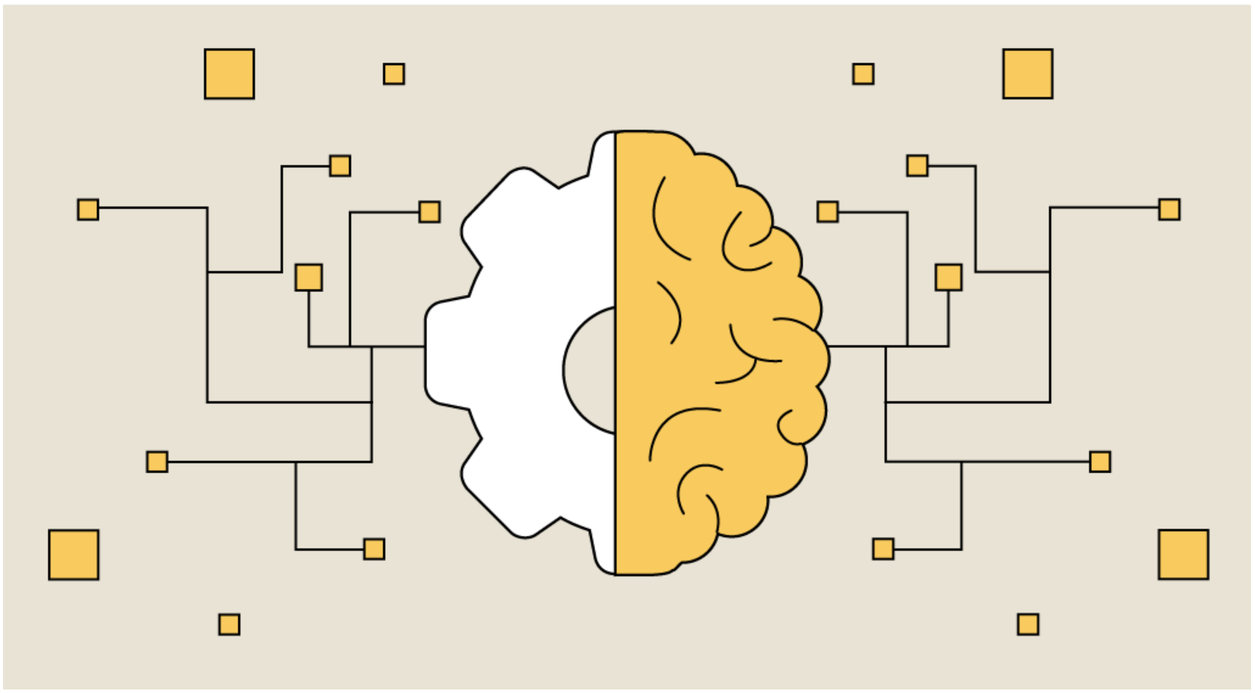
CRITICAL ANALYSIS REPORT

DATA MINING



BY: DIVYANK VERMA

Supervised Text Mining

Unsupervised Data Mining

Table of Contents

TEXT MINING 2

CRITICAL ANALYSIS REPORT 3

PART 1: SUPERVISED TEXT MINING 3

Question 1 3

Dataset 3

Cleaning of the Reviews 3

Creation of TF-IDF Matrix and hyperparameters 4

Model Training 6

Question 2 7

Choice of Precision Metrics 7

Choice of NBC for Model Deployment 7

Functionality of NBC and performance 8

Limitations of NBC 8

Performance of Naïve Bayes Classifier (Multinomial) 8

Deployment of Model 9

PART 2: UNSUPERVISED DATA MINING 10

Question 1 10

Umap 10

N_neighbours and Tuning: 10

Min_dist and Tuning 12

Question 2 14

Clusters 14

Sub-Clusters 15

CONCLUSION 16

Text Mining

- Firstly, we need to convert text data into numerical data so that we can use our model, because text data is unstructured and linguistic in nature.
- Once the data has been divided into numerical structure, we can train our model to predict the sentiment of the review.
- Collection of documents is here are collection of our 1000 reviews of movies.
- Each of the document is made up of tokens, individual words, or group of words. This may be referred to as *unigrams*, *bigrams*, and *trigrams*.
- TF – If a token occurs man times in a document, then its value is high.
- IDF – If a token occurs too many times in entire corpus, then its value is low.
- TF-IDF Frequency: As Term Frequency is capturing the significance of a token to individual document whereas Inverse document Frequency calculates the significance of the token to the entire corpus. To capture both we must use TF-IDF Matrix.

