Sentiment Analysis Model with TF-IDF and Logistic Regression

1. Introduction:

This document outlines the process of building a sentiment analysis model using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and Logistic Regression. The goal of this model is to predict the sentiment of product reviews, classifying them into positive, neutral, or negative sentiments.

The following steps were performed:

- Data preprocessing
- Feature extraction using TF-IDF
- Model training with Logistic Regression
- Model evaluation
- Deployment preparation (saving model and vectorizer)

2. Steps to Build the Sentiment Analysis Model:

Step 1: Label Sentiments

In this step, sentiments were labeled based on the review score:

- Positive sentiment for scores >= 4
- Neutral sentiment for scores = 3
- Negative sentiment for scores <= 2

Step 2: Data Visualization

The sentiment distribution was visualized using a countplot to observe the distribution of positive, neutral, and negative sentiments in the dataset.

Step 3: Text Preprocessing

Text preprocessing included removing any missing data and keeping relevant columns for analysis, such as the review text and sentiment labels.

Step 4: Feature Extraction using TF-IDF

TF-IDF was used to convert the textual data into numerical features. This technique captures the importance of words in the reviews and assigns weights based on their frequency and uniqueness across the corpus.

Step 5: Model Training using Logistic Regression

A Logistic Regression model was trained to predict sentiment labels based on the TF-IDF features of the reviews.

Step 6: Predictions

Predictions were made on the test data, generating sentiment labels (positive, neutral, or negative) for each review.

Step 7: Evaluation

The model's performance was evaluated using accuracy, precision, recall, F1-score, and confusion matrix. These metrics provided insights into how well the model is classifying sentiment in the test data.

Step 8: Confusion Matrix Visualization

A heatmap was used to visualize the confusion matrix, making it easier to interpret how well the model is performing for each sentiment class.

3. Model and Vectorizer Saving for Deployment:

Once the model and vectorizer were trained, they were saved using the joblib library to facilitate deployment. These files can be loaded later for making predictions without retraining the model.

4. Example Usage:

To use the trained model and vectorizer for new predictions, the following code can be used:

```
# Load the vectorizer and model
vectorizer = joblib.load('tfidf_vectorizer.pkl')
model = joblib.load('sentiment_model.pkl')

# Example review
example_review = ["This product is amazing! I love it."]
example_tfidf = vectorizer.transform(example_review)
example_prediction = model.predict(example_tfidf)

# Output the prediction
print("Example Prediction:", example_prediction[0])
```