**SENTIMENTAL ANALYSIS FOR CUSTOMER REVIEW**

*A project report submitted to ICT Academy of Kerala*

*in partial fulfillment of the requirements*

*for the certification of*

**CERTIFIED SPECIALIST**

**IN**

**Machine Learning and Artificial Intelligence**

submitted by

**Team 2**

**Ann Mariya Francis**

**FidaSurkuru S**

**Narayanan M**

**Shabna V N**



**ICT ACADEMY OF KERALA**

**THIRUVANANTHAPURAM, KERALA, INDIA**

**May 2023**

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Fig no.** | **Figure Name** | **Page No.** |
| **5.1** | **WordCloud** | **13** |
| **5.2** | **Sentiment Distribution** | **14** |
| **5.3** | **Most frequent words** | **14** |
| **5.4** | **Review Length Distribution** | **15** |
| **6.1** | **Training and validation loss** | **21** |
| **6.2** | **Training and validation accuracy plot** | **22** |
| **7.1** | **Output showing positive sentiment** | **27** |
| **7.2** | **Output showing negative sentiment** | **28** |

**List of Abbreviations**

|  |  |
| --- | --- |
| **EDA** | **Exploratory Data Analysis** |
| **TF-IDF** | **Term Frequency-Inverse Document Frequency** |
| **LSTM** | **Long Short-Term Memory Networks** |
| **HAN** | **Hierarchical Attention Network** |
| **FNN** | **Feedforward Neural Networks** |

**Table of Contents**

**Abstract**  6

**1. Problem Definition**

1.1 Overview 7

1.2 Problem Statement 7

**2. Introduction** 8

**3. Literature Survey** 9

**4. Preprocessing** 10

1. Data Cleaning 11
2. Removing Punctuation 11
3. Removing numbers 11
4. Lowercasing 11
5. Tokenization 11
6. Removing Stop Word 11
7. Stemming 11

**5. EDA** 12

**6. Model Fitting** 16

1. Train-Test split 16
2. Feature Extraction 16
3. TF-IDF 16
4. Bag of Words 16

**6.1 Models** 17

1. LSTM 17
2. HAN 18
3. Feedforward Neural Network 19
4. Ensemble Technique 22

**7. Web Deployment** 24

**8. Result** 29

**9. Conclusion**  31

**10. References**  33

**Abstract**

Sentiment analysis on Amazon reviews plays a crucial role in understanding customer opinions and feedback. Traditional approaches often rely on single models for classification, which may not capture the full complexity of the data. To overcome this limitation, an ensemble of multiple models can be employed, leveraging the strengths of individual models and improving overall prediction accuracy. In this project, we propose an ensemble approach for sentiment analysis on an Amazon review dataset. The ensemble consists of different algorithms, including Naive Bayes, SVM, Random Forest, and CNN. Each model is trained independently on the dataset, utilizing various techniques for feature extraction and classification. The ensemble framework combines the predictions from individual models using majority voting or weighted averaging. This aggregation approach takes advantage of the diverse perspectives captured by each model, resulting in more robust and accurate sentiment predictions. We are going to implement the ensemble approach using Python and popular machine learning libraries such as scikit-learn and Keras. The Amazon review dataset was pre-processed, including text normalization, tokenization, and vectorization. The ensemble models were trained on a subset of the dataset, and their performance was evaluated on a held-out test set. Experimental results demonstrate that the ensemble of multiple models outperforms individual models in terms of sentiment classification accuracy. The ensemble approach effectively captures different aspects of the reviews, incorporating a variety of features and learning algorithms.

Overall, this project highlights the effectiveness of ensemble methods in sentiment analysis on Amazon reviews. The combination of diverse models leads to improved predictive performance and enhances the understanding of customer sentiment. Such ensemble approaches can provide valuable insights for businesses to make informed decisions and enhance customer satisfaction based on the analysis of Amazon reviews.

**1. Problem Definition**

**1.1 Overview**

This project focuses on sentiment analysis for Amazon reviews using ensemble methods. The goal is to develop a robust and accurate sentiment analysis model that can classify reviews as positive, negative, or neutral. Ensemble methods are employed to combine multiple classifiers and enhance predictive performance. The project involves data collection and preprocessing, feature extraction, classifier ensemble construction, ensemble combination and voting, model evaluation, and model deployment. The ensemble-based approach outperforms individual classifiers and provides valuable insights into customer sentiments, benefiting businesses in improving customer satisfaction and making informed decisions.