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

- Now if a token is present in a document and both the values of TF and IDF is on higher side then the value of its TF-IDF will also be higher. Therefore, not only if the token is present in a document but in entire corpus also then its TF-IDF value will be high. If the TF is high and IDF is low, then TF-IDF will be towards lower side. This value will be between 0 and 1.
- This TF-IDF matrix will now be used to train our model.
- Usually not more than combination of three words make sense when doing the tokenization hence we do not increase the value of n_grams more than 3.

Critical Analysis Report

Part 1: Supervised Text Mining

Question 1

Dataset

- My dataset of Film Reviews has 1000 rows of film reviews of all genres created with the help of open ai's chat_gpt.
- Nature of my dataset is semi-structured dataset as it has columns of Reviews, and Labels (Positive and Negative).
- Chat_gpt created an imbalance dataset of mix of positive and negative labels and reviews of movies.

Cleaning of the Reviews

- After the importing of the necessary libraries, packages, and the dataset the first step is to clean the dataset so that it is ready to train our model upon.
- This is done using the function 'cleaner'.
- Cleaned data is stored in the variable 'souped' after checking for any html links in reviews.
- Regular expression is used to clean the data further and replaced with empty space instead of symbols or un-needed characters.
- Cleaning of data is important as every review has lot of noise. Stop words such as 'a' 'and' 'the' are removed as they do not contribute towards training the model. This is done by installing the stopwords corpus from nltk package and used to check if those words occur in our reviews so that they can be removed by setting the language to 'english' as it has 8 more languages. Filter method is used.
- All the full stops, commas, and symbols are also removed.
- All the documents are converted into lowercase to better train our model and avoid our model to identify same words like 'Movie' and 'movie' as different words. This is done after converting all the reviews through word_tokenizer function to create it into tokens and then passing it to lower function to convert all the tokens to lowercase.

- Different Parts of speech such as ‘keep’, ‘kept’ and ‘keeping’ is removed as all the words mean same thing. This lemmatization is done by the function WordNetLemmatizer and setting it to ‘v’ so that only verb form of the part of speech is considered to make tokens. This also gives us the singular forms of the verb and removes all the plurals while also removing the synonyms. We do use one more corpus to achieve that which is ‘omw-1.4’ from nltk package as well.
- One last step is to check that no row should have any missing values or be empty after the cleaning is performed. This is done by keeping only the reviews whose length is more than 0.

Creation of TF-IDF Matrix and hyperparameters

- Shape of tfidf matrix that was created = 1025 x 578 signifying 1025 rows and 578 dimensions or variables created after all the *unigrams* and *bigrams* and *trigrams*.
- **n_grams**: This helps to create meaningful phrases without the linguistic structure knowledge. Although it does increase the number of features required to represent text. Bigrams requires more features than unigrams and trigrams requires even more. Lot of these features can be useless as those might not make any sense. This is a parameter that can be used to create:
  - *Unigrams*: single individual unique words
  - *Bigrams*: combination of two tokens/words e.g.: very good. Action scene, mind blowing etc.
  - *Trigrams*: combination of three tokens/words. e.g.: exceed customer expectations etc.
- For my dataset I have chosen the maximum value of n grams to be 3, as upon creating only unigrams and bigrams a lot of good trigrams were left out. This was checked by creating the vocabulary file of the unigrams, bigrams and trigrams and checking it manually by downloading it.
- As we previously broken our dataset into several tokens, now we must stitch them together as TFID Vectorizer function does not accept any input which is already tokenized. Cleaned_review stores now the stitched back all the tokens to create single documents again without any noise and unnecessary symbols.

