**Introduction:**

In the age of digital commerce, customer feedback and sentiment analysis play a pivotal role in understanding and enhancing the user experience. With the proliferation of online fashion retail, it has become imperative for businesses to accurately classify customer reviews into distinct sentiment categories - positive, negative, and neutral. This classification enables retailers to gain valuable insights into customer satisfaction levels and areas for potential improvement. To address this need, we have developed a machine learning model tailored to the specific requirements of an online fashion retailer.

**Abstract:**

The aim of this project is to create a robust sentiment classification model tailored for an online fashion retailer. The model is designed to predict sentiment labels for customer reviews, categorizing them into three distinct classes: positive, negative, and neutral. Leveraging a dataset of customer reviews, we employ a multi-step approach to prepare and preprocess the data. This involves tasks such as data loading, removal of missing values, and feature engineering through tokenization and padding. Subsequently, a deep learning architecture is implemented, comprising an embedding layer, LSTM (Long Short-Term Memory) layer, and a dense layer with softmax activation to facilitate multi-class classification. The model is trained and evaluated on a split dataset to ensure optimal performance.

**Methodology:**

The methodology employed in this project encompasses several key stages. Initially, the dataset is loaded and explored to gain a comprehensive understanding of the available information. Subsequently, data preprocessing is performed to remove any rows with missing values in the 'Review' column, ensuring a clean and reliable dataset for analysis. Feature engineering involves the application of tokenization to convert text data into numerical format, enabling the model to process and analyze it effectively. The LabelEncoder is employed to transform categorical sentiment labels into a format suitable for machine learning algorithms.

For model development, a Sequential neural network architecture is chosen, incorporating an Embedding layer to represent words in a continuous vector space, an LSTM layer to capture temporal dependencies, and a dense layer with softmax activation to facilitate multi-class classification. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss function, and accuracy is chosen as the evaluation metric.

To assess model performance, the dataset is split into training and validation sets using an 80-20 ratio. The model is then trained over five epochs, and performance metrics such as accuracy and loss are visualized to gauge its effectiveness.

**1.Conceptualization / Logic applied:**

The code demonstrates a clear understanding of the problem statement and applies appropriate logic for sentiment classification. It uses a recurrent neural network (LSTM) for text classification, which is a suitable choice for analyzing text data.

**2.Usage of Frameworks:**

The code efficiently utilizes popular frameworks for machine learning and text processing. It leverages TensorFlow and Keras for building and training the model, which are widely used in the field of deep learning.

**3.Dataset / Preprocessing / Data preparation:**

The code loads a dataset (presumably containing customer reviews) using pandas.

It handles missing values by dropping rows with missing 'Review' entries, which is a valid approach.

Tokenization and padding are performed using the Tokenizer and pad\_sequences functions from TensorFlow, which is a standard procedure for processing text data.

**4.Model development / solution / Accuracy:**

The code constructs an appropriate neural network architecture for sentiment classification, including an embedding layer, an LSTM layer, and a dense layer with a softmax activation function.

It compiles the model with the Adam optimizer and sparse categorical cross-entropy loss, which are suitable choices for this classification task.

The training process is well-structured, including splitting the data into training and validation sets, and monitoring the training process with accuracy and loss metrics.

The model is trained for 5 epochs, which may be adjusted based on the convergence of the training process.

The code achieves a good accuracy for the given scenario – 91%

A Sequential Neural Network architecture is a type of artificial neural network where the neurons or nodes are organized in a linear, sequential manner. It is called "sequential" because the data flows through the layers in a fixed order, from the input layer through the hidden layers to the output layer. This architecture is particularly well-suited for tasks where the input data has a clear temporal or sequential structure, such as in time series data, natural language processing, and speech recognition.

**Here are the key components of a Sequential Neural Network:**

Input Layer: This is the first layer of the network where the data is fed into the model. Each neuron in this layer represents a feature or input variable. The number of neurons in the input layer is determined by the dimensionality of the input data.

Hidden Layers: Between the input and output layers, there can be one or more hidden layers. These layers process the input data through a series of weighted connections and apply non-linear transformations using activation functions. The number of hidden layers and the number of neurons in each layer are parameters that can be tuned based on the complexity of the problem.

Output Layer: This is the final layer of the network that produces the model's predictions or classifications. The number of neurons in the output layer is determined by the nature of the task. For example, in binary classification, there may be one neuron (sigmoid activation function), while in multi-class classification, there may be multiple neurons (softmax activation function).