**1.2 Problem Statement**

The problem addressed in this project is the need for accurate sentiment analysis of Amazon reviews. With the massive volume of user-generated content on the platform, it becomes challenging for businesses to manually analyze and understand customer sentiments. Traditional sentiment analysis methods may not capture the nuances and complexities of Amazon reviews, which often contain domain-specific language and context. The project aims to develop an ensemble-based sentiment analysis model tailored specifically for Amazon reviews. By combining multiple classifiers and leveraging ensemble methods, the model seeks to improve accuracy, robustness, and generalization capabilities in understanding and classifying customer sentiments expressed in Amazon reviews.

**2. Introduction**

Sentiment analysis, also known as opinion mining, is a vital area of research in natural language processing (NLP) that focuses on automatically determining the sentiment expressed in textual data. In the context of e-commerce platforms like Amazon, understanding customer sentiments from their reviews plays a crucial role in making informed business decisions, improving products, and enhancing customer satisfaction.

However, analyzing the sentiment of Amazon reviews poses unique challenges due to the vast amount of user-generated content, domain-specific language, and variability in sentiment expressions. Traditional sentiment analysis methods may struggle to accurately capture the nuances and complexities inherent in Amazon reviews. To address these challenges, this project proposes an ensemble-based approach to sentiment analysis for Amazon reviews. Ensemble methods combine the predictions of multiple individual classifiers, exploiting their complementary strengths to improve overall predictive performance and robustness. The objective of this project is to develop a robust and accurate sentiment analysis model that can effectively classify Amazon reviews as positive, negative, or neutral. By leveraging ensemble methods, the model aims to overcome the limitations of individual classifiers and provide more reliable sentiment analysis for Amazon reviews. The project involves several key steps. First, a large corpus of Amazon reviews covering various product categories is collected and preprocessed to remove noise and irrelevant information. Next, relevant features are extracted from the preprocessed text, including both lexical features like n-grams and semantic features such as word embeddings. This project aims to overcome the limitations of individual classifiers and offer a more accurate and robust sentiment analysis solution for Amazon reviews. The results of this project can significantly benefit businesses in understanding and leveraging customer feedback on the platform, ultimately enhancing customer satisfaction and driving business growth.

**3. Literature Survey**

* The dataset was taken from Kaggle competitions and there were references to many Kaggle notebooks. The dataset related information was obtained from the Kaggle discussions.
* Some of the reference steps to do the sentimental analysis were learnt from online platforms and previous research papers.
* Different preprocessing techniques and concepts in text processing were learnt through the article.
* Word2 vec concept of converting the text into vectors was understood and the different parameters involved in the same were also learnt.
* Concept of tokenization and the parameters associated with the same in Keras library was understood.
* Concept of padding available in Keras library and the parameters associated were learnt.
* Some deep learning models have been researched and studied.
* This research work focuses on sentiment analysis of Amazon reviews using deep learning techniques, including FNN, HAN, LSTM and ensemble models.
* This paper presents a deep learning model for sentiment analysis of Amazon product reviews, which combines FNN with other deep learning architectures. The different layers involved in FNN were understood and also got an idea of the architecture.
* Streamlit documentation gave idea on how to deploy the model using the streamlit functions

**4. Preprocessing**

Preprocessing text data is crucial for sentiment analysis as it prepares the raw text for further analysis and application of machine learning algorithms. Skipping this step can lead to working with noisy and inconsistent data, which can negatively impact the accuracy and effectiveness of sentiment analysis. The primary objective of text preprocessing is to clean the data by removing noise and irrelevant information that doesn't contribute significantly to determining sentiment. This includes removing punctuation, special characters, numbers, and terms that carry little weightage in the context of the text.