- **Min_df:** Minimum document frequency is the hyperparameter which create a value by multiplying the number of total rows/documents to the hyperparameter value, which gives us minimum accepted document frequency count.
- Can be set to any number between 0 to 1. The default min df value in the package of TFID Vectorizer in Scikit Learn is set to 1. If the value is set to 1 then it means that our model will ignore all the *terms/tokens that will appear in less than 1 document*. Hence, I started with slightly high values of min df.
    - I tried a value Min df **0.008** which means our model will consider terms if they appear for at least 8 columns among the dataset of 1025 rows. I checked the mean precision score of both SVC and NBC which were 95.69 and 96.88 respectively. Although the score was quite good but the vocabulary size of only 280. A lot of good tokens were getting ignored at 8 columns as my minimum document occurring frequency was quite big to take all the necessary tokens.
    - Then I tried an extremely low value of 2 rows. **0.002** which gave me a lot of non sensical unigrams, bigrams, and trigrams like 'acting phenomenal story', 'visually stunning emotionally'. This helped me to concluded that I need to select a value slightly higher value than 2 rows and lower than 8. Vocabulary size was 1099.
    - For my dataset I chose **0.004** as minimum document frequency after a lot of experimentation. This got me up to **578** variables. Most of my bigrams and trigrams now made sense and almost all unigrams made perfect sense.

Min Document Freq = 1000 x 0.004 = 4 (approx.)

- This means if our unigrams and bigrams occur in at least 4 documents in the corpus then those tokens will be considered important and will be included in the vocabulary file.
- **Min_df: 0.004**, **n_gram (1,3)** = Precision score for SVC: 96.95 and NBC: 97.56 (This was the best value of min df which gave me sensible vocabulary like 'mind-blowing', 'absolute-disappointment', 'definitely-recommend' while also giving a fairly good performance on SVC and NBC). Value of 0.004 got rid of useless trigrams, bigrams, and unigrams. Tokens that are not

present in at least 4 documents or reviews will be discarded as they play less or almost no importance due to their less occurrence.

- Going lower than this gave me lot of non-sensible words in dictionary of vocabulary which didn't made sense like 'beginning end' 'absolute also'. It did increase the model performance, but the vocabulary was not good enough to differentiate between good and bad review as it has lot of unnecessary *unigrams*, *bigrams*, and *trigrams*.
- Further Steps
    - Now using the 'transform' method I can populate my tf-idf matrix with tf-idf values.
    - Using the get_features_out method I saved all my features of unigrams and bigrams to the vocabulary csv file so that I can review it and import it again during the deployment of my model on unlabelled dataset.
    - Shape of my tf-idf matrix that I finally got with optimal values was (**1025** rows x **578** columns). This means I now have 578 unique variables including unigrams and bigrams.
- This concludes our conversion of text data to numerical data and any model can be deployed like SVC or NBC.
- Table 1 refers to the experimental values tried to create tfidf matrix.

Min_df	Shape of TFIDF	LinearSVC	NBC
0.008	1025 x 280	96.88	95.69
0.002	1025 x 1099	98.13	98.97
0.004	*1025 x 578*	*96.95*	*97.56*

TABLE 1

Model Training

- At first, I trained the model of SVC on my tf-idf matrix as it does not depend upon the dimensionality of the dataset like Random Forest. It just requires few rows of the dataset. Therefore, it is a good model to use in case of text data.
- We also do not need to use any kernel here as our dataset is already in a very high dimensionality of 578 variables and we do not want to increase it further.
- Then I trained NBC for the classification of film reviews. NBC is much simpler and interpretable than LinearSVC. This theorem makes a 'naive' assumption that all the variables are independent of each other.

- NBC is a linear classifier, which works well over here as our dataset is in high dimensional space and is also linearly separable.
- NBC predicts a class by using the probabilistic formulae which is called bayes theorem.
- This bayes theorem helps to calculate conditional probabilities which is Probability of A given B occurs.

$$P(A|B) = [P(A) \times P(B|A)] / P(B)$$

- Here we are predicting the probability of a class given a particular document therefore if class = C (positive and negative) and document(reviews) = D our formulae which NBC is using:

$$P(C|D) = [P(C) \times P(D|C)] / P(D)$$

Question 2

Choice of Precision Metrics.

