## **Project Documentation: Sentiment Analysis using NLP**

### **Project Overview**

The project aims to perform sentiment analysis on textual data using Natural Language Processing (NLP) techniques. By analyzing user reviews or opinions, the goal is to classify sentiments into three categories: **Positive**, **Negative**, or **Neutral**. This project uses libraries such as NLTK, Scikit-learn, and TfidfVectorizer for text processing, feature extraction, and model building.

### **Objectives**

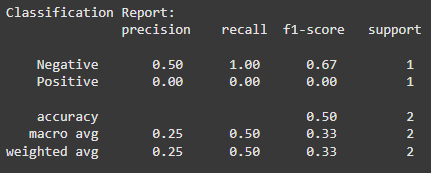
* **Preprocessing Text**: Clean and preprocess raw text data, including tokenization, stopword removal, and conversion to lowercase has been used.
* **Feature Extraction**: Used TF-IDF vectorization to transform text data into numerical features suitable for machine learning.
* **Model Training**: Implemented a Naive Bayes classifier to classify the sentiment of given text data.
* **Evaluation**: Evaluated the model using accuracy, classification report, and confusion matrix.
* **Prediction**: Tested the model by predicting sentiments of new, unseen data.

### **Technologies Used**

* **Python**: Programming language.
* **NLTK**: Natural Language Toolkit for text preprocessing.
* **Scikit-learn**: Machine learning library for model training and evaluation.
* **Matplotlib / Seaborn**: Libraries for data visualization (confusion matrix heatmap).
* **Google Colab**: Development environment for executing the project code.

### **Steps Involved**

1. **Dataset Preparation**:
   * I have used a sample dataset of textual reviews labeled with sentiments ("Positive", "Negative", "Neutral").
   * The dataset consists of a Text column with user reviews and a Sentiment column with corresponding sentiment labels.
2. **Text Preprocessing**:
   * **Tokenization**: Split the text into individual words.
   * **Stopword Removal**: Filter out common words (e.g., "the", "is", "and") that do not carry much meaning.
   * **Lowercasing**: Convert all text to lowercase to maintain uniformity.
   * **Alphanumeric Filter**: Keep only words that are alphabetic (remove punctuation and numbers).
3. **Feature Extraction**:
   * I have used TfidfVectorizer from sklearn to convert the preprocessed text into a numerical format. This technique gives higher weight to words that appear less frequently in the corpus but are important for classification.
4. **Model Training**:
   * A **Multinomial Naive Bayes** classifier was trained on the vectorized text data. Naive Bayes is a suitable choice for text classification problems, especially with features that are conditionally independent, as in the case of word frequencies.
5. **Model Evaluation**:
   * After training, the model was evaluated on a test set. Calculated various metrics, including accuracy, precision, recall, and F1-score, using the classification report from sklearn.

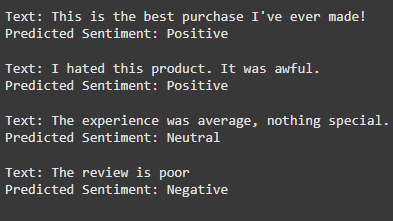


* + A **confusion matrix** was plotted to visualize how well the model classified the sentiments.

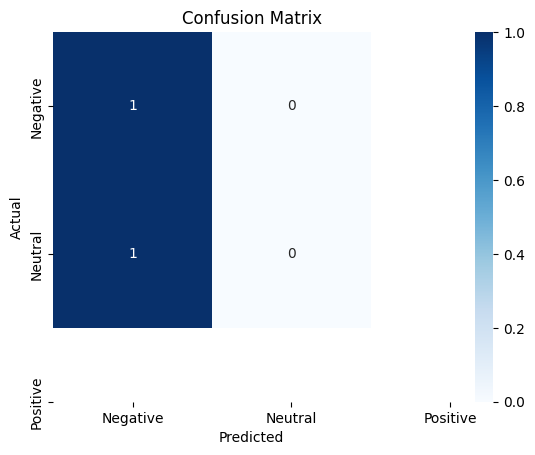
1. **Prediction**:
   * The model was used to predict the sentiment of new text data (unseen reviews). The model assigns one of the three possible sentiments: "Positive", "Negative", or "Neutral".

### **Results**

* **Classification Report**: The model was evaluated using precision, recall, and F1-score metrics, which provide a comprehensive understanding of the model's performance in each class (Positive, Negative, Neutral).



* **Confusion Matrix**: The confusion matrix showed the true positives, true negatives, false positives, and false negatives, providing a visual overview of the classifier’s performance.



* **Accuracy**: The final accuracy of the model was computed on the test dataset, giving a quick evaluation of how well the model performs overall.

### 

### 

### 

### **Future Work**

* **Model Enhancement**: Experiment with more advanced models like **Support Vector Machines** (SVM), **Random Forest**, or **Deep Learning** models (e.g., LSTM or BERT).
* **Data Augmentation**: Use larger datasets or scrape data from different platforms like Twitter or Amazon product reviews to create a more generalized model.
* **Hyperparameter Tuning**: Perform hyperparameter optimization (e.g., grid search or random search) to improve the model's performance.
* **Real-time Sentiment Analysis**: Implement a real-time sentiment analysis system to analyze social media posts or live customer feedback.

### **Conclusion:** This project demonstrated the application of NLP and machine learning techniques for sentiment analysis. The Naive Bayes model was able to effectively classify sentiment into predefined categories. Future improvements could include using a larger and more diverse dataset, exploring other models like SVM or neural networks, and fine-tuning hyperparameters for better accuracy.