Data Mining: CA 2
Critical Analysis Report

Task 1:

Q1. Conversion to structured format/ TF-IDF Matrix – Discuss how tuning hyperparameters 'ngram_range' and 'min_df' affected the TF-IDF matrix and classifiers' performance.

Ans 1- The transformation of raw textual data into a structured format, specifically a TF-IDF (Term Frequency-Inverse Document Frequency) matrix, is a vital step. The TF-IDF vectorization technique is used to convert the cleaned reviews into numerical features.

Coding Steps:

1. Importing necessary libraries
2. Reading the dataset
3. Cleaning the reviews
4. Removing empty rows
5. Printing and dropping columns
6. Saving cleaned reviews to a new CSV file
7. Preparing data for modelling
8. Counting the distribution of sentiment
9. Implementing Support Vector Classifier (SVC)
10. Implementing Naive Bayes Classifier (NBC)
11. Saving the best-performing model

Data Preprocessing:

The provided dataset, MovieReviews.csv, is read into a DataFrame. Text cleaning is performed using Beautiful Soup to remove HTML entities and regular expressions to replace non-alphabetic characters with whitespace. Tokenization, lowercase conversion, stop word removal, and lemmatization are applied to the

cleaned reviews. Rows with empty cleaned reviews are removed from the DataFrame.

Feature Development:

TfidfVectorizer is used to convert the cleaned reviews into TF-IDF values. The TF-IDF matrix's information is controlled by the hyperparameters min_df and ngram_range.

Evaluation and modelling:

The models Linear Support Vector Classifier (SVC) and Naive Bayes Classifier (NBC) are used. The models' accuracy on different subsets of data is evaluated using 10-fold cross-validation. For both models, the mean accuracy over all cross-validation folds is obtained.

Model Choice and Saving:

Between SVC and NBC, the model with the highest mean cross-validation accuracy is chosen. The chosen model is fitted to the entire dataset and stored using joblib.

Observations:

The notebook describes the whole workflow, beginning with data loading and preprocessing and ending with feature engineering and model training. Cross-validation is used to assess the correctness of both the SVC and NBC models. The time spent training and evaluating each model is recorded.

Observation and interpretation:

The notebook shows how changing hyperparameters, notably min_df and ngram_range, can alter the TF-IDF matrix and thus the performance of the classifiers. The code shows how to implement and compare two prominent classification algorithms for sentiment analysis, SVC and NBC. The comparison of accuracy ratings indicates which model outperforms the supplied dataset. The notebook demonstrates the model selection decision-making process based on cross-validation findings.

'ngram_range': Larger n-grams collect context, but they have more noise and dimensionality. Smaller n-grams indicate a more compact matrix but less context. Richer context can improve classifier performance, but too much context can cause overfitting.

'min_df': Increasing the value results in a sparser matrix by deleting rare words. Lower values result in a denser matrix with more noise. Increasing ‘min_df' reduces noise, but crucial words may be removed.

The impact on the TF-IDF matrix is as follows: We may customize the level of contextual information acquired by the TF-IDF matrix by adjusting the 'ngram_range'. Higher n-gram counts can capture more contextual information.

Effect on classifier performance: Increasing the number of n-grams may increase classifier performance by capturing more contextual information and enhancing the model's ability to detect patterns.

Model Selection and Evaluation: The code implements two distinct classifiers, LinearSVC and MultinomialNB, and evaluates their performance using 10-fold

cross-validation. Each fold's accuracy score is determined, and the mean accuracy is reported.

Implementing SVC keeping ngram at (1,3) with different min_df, got a different score like min_df=.0015 accuracy is 0.88, min_df=.0006 accuracy is 0.89, min_df=.0086 accuracy is 0.87 and at ngram at (1,1) accuracy is 0.88.

Implementing NBC Ingram at (1,3), got scores like min_df=.0015 accuracy is 0.86, min_df=.0006 accuracy is 0.87, min_df=.0086 accuracy is 0.85 and at ngram at (1,1) accuracy is 0.85.

The time taken by NBC is 2.008; the Mean cross-validation accuracy is 0.87 while in SVC the time taken is 13.302; the Mean cross-validation accuracy is 0.89

Q2. Model Deployment –Deploy the best-performing model on Unlabelled.csv and save its predictions to the same csv file. Discuss whether its performance was found satisfactory or unsatisfactory on deployment. Would it perform well if deployed on a dataset containing smartphone reviews? Why or why not?

Ans 2: Model Selection: The final model (SVC or NBC) is chosen on the basis of mean accuracy.

The first iteration (SVC): Mean Cross-Validation Precision: 0.6515 and time taken is 4.53 seconds at ngram (1,1) while at ngram (1,3) results were min_df=.0015 accuracy is 0.62, min_df=.0006 accuracy is 0.65, min_df=.0086 accuracy is 0.57

The second iteration (NBC): The mean cross-validation accuracy is approximately 0.6511 and the time taken is 2.07 seconds at ngram (1,1) while at ngram (1,3) results min_df=.0015 accuracy is 0.62, min_df=.0006 accuracy is 0.65, min_df=.0086 accuracy is 0.56

Both models performed about average. Given that the first iteration's mean cross-validation accuracy is marginally higher, the first model performed slightly better. However, the difference in accuracy between the two iterations is minor.

Smartphone Reviews: Because of differences in language, sentiment, and writing style in smartphone reviews, model performance may differ from the original dataset. The model's performance is determined by the quality and diversity of the smartphone review dataset. If the sentiment expressions differ significantly, the result may differ.

Task 2:

Q1. Using the given dataset, create two Word Clouds – one containing 10 MOST USED WORDS in the positive sentiment reviews and the other containing 10 MOST USED WORDS in the negative sentiment reviews. (Note – the word clouds must not include the commonly occurring words in English such as a, an, the, is, of, in, etc.) In addition to providing the Python code, briefly discuss your approach and provide Word-Clouds in the pdf report. The size of the words should get bigger as they become more frequently used, as shown in the sample Word Cloud below.

Ans 1- The offered code builds and presents two Word Cloud visualizations to provide insights into the most commonly used words in positive and negative sentiment reviews. Word Clouds are graphical representations in which word size corresponds to word frequency. We can create a word cloud in three steps:

- Extract the review (text document)
- Create and generate a word cloud image
- Display the cloud using matplotlib

DataFrame Creation: df = pd. DataFrame(data)

This line creates a data frame named df using the provided data. The structure and columns of this data frame are determined by the structure of the data you've loaded.

Positive and Negative Review Filtering:

These lines distinguish positive and negative sentiment reviews in the data frame. The cleansed reviews of each sentiment are then combined into distinct strings (positive_text and negative_text).

Generating Word Clouds:

positive_wordcloud = WordCloud (max_words=10, width=400, height=300, background_color='white'). generate(positive_text)

negative_wordcloud = WordCloud (max_words=10, width=400, height=300, background_color='white'). generate(negative_text)

These lines use the WordCloud class to generate Word Cloud visualizations. The max_words parameter specifies the maximum number of words to display in the Word Cloud. Other parameters control the dimensions of the Word Cloud and the background colour.

Word Cloud Plotting:

The matplotlib library is used in these lines to build a figure with two subplots. Each subplot shows a Word Cloud produced from the generated Word Cloud data. The imshow function shows the Word Cloud image, while the title function gives each subplot a title. Finally, plt. show () shows the complete figure together with the Word Clouds.

Approach:

1. Using the pd. DataFrame(data) command, a DataFrame named df is produced.
2. The data frame is filtered for positive and negative reviews using boolean indexing based on the sentiment column.
3. Positive and negative review text data is concatenated into separate strings (positive text and negative text).
4. Word clouds for the ten most frequently used words in both positive and negative sentiment reviews are generated.
5. The visualizations are created using the Word Cloud class from the Word Cloud library.

6. For plotting, the matplotlib library's plt module is used.

Generated Word Clouds:

The generated Word Clouds visually represent the most frequent words in both positive and negative reviews. Larger words indicate a higher frequency of appearance in the respective sentiment. We can also change some optional arguments of the word cloud like max_font_size, max_word, and background_color and used interpolation="bilinear" in plt. imshow() to make the presented image appear more smoothly. By examining these visualizations, you can gain insights into the language and terms commonly associated with each sentiment.