I chose **Precision** Metrics for my NBC and Linear SVC model as:

- Any movies that are not good or have negative reviews, I didn't want them to be predicted as positive or a good movie.
- Hence my objective was to reduce the false positives that might be mis interpreted by my model.

Choice of NBC for Model Deployment

I chose Naïve Bayes Model for classification on my Unlabelled dataset as it was also performing better than Linear SVC. I used a conditional statement just before dumping my model as **.sav** file to check which model is performing better and saving the better performing model which can be used later Unlabelled dataset for prediction.

LinearSVC mean precision = 96.95

Naïve Bayes mean precision = 97.56

Functionality of NBC and performance

- **Naïve Bayes Theorem** is a supervised learning algorithm which is based on Bayes theorem. This theorem is used for classification.
- Naïve Bayes is a very simple model as compared to Linear SVC and gives very effective classification. It is very quick in giving prediction as compared to Support Vector.
- As this theorem and this model works on the probabilistic classifier therefore prediction on the basis of the probability of an object given a particular class is very accurate as compared to Linear SVC.
- Upon testing both LinearSVC and NBC, I observed in my testing that Naïve Bayes was giving much better predictions than Linear SVC.
- In my case LinearSVC gave the precision score of 96.95 and NBC gave the mean precision score of 97.56. hence making NBC a clear winner. Although I also took my prediction of model on Unlabelled dataset to determine if NBC was better or not. NBC was making better predictions than SVC.

Limitations of NBC

- It is true that NBC was predicting the Unlabelled dataset very accurately but there still are few limitations of this model:
    - One of the biggest limitations of NBC is that it assumes all the features as independent and unrelated which does not happen in real life deployment scenarios.
    - Another major limitation of NBC is that in certain scenarios if it finds data or words in test sample which was not present in the training data, then it assigns the probability 0 and we could end up with **zero class probabilities**. To nullify this drawback, we must add certain smoothing factor or a kind of penalty to both numerator and denominator of the equation to avoid any zeros.

Performance of Naïve Bayes Classifier (Multinomial)

- NBC does works well and gives better predictions on Unlabelled dataset therefore in our situation, NBC was chosen for deployment too.
- Another advantage of NBC is that this algorithm has no hyperparameters to tune and hence is much easier to implement generally in real-life scenarios too.

- I used 10 cross fold validation as well for NBC and used a for loop to give precision score on each iteration and then finally the mean of all the scores.
- Multinomial Naïve Bayes is used here instead of other types (Gaussian, Bernoulli) of NBC as our dataset was neither continuous in nature nor binary and had just two class values, positive and negative.

Deployment of Model

- After saving the model of NBC it was ready for deployment on the Unlabelled dataset which was also created using open ai's chat_gpt.
- Unlabelled dataset is just like Labelled dataset with just **40** rows and no label column of positive or negative class.
- Deployment is done by using 3 files altogether:
    - Unlabelled csv file
    - Vocabulary file saved earlier.
    - Sav file which has our trained model of Naïve Bayes.
- When we got the predictions using the NBC model sav file, I opened the file to check if the Positive label (1) and Negative label (0) has been correctly predicted or not.
- The model worked very well and the results very more than satisfactory as among 40 rows 39 rows were predicted correctly as positive or negative.
- Example: "This is one of the best movies I've seen in years with a gripping plot and incredible performances" was predicted as 1 which is positive.
- In comparison the LinearSVC only gave me 37 correct predictions when I checked the prediction csv file.

