**3.2 Sentiment Analysis of Customer Reviews**

**Step – 1: Identifying the dependent and independent features**

-There are only 2 columns in this dataset and the review text is the independent variable

-The rating is the dependent variable. I have classified the ratings as Positive, Negative and Neutral. The reviews which have a rating of 4 or 5 stars are considered positive, a rating of 3 is considered neutral and a rating of 2 or 1 star(s) is taken as a negative review.

-By classifying the ratings into these three categories, I have an explicit dependent feature, which makes this problem a supervised machine learning problem.

**Step – 2: Split the dataset into training dataset and test dataset**

-In order to evaluate and generalize the model, we need to see how well the model performs on unseen data, part of the data is used as test dataset.

- 80% of the dataset is used for training the model and 20% of it is used for testing the model to check generalizability.

**Step - 3: Data preprocessing:**

-Convert the text into lowercase

-Tokenize the text using word tokenizer

-Remove stopwords (e.g.: the, is, are, was, etc.) since they do not provide any information regarding the emotion of the review

-Lemmatize the text

-Vectorize the lemmatized tokens using Bag of Words vectorization technique or TfIdf vectorization

**Step - 4: Model Building:**

-Building a model using classification algorithms such as Logistic Regression, Decision Tree, Naïve Baye’s algorithm or Random Forest classifier. The model should be able to correctly classify the reviews as Positive, Negative or Neutral.

**Step – 5: Model Evaluation and Selection:**

-Evaluate the model using metrics such as accuracy score to see how well the model performs on the training dataset and the test dataset.

**Step – 6: Model Deployment using Streamlit:**

-Save the models as well as the vectorizer as pickle files

-Deploying and testing the model using streamlit framework includes loading the best model and the preprocessed training data (X\_train\_preprocessed). Using TfIdf vectorizer, fit\_transform the preprocessed training data to get the vectors. Then vectorize the input review as well using vectorize.fit(review\_text). Then make predictions using the best model pickle file and display the result to the user.

**Explanation:**

* The data is imbalanced, as there are a large number of positive reviews but very few negative and neutral ones. In order to reduce the effect of this imbalance on the predictions, I made sure that the ratios of the three categories remained same after splitting the dataset into training and test dataset.
* Data preprocessing is done separately on the training and test data (i.e., after splitting) in order to avoid data leakage problem.
* CountVectorizer uses Bag of Words (BOW) technique in order to get a large dimensional matrix that accounts for every word found in the review text and counts how many times each word appears in each review. Upon vectorization using BOW, we get a large numerical matrix which can be passed into any relevant ML algorithm.
* Tfidf on the other hand handles stopwords by reducing their relevance as their frequency increases. The words that appear frequently possess least value and hence will not be added into the vector matrix.
* Since this is a multi-factor classification problem (as the dependent variable has 3 discrete values), I used classification algorithms to train my model.
* I used Tfidf vectorizer in order to vectorize the review text. And to build the models, I used Logistic Regression, Decision Tree, MultinomialNB and Random Forest classifier and among them, Logistic Regression had high accuracy in both training and test data. Hence, I trained my model with Logistic Regression and saved the trained model using pickle.
* I built an app using streamlit. I saved the preprocessed training text and fitted the TfidfVectorizer. In order to vectorize the user input text, I used TfidfVectorizer and made predictions using the pickle file of the Logistic Regression model. Based on what the model predicts, the app displays if the input review is a positive, negative or neutral review.
* My model made accurate predictions when I gave any positive or negative reviews as input but it did not perform very well when a neutral review is given. Only if the word “ok” is included in the input review the model predicts it to be a neutral one. The very few neutral and negative reviews could be affecting the predictions but for the most part, the model was able to correctly predict and hence can be generalized.