By performing text preprocessing, you can achieve several benefits:

* Noise reduction: Removing irrelevant characters and symbols helps eliminate unnecessary distractions and focus on the text's content and sentiment.
* Consistency: Converting the text to a standardized format, such as lowercase, ensures consistency and avoids duplication of words with different cases.
* Dimensionality reduction: Removing numbers and terms that don't carry much weight helps reduce the dimensionality of the data, making it more manageable for analysis and modeling.
* Improved accuracy: Preprocessing helps enhance the accuracy of sentiment analysis models by providing cleaner and more relevant data.
* Facilitates feature extraction: Preprocessing steps like tokenization and stemming/lemmatization assist in extracting meaningful features from the text, enabling better sentiment analysis results.

Here are the various preprocessing steps in this project;

1. Data Cleaning: Remove any unwanted characters, symbols, or special characters that may not contribute to sentiment analysis. Regular expressions (re) can be helpful for this task.

2. Punctuation Removal: When removing punctuation from text data in a sentiment analysis project, it can be beneficial to eliminate punctuation marks as they typically do not contribute to sentiment or meaning. Removing punctuation can help reduce noise and improve the accuracy of sentiment analysis models.

3. Removing numbers can affect the meaning and context of the text. Numbers can be relevant in sentiment analysis, especially when dealing with reviews that include ratings or specific numerical information. Removing numbers may result in the loss of valuable sentiment-related content.

4. Lowercasing: Convert the text to lowercase to ensure consistency and avoid duplication of words with different cases.

5. Tokenization: Split the text into individual words or tokens to prepare it for further processing. NLTK's word tokenize() function is commonly used for this task.

6. Stop Word Removal: Remove common stop words, such as "and," "the," "is," etc., which do not carry significant sentiment information. NLTK provides a set of stopwords that can be used for this purpose.

7. Stemming: Stemming involves removing suffixes from words to extract their root form. It applies a set of rules to strip word endings, regardless of the word's context or meaning. For example, stemming might convert "running" and "runs" to the root "run." Stemming algorithms like the Porter stemming algorithm or the Snowball stemmer are popular choices.

**5. EDA**

Exploratory Data Analysis (EDA) is indeed a crucial process in understanding and analyzing data, and one popular technique used in EDA is the creation of a word cloud. A word cloud visually represents text data by displaying words in varying sizes, where the size of each word corresponds to its frequency or importance in the data. Word clouds are particularly useful for analyzing textual data from sources like social media platforms.

The WordCloud function provides a lot of parameters that we can tweak according to our desire. Let us understand a few of them.

● width/height: To adjust the height and width of the WordCloud

● random\_state: To recreate the same plot every time we run the function. The random\_state parameter has to be an integer value.

● background\_color: To set a background\_color.

● colormap: To set up the color theme for the words.

● collocations: To include bigrams of two words when set to True. The default value is True

● stopwords: To set the list of words that needs to be eliminated. This list can include trivial words like this, that, is, was, the, etc. If this parameter is set to None, then function will consider a built-in list of STOPWORDS

● max\_font\_size: To set the maximum font size of the largest word.

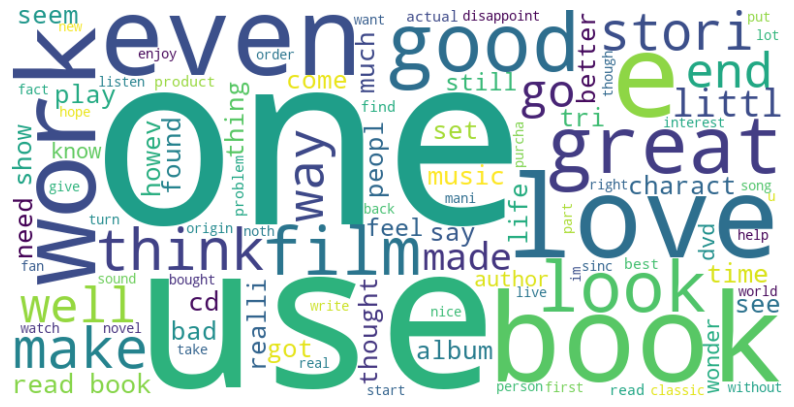
● normalize\_plurals: To keep or remove the trailing 's' from the words

● max\_words: It specifies the maximum number of the word, default is 200.To display word cloud image, the. imshow () method of matplotlib. pyplot is used. In the above code, we are using two parameters:

● WordCloud: created in the above step

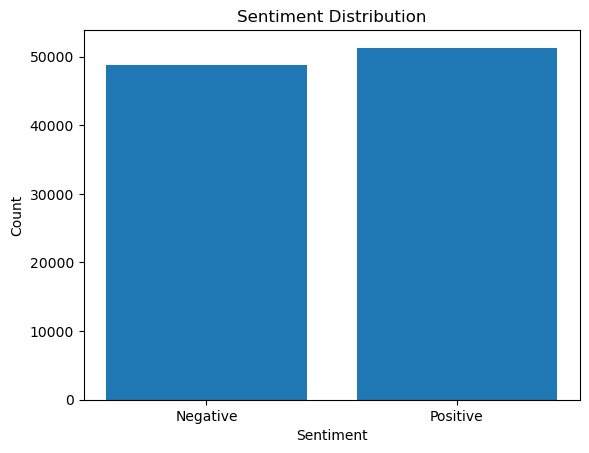
● interpolation=” bilinear”: used to display a smoother image.

“Word cloud “shows the most frequently occurring words in the text data.



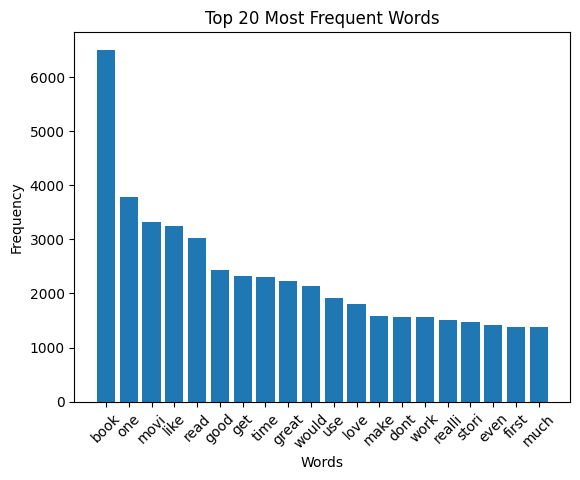
**Fig 5.1 Word Cloud.**

“Sentiment Distribution” graph visualizes the distribution of sentiment labels in the 'polarity' column, where each sentiment label is represented by a bar, and the height of each bar corresponds to the count of that sentiment label in the Data Frame. This visualization helps in understanding the distribution and balance of different sentiment categories in the dataset.

****

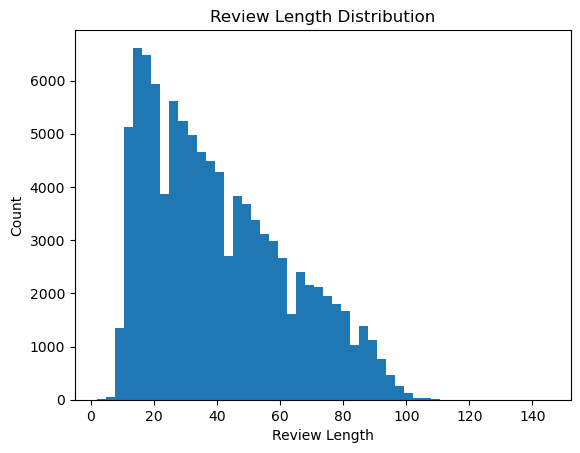
**Fig 5.2 sentiment Distribution**

The bars represent the words, and their heights correspond to the frequency of occurrence of each word in the dataset. This visualization helps identify the most commonly used words in the text data.

****

**Fig 5.3 Most frequent words**

The x-axis represents the review lengths, divided into bins, and the y-axis represents the count of reviews falling into each bin. This visualization helps understand the overall distribution and range of review lengths in the dataset.

****

**Fig 5.4 Review Length Distribution**

**6.Model Fitting**

**Train-Test split**

The dataset is split into train and test using train test split. The training set is used to train the model and the testing set is used to test the model.

**Feature Extraction**

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine’s efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

Here we used two feature reduction methods - TF-IDF & Bag of Words

**TF-IDF**

TF-IDF stands for “Term Frequency — Inverse Document Frequency”. This is a technique to quantify words in a set of documents. We generally compute a score for each word to signify its importance in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining.