Part 2: Unsupervised Data Mining

Question 1

Umap

- Unified Manifold Approximation and Projection is a dimensionality reduction technique which can reduce the high dimensional dataset to just two dimensions. We had a dataset of **578** dimensions which was reduce to just two using UMAP.
- This is primarily used for data-visualization.
- Similar datapoints are coloured identical.
- A radius is projected from each datapoint, and if the radii overlaps any other radius of any other near datapoint then those datapoints are put together called as cluster. In our case **50** n_neighbours was the optimal value.
- Picking a correct radius is important. Too big radius may result in putting many data points under single cluster and if the radius is too small, we can end up isolating a lot of data points into separate grouping. As choosing fixed radius is impossible therefore UMAP choses variable radius.
- This radius projecting out of any data point depends on its K'th nearest neighbour. This could be 3rd, 4th, 10th, 100th nearest neighbour. This is a hyperparameter which is called n_neighbour which can be tuned in the code. The radius is then extended to that K'th nearest neighbour. The probability is calculated of being a neighbour.
- If a surrounding data point is nearby, then its probability of becoming a neighbour is quite high and vice versa. This depends on the distance. Fuzzy radii is used which fades with distance.

N_neighbours and Tuning:

- This is the most important hyperparameter when it comes to visualization of this algorithm.
- Up to which **K'th neighbour** we should extend the radius is controlled by this hyperparameter.
- First, I started with low values of n_neighbours like **40** and min_dist **0.3** which gave me two clear clusters of Positive and Negative class. As we can see in figure 1 the red plots represent positive label class which are clustered on top

left corner and purple plots represent Negative class which are mostly present in bottom right corner. The clusters are clearly separable.

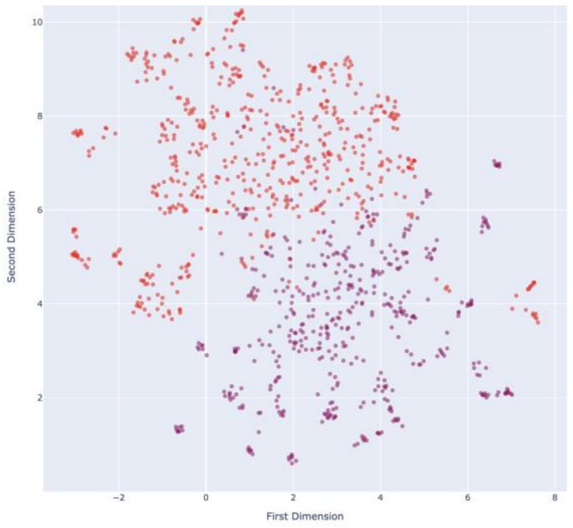


FIGURE 1

- Then I moved to a very high value of Kth neighbour of **200** and min_dist was still **0.3** as my data points were neither too dense nor too sparse in my visualisation. Visualisation is in Figure 2.

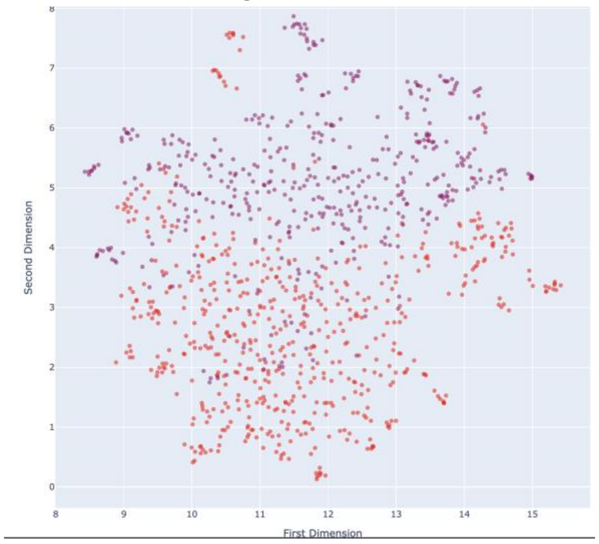


FIGURE 2

- In figure 2 we can see that the classes are still clearly separable, although there are a lot of data points that are plotted in the wrong cluster. Hence, I got a better plot when I was clustering with low value of n neighbour.

- I tried different n neighbours and the best plot that was most separable visually was at **50 n_neighbours** which made most clearly separable cluster as the radii projected to the kth neighbour was helping the plots to be clustered separately.

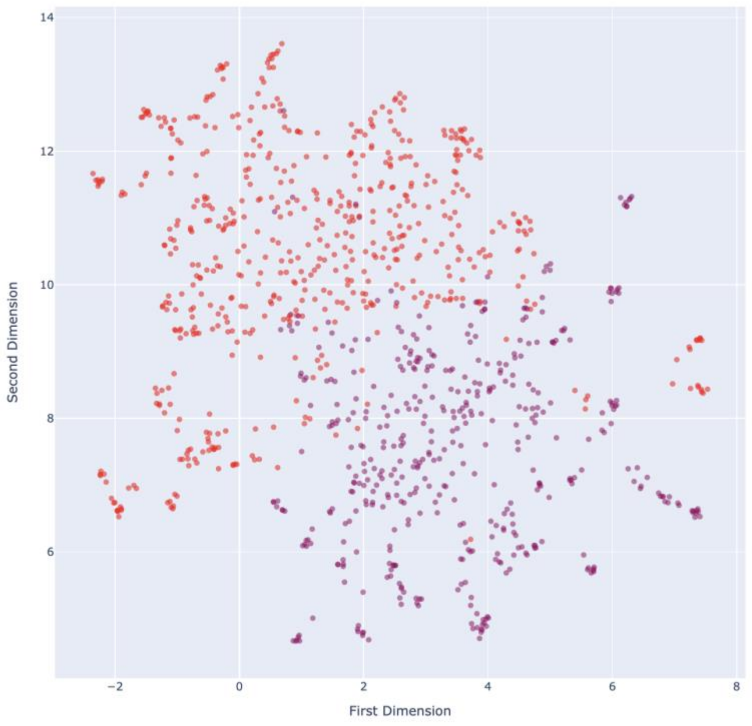


FIGURE 3

- The best N_neighbour after a lot of testing along with Min_dist was of 50 that I could rely upon to create. In Figure 3 upon testing on 50 n neighbours the classes were most clearly separable.

Min_dist and Tuning

- Minimum distance between similar data points. Establishes if the UMAP plots will be dense in nature or sparse in nature.
- I started with a very low value of 0.1 (Figure 4) and although both the classes were clearly separable but similar data points were too close to each other hence not giving a proper insight from the visualisation, thus defeating the objective of creating the UMAP.

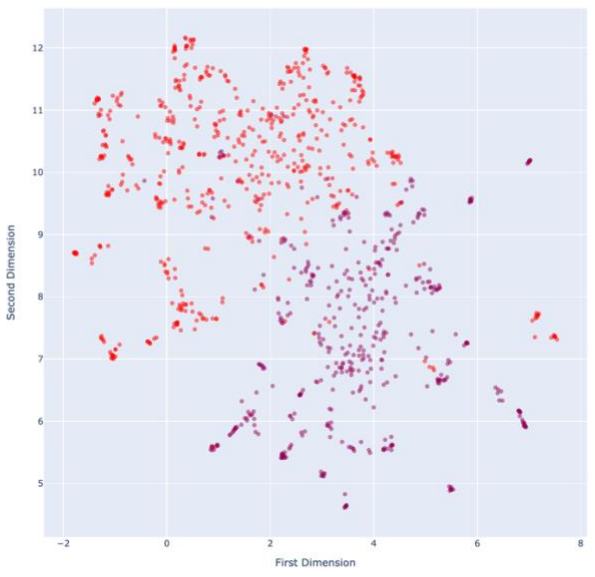


FIGURE 4

- Then I tried a high value of 0.8 (Figure 5) which increased the distance between similar datapoints and hence the plot was very sparse in nature. This helped me to conclude that I must try a lower value of min_dist but not too low of 0.1.

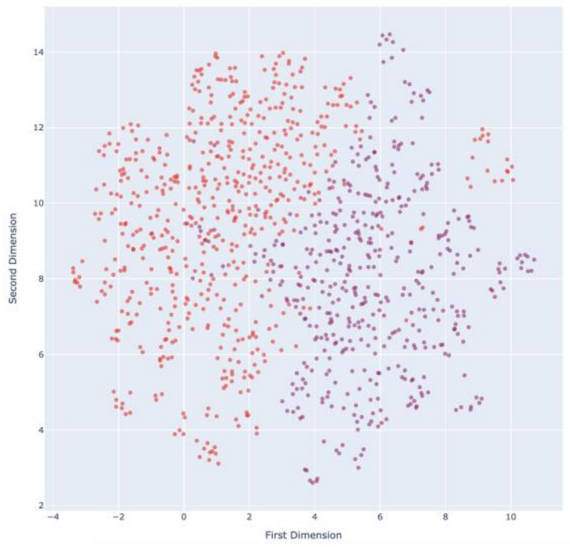


FIGURE 5

- The best value after experimentation of several values the best value I could get was of 0.5 (Figure 6). In this value all the data points were clearly separable, and the cluster was formed visually separable.

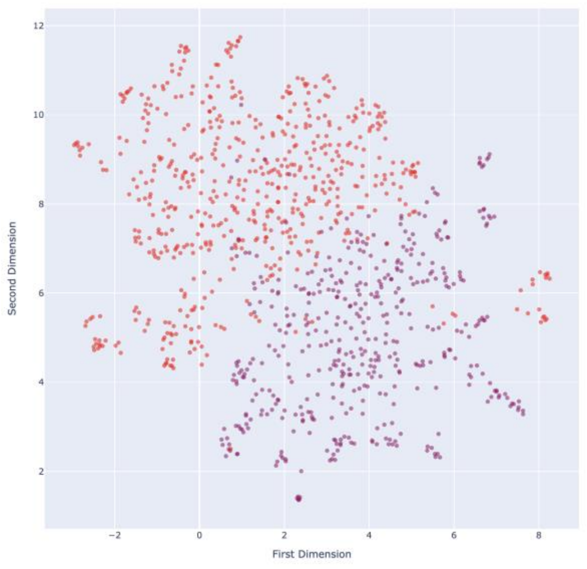
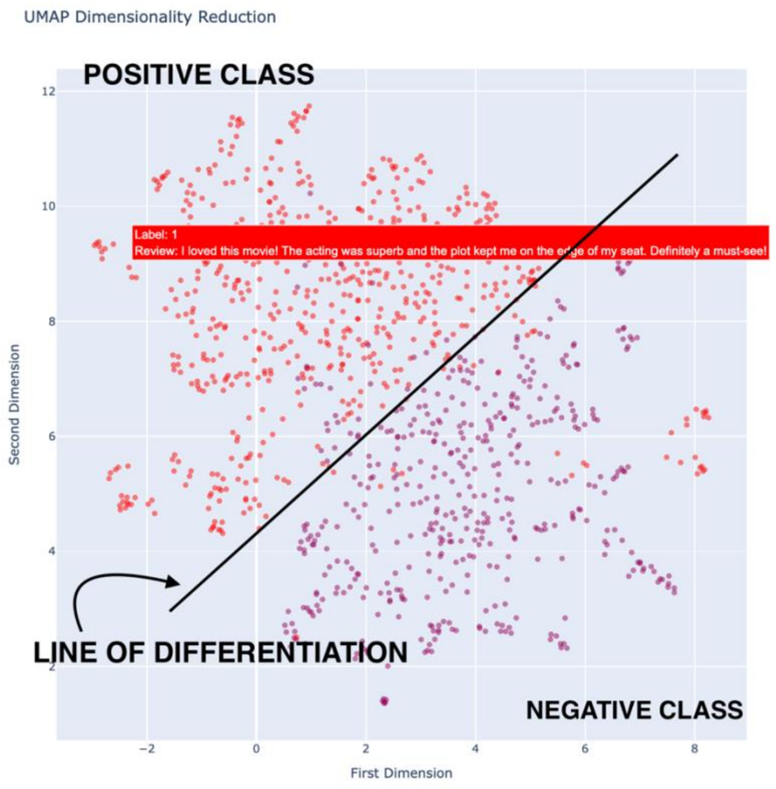


FIGURE 6

- Best hyperparameters hence finalized for my dataset were:
  - **50** n_neighbours
  - **0.5** min_dist

Question 2

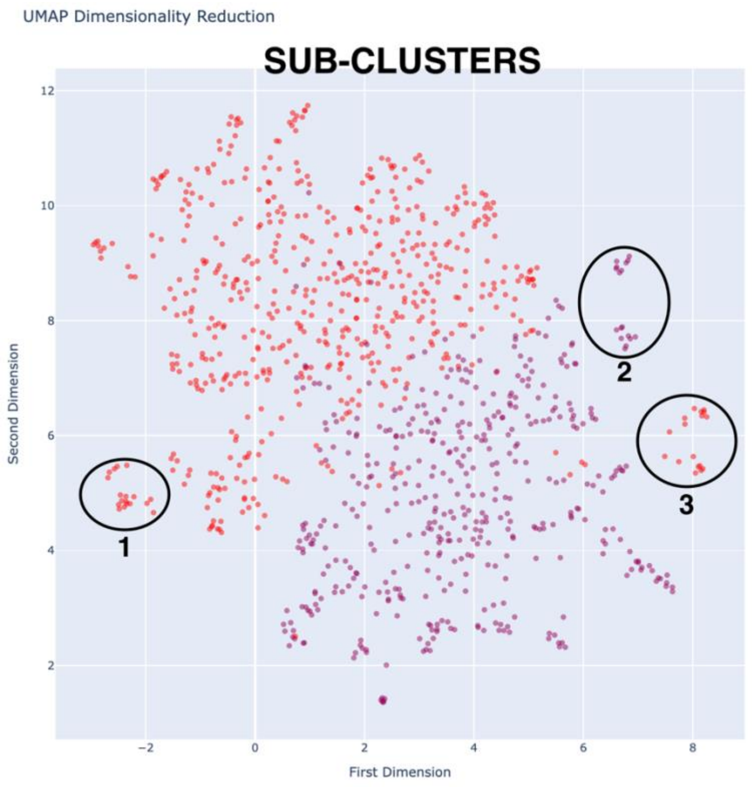


Label: 1
Review: I loved this movie! The acting was superb and the plot kept me on the edge of my seat. Definitely a must-see!

Clusters

- After tuning the hyper-parameters of n neighbours and minimum distance. I can say that my dataset of Movie reviews was clearly separable on plotting the UMAP.
- In the top left hemisphere of the MAP the Positive Cluster can be clearly identified and so can the Negative Cluster in the Bottom right.
- I have demonstrated how both classes are separable using a line of differentiation.

Sub-Clusters.



- There were three clearly visible subclusters that I found in my visualisation.
- All the data – points in these sub clusters had a similarity of tokens (unigrams, bigrams and trigrams) and were talking in general about the same things regarding the movie reviews.
- **1st sub-cluster**: In this all the data points were talking about the performance being excellent and the story being emotional. Hence bigrams such as 'emotionally resonant', 'stories were excellent' were common in this cluster.
- **2nd sub-cluster**: In this sub-cluster all the reviews were talkin about the humour being crude and animation being over the top. Hence unigrams such as 'animation' and trigrams 'humour was crude' was common. This sub cluster was of negative class and were making critics on the movies.
- **3rd sub-cluster**: This sub-cluster contained data points which acted as kind of outliers as they were positive reviews but they were lying on the negative class cluster. I found out on examining the tokens of these sub-clusters that 'coreography' and 'music' were usually very common in negative cluster class and not in positive class. This could be the reason why these data points lied on the opposite class. All of these reviews talked about the music of the movie being very good and catchy.

Conclusion

- I conclude that the dataset that was created using open ai's chat_gpt was not very good as upon plotting the UMAP I uncovered that it has given me a lot of similar reviews, which was expected of chat_gpt 3.5 as it has a low token size and starts giving similar results after some time.
- LinearSVC and NBC both performed good. Although Naïve Bayes classifier performed exceptionally well with 39/40 correct predictions whereas Support Vector Machine gave 37/40 correct predictions and a lower mean precision score.
- False Positives were reduced with a very good efficiency by both the models upon training on the 1000 labelled dataset with both giving mean precision score above 95%.
- UMAP visualization did give a very distinguishable cluster due to the clear and precise vocabulary that was created and the good performance of Naïve Bayes on Text Mining use case. Both the classes were clearly separable in my visualisation and few sub-clusters could be spotted as well.