0.78	0.75	60
Neutral	0.76	0.68	0.72	60
Positive	0.84	0.85	0.84	60
Accuracy			0.77	180
Macro Avg	0.77	0.77	0.77	180
Weighted Avg	0.77	0.77	0.77	180

Table 21: Static Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.73	0.78	0.76	60
Neutral	0.75	0.72	0.74	60
Positive	0.86	0.85	0.86	60
Accuracy			0.78	180
Macro Avg	0.78	0.78	0.78	180
Weighted Avg	0.78	0.78	0.78	180

Table 22: Sliding Model Metrics

Class	Precision	Recall	F1-Score	Support
Negative	0.72	0.78	0.75	60
Neutral	0.77	0.72	0.74	60
Positive	0.86	0.85	0.86	60
Accuracy			0.78	180
Macro Avg	0.79	0.78	0.78	180
Weighted Avg	0.79	0.78	0.78	180

Table 23: Incremental Model Metrics

The confusion matrices are shown below:

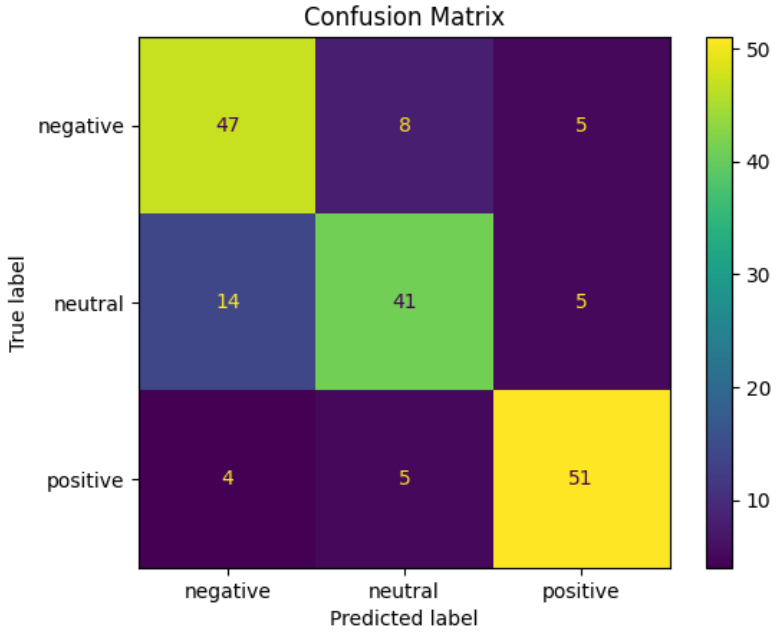

Figure 25: Static Model Confusion Matrix

Figure 26: Sliding Model Confusion Matrix

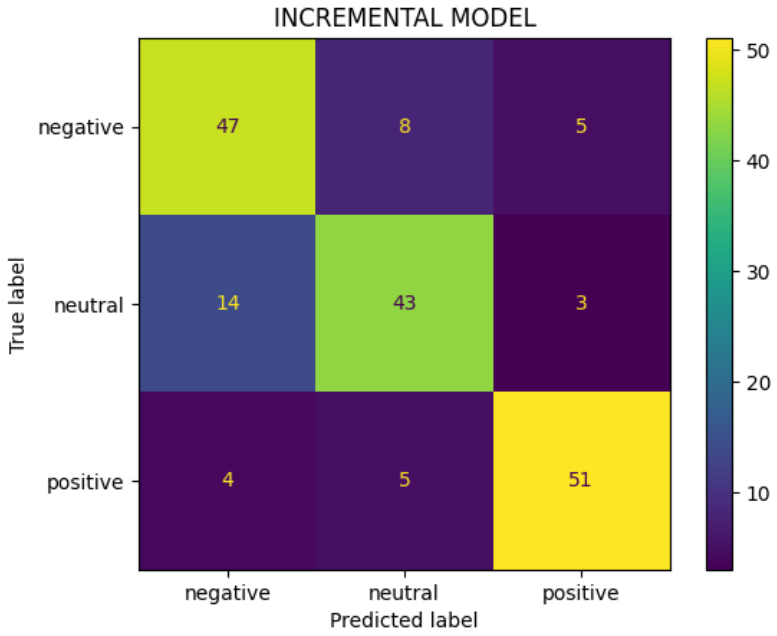

Figure 27: Incremental Model Confusion Matrix

4.7 Conclusion

The results collected from the previous analysis are shown in the following chart:

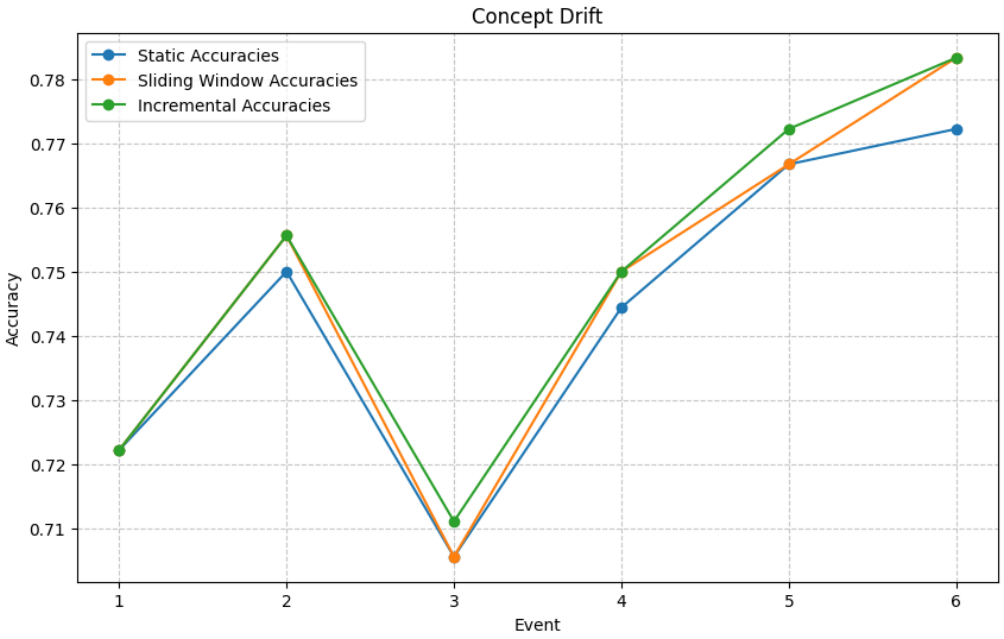

Figure 28: Concept Drift

The analysis reveals that the accuracies of the three models remain stable over time, with no apparent signs of deterioration. The examination did not explicitly identify significant concept drift, suggesting that users maintained a consistent manner of expressing opinions throughout the observed period. Additionally, it is worth noting that the incremental model demonstrates slightly better results, suggesting that continuous training could be advantageous in maintaining the model's performance; however, the difference is not substantial enough to necessitate the retraining of the classifier. Hence, the current model appears to adequately handle the data without a pressing requirement for a change.

Another aspect to consider concerns to the size of the dictionary. While the static model employs the same dictionary consistently, the sliding and incremental models undergo dictionary updates over time. Despite this, an increase in the vocabulary size is not observed for either the sliding model or the incremental model, as depicted in the graph below:

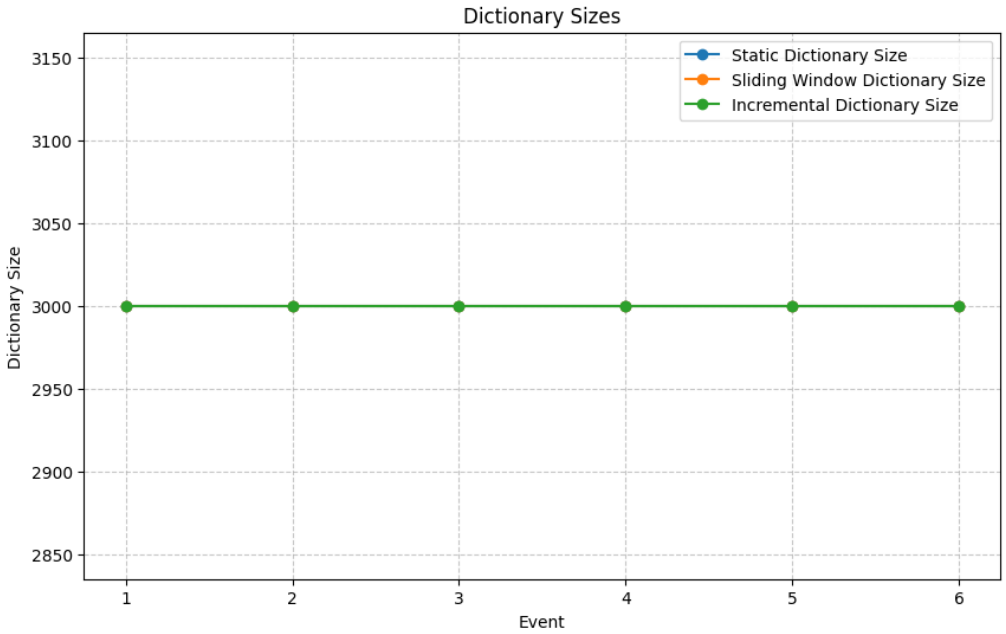

Figure 29: Concept Drift

As we can see, there is no change in the vocabulary despite receiving new input data. For this reason, it is decided to use the static model.

Considering the results with a training set of about 78,000 elements, it's important to note that having only 60 new samples per class for each event might not impact my initial training set significantly. This could make it hard to fully assess if changes have occurred. The decision to limit the samples to only 60 per class was influenced by the majority of reviews being positive, with only a few having neutral or negative sentiments. To keep a balanced dataset, the maximum allowable sample size was set at 60. Therefore, it's possible that with more samples, more reliable and satisfactory results could have been achieved.