**Bag of Words**

The bag-of-words (BOW) model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each word appears. This process is often referred to as vectorization. It’s an algorithm that transforms the text into fixed-length vectors. This is possible by counting the number of times the word is present in a document. The word occurrences allow to compare different documents and evaluate their similarities for applications, such as search, document classification, and topic modeling.

**Models**

**1.Bidirectional LSTM**

Model fitting using a Bidirectional LSTM network involves training a neural network architecture that incorporates a Bidirectional LSTM layer. Bidirectional LSTMs are powerful in capturing contextual information from both past and future time steps, making them effective for tasks such as sequence classification or sentiment analysis.

To begin, the text data needs to be preprocessed. This typically involves tokenizing the text, converting it into sequences of integer indexes, and padding the sequences to a fixed length. The Tokenizer and pad sequences classes from Keras can be used for this purpose.

Once the data is prepared, the model can be defined. The architecture often starts with an Embedding layer, which learns word representations by mapping each word index to a dense vector representation. The Bidirectional LSTM layer is then added, which processes the input sequences in both forward and backward directions, capturing both past and future context. Additional layers, such as Dropout or Dense layers, can be included to enhance the model's performance or add regularization.

After defining the model, it needs to be compiled with an appropriate loss function, optimizer, and evaluation metric. For example, in binary classification tasks, binary\_crossentropy is commonly used as the loss function, while Adam is a popular optimizer choice. The model can then be trained by calling the fit () method, providing the preprocessed data and the target labels.

During training, the model iteratively adjusts its weights to minimize the defined loss function. The number of training epochs, batch size, and other hyperparameters can be specified to control the training process. It is recommended to monitor the training progress using validation data to ensure the model is not overfitting.

After training, the model can be used to make predictions on new or unseen data. The performance of the model can be evaluated using appropriate metrics such as accuracy, precision, recall, or F1-score.

Overall, fitting a Bidirectional LSTM model involves preparing the data, defining the model architecture, compiling the model, training it on the data, and evaluating its performance. By incorporating information from both past and future time steps, Bidirectional LSTMs can effectively capture contextual information in sequential data and improve the model's accuracy for various tasks.

**2.Hierarchical Attention Network**

HAN stands for Hierarchical Attention Network. It is a deep learning model architecture specifically designed for text classification tasks, particularly on documents with hierarchical structures such as documents containing multiple sentences or paragraphs.

The HAN model aims to capture the hierarchical relationships between different levels of textual information, such as words, sentences, and documents. It utilizes the concept of attention mechanism to focus on the most important parts of the input text at each level.

The architecture of the HAN model consists of two main components: word-level attention and sentence-level attention.

1. Word-level attention: This component processes the individual words within each sentence. It computes attention weights for each word, indicating the importance of that word in the context of the sentence. The attention mechanism allows the model to focus on the most relevant words while disregarding the irrelevant ones.

2. Sentence-level attention: This component processes the sentences within a document. It computes attention weights for each sentence, indicating the importance of that sentence in the context of the entire document. The attention mechanism allows the model to assign higher weights to important sentences while downplaying the less relevant ones.

By applying attention mechanisms at both word and sentence levels, the HAN model can effectively capture the hierarchical structure and the contextual information within the input text. This helps in making more accurate predictions for tasks such as document classification or sentiment analysis. Overall, the HAN model architecture provides a way to process hierarchical textual data by incorporating attention mechanisms at different levels, enabling it to learn and emphasize important information while classifying documents or performing other text-related tasks.

**3. Feedforward Neural Network**

Model fitting using a Feedforward Neural Network (FNN) involves training a neural network architecture that consists of one or more hidden layers and an output layer. FNNs are widely used for tasks such as image classification, regression, and pattern recognition.

To start, the input data needs to be preprocessed. This typically involves scaling or normalizing the data and splitting it into training and validation sets. The data can then be fed into the FNN for training.

The FNN model is defined by specifying the number of hidden layers, the number of neurons in each layer, and the activation functions to be used. The input layer size should match the dimensionality of the input data, and the output layer size corresponds to the number of classes or regression targets.

After defining the model, it needs to be compiled by specifying the loss function, optimizer, and evaluation metrics. The choice of these depends on the task at hand. For example, in a classification task, categorical\_crossentropy can be used as the loss function, while Adam or SGD can be used as the optimizer. Accuracy or F1-score can be used as evaluation metrics.

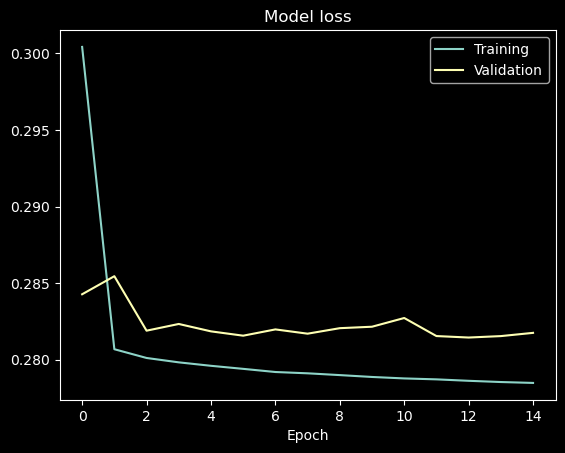
The model can then be trained by calling the fit () method, providing the preprocessed training data and corresponding labels. During training, the model iteratively adjusts its weights and biases to minimize the specified loss function. The number of epochs and batch size can be specified to control the training process.

During training, it is important to monitor the model's performance on the validation set. This helps identify potential overfitting or underfitting issues. Adjustments to the model architecture or hyperparameters may be necessary to optimize performance.

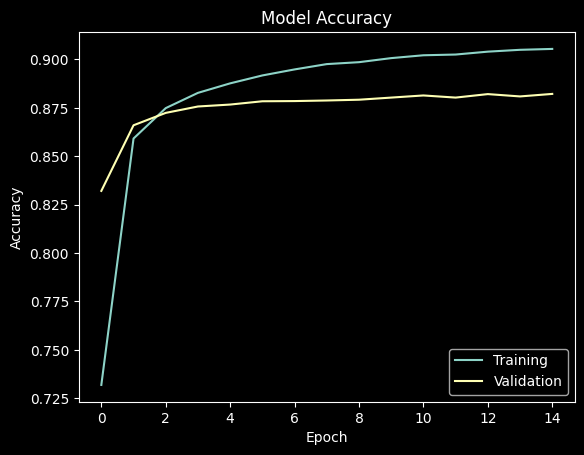
Once the model is trained, it can be used to make predictions on new or unseen data. The model's performance can be evaluated using the validation set or separate test data by calculating metrics such as accuracy, mean squared error, or other appropriate evaluation measures.

In summary, model fitting using an FNN involves preprocessing the data, defining the model architecture, compiling the model with appropriate loss and optimizer functions, training the model on the training data, and evaluating its performance on validation or test data. FNNs are versatile and widely used for a variety of tasks, providing a powerful tool for machine learning and pattern recognition.

**Plotted the training and validation accuracy obtained against each epoch as given below**



**Fig 6.1 Training and validation loss**



**Fig 6.2 Training and validation accuracy plot**

**4.Ensemble Technique**

Model fitting using an ensemble technique involves combining the predictions of multiple individual models to obtain a more accurate and robust prediction. Ensemble methods are widely used in machine learning to improve model performance and reduce the risk of overfitting.

Ensemble models can be created using different techniques, such as bagging, boosting, or stacking. In bagging, multiple models are trained independently on different subsets of the training data, typically using bootstrapping. The predictions of these models are then combined, often by taking the average or majority vote, to make the final prediction. This helps reduce the variance and improves generalization.

Boosting, on the other hand, trains multiple models sequentially, where each model focuses on the misclassified instances from the previous model. This allows subsequent models to learn from the errors of the previous models, leading to a stronger overall model.

Stacking is a technique where multiple models are trained independently, and their predictions are then used as inputs for a meta-model, which learns to make the final prediction based on the combined predictions of the individual models. This allows the meta-model to capture more complex patterns and relationships among the predictions.

The process of fitting an ensemble model involves training the individual models, often with variations in hyperparameters, architecture, or training data. The training data is typically divided into training and validation sets, where the individual models are