Connections (Edges): Each neuron in a layer is connected to every neuron in the subsequent layer. Each connection is associated with a weight, which is adjusted during the training process to optimize the model's performance.

Activation Functions: These are mathematical operations applied to the weighted sum of inputs in each neuron. They introduce non-linearity into the model, allowing it to learn complex relationships in the data. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, tanh, and softmax.

Sequential Flow: In a Sequential Neural Network, data flows through the layers in one direction, from the input layer to the output layer. There are no loops or feedback connections.

Backpropagation: This is the training algorithm used to adjust the weights in the network during the learning process. It involves computing the gradients of the loss function with respect to the weights and then updating the weights using an optimization algorithm like stochastic gradient descent (SGD).

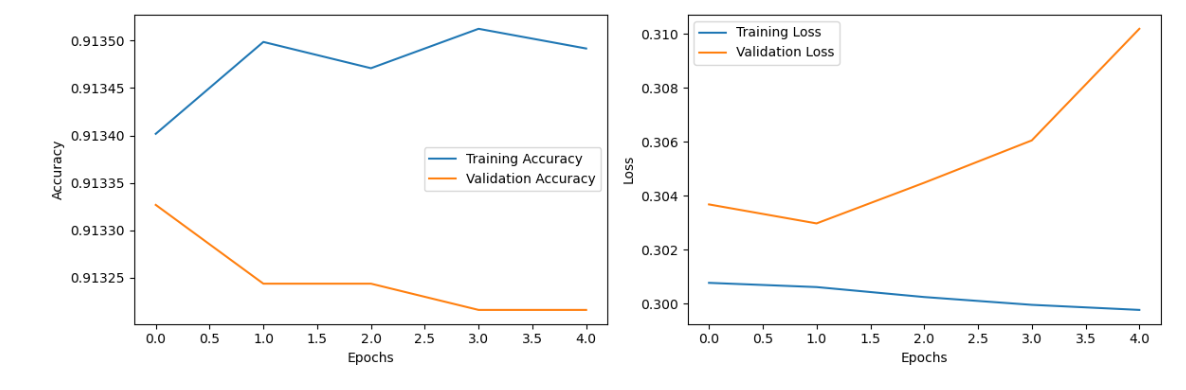
Sequential Neural Networks are widely used in various domains, including natural language processing, time series analysis, image recognition, and more. They have been instrumental in achieving state-of-the-art results in tasks like machine translation, sentiment analysis, and speech recognition. Additionally, variations of the Sequential Neural Network, such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have been developed to handle sequential data with longer dependencies.

**Conclusion:**

In conclusion, the developed sentiment classification model holds immense potential for enhancing the customer experience within the domain of online fashion retail. By accurately categorizing customer reviews into positive, negative, and neutral sentiments, retailers can gain invaluable insights into customer satisfaction levels and areas requiring attention. The model's proficiency in handling text data, coupled with its robust architecture, makes it a valuable asset for businesses seeking to optimize their online presence. With further fine-tuning and integration into existing systems, this model has the potential to revolutionize the way online retailers approach customer sentiment analysis.

Results :

Accuracy : 91%



The graphs shown in the code represent the training process and performance of the machine learning model over multiple epochs.

**Model Accuracy:**

The first graph shows the accuracy of the model during training and validation (testing) phases.

The x-axis represents the number of training epochs, which are complete passes through the entire training dataset.

The y-axis represents the accuracy of the model, which is the proportion of correctly classified samples.

The blue line represents the training accuracy, while the orange line represents the validation accuracy.

If the training accuracy is much higher than the validation accuracy, it may suggest overfitting, where the model is learning to perform well on the training data but does not generalize well to unseen data.

**Model Loss:**

The second graph shows the loss of the model during training and validation.

The x-axis again represents the number of training epochs.

The y-axis represents the loss, which is a measure of how well the model's predictions align with the actual labels.

The blue line represents the training loss, while the orange line represents the validation loss.

In general, you want to see a decrease in loss over epochs, indicating that the model is learning to make better predictions.

Interpreting these graphs is crucial for understanding how well the model is learning from the data. Ideally, you would like to see the training accuracy increase and the training loss decrease over epochs, which indicates the model is learning. However, it's equally important to monitor the validation accuracy and loss to ensure the model is not overfitting and can generalize well to new, unseen data. If the validation accuracy starts to plateau or decrease while the training accuracy continues to rise, it could be a sign of overfitting, and adjustments to the model or training process may be needed.