Project Report: Sentiment Analysis using Support Vector Classifier (SVC)

1. Introduction

This project focuses on performing sentiment analysis using text data from a CSV file that contains user reviews or comments labeled with three possible sentiments: positive, negative, and neutral. Sentiment analysis is a powerful tool used to analyze the sentiment behind textual data, helping businesses understand customer opinions, product feedback, or public sentiment on specific topics.

In this project, we used a Support Vector Classifier (SVC) as the primary algorithm for classifying sentiment. The text preprocessing includes various steps such as cleaning, tokenization, stopword removal, and lemmatization, implemented using popular Python libraries such as `nltk`, `spacy`, and `sklearn`.

---

2. Objectives

- To clean and preprocess the raw text data for sentiment classification.

- To build a Support Vector Machine (SVM) model using the cleaned data.

- To perform hyperparameter tuning to improve model performance.

- To analyze and evaluate the performance of the model using a confusion matrix and other performance metrics.

- To predict sentiments for unseen text and save results for future analysis.

---

3. Dataset

The dataset used for this project is in CSV format, and it consists of two main columns:

1. Text: Contains user reviews or comments.

2. Sentiment: Labels that classify each review into three categories:

  - Positive

  - Negative

  - Neutral

The dataset is divided into training and testing data for model development and evaluation. Additionally, external review data is used to test the model's ability to generalize.

---

4. Libraries and Tools

The following Python libraries were used in the project:

- numpy: For numerical operations.

- pandas: For data manipulation.

- matplotlib & seaborn: For data visualization.

- nltk & spacy: For text preprocessing, tokenization, stopword removal, and lemmatization.

- sklearn: For machine learning tasks, including vectorization, model training, and evaluation.

- collections: For counting word occurrences.

---

5. Methodology

5.1 Text Preprocessing:

The text data underwent several preprocessing steps:

- Lowercasing: Converting all text to lowercase to ensure uniformity.

- Punctuation Removal: Removing all punctuation marks using `string.punctuation`.

- Tokenization: Breaking down the text into individual words using `nltk`.

- Stopword Removal: Removing common stopwords (e.g., “the”, “is”) using `nltk` to retain meaningful words.

- Lemmatization: Converting words to their base form using the `spacy` library.

5.2 Vectorization:

After preprocessing the text data, Count Vectorization was applied using `sklearn`'s `CountVectorizer` to convert the cleaned text into numerical format. This numerical representation allows the model to understand the text data for classification.

5.3 Model Training:

We employed a Support Vector Classifier (SVC) for classification, known for its effectiveness in high-dimensional spaces. We used a linear kernel with fine-tuned hyperparameters:

- C (Regularization parameter): 0.2

- Gamma (Kernel coefficient): 0.8

The training data was split into 80% training and 20% testing to evaluate model performance.

5.4 Hyperparameter Tuning:

A loop was implemented to tune the `C` and `gamma` parameters for optimizing the SVC model's performance. Various values of `C` and `gamma` were tested in a grid-like search.

5.5 Model Evaluation:

The model's performance was evaluated using:

- Confusion Matrix: Visual representation of true vs. predicted labels.

- Classification Report: Includes precision, recall, F1-score, and support for each class (positive, negative, neutral).

- Accuracy Score: Measures the overall correctness of the model.

---

6. Results

6.1 Confusion Matrix

	Predicted Negative	**Predicted Neutral**	**Predicted Positive**
True Negative	50	8	2
True Neutral	10	70	5
True Positive	3	7	65

The confusion matrix indicates that the model performs well in predicting all three sentiment classes, though there is a slight mix-up between neutral and the other sentiments.

6.2 Classification Report:

- Precision: 0.85 (overall precision for all classes)
- Recall: 0.83 (how many actual positives our model correctly identified)
- F1-score: 0.84 (harmonic mean of precision and recall)
- Accuracy: 0.85 (overall accuracy)

6.3 Example Usage:

Input:(some sentences)

["I love this movie!", "This product is terrible.", "The food was delicious.", "This product is alright"]

Output:

I love this movie! 😃
This product is terrible. 😔
The food was delicious. 😃
This product is alright. 😐
```

### 6.4 Sentiment Predictions for Real Review Data:

The trained model was used to predict the sentiment for real customer reviews in the TeePublic_review.csv file. The results were saved to a new CSV file called `TeePublic_review_with_sentiments.csv`.

---
```

7. Conclusion

The SVC model successfully classified sentiment with an accuracy of 85%. The confusion matrix and classification report indicated that the model performs well in distinguishing between positive, negative, and neutral sentiments. Future improvements can be made through additional tuning and exploring different model architectures like deep learning.

8.Future work

- ❑ use it in ecommerce website .
- ❑ building it in form of webpage where you write text and return sentiment maybe.
- ❑ build an API to allow other websites to use it.
- ❑ improve model and train it on more data.
- ❑ improve vectorizer.

---

9. References

- nltk Documentation: https://www.nltk.org/

- spacy Documentation: https://spacy.io/

- sklearn Documentation: https://scikit-learn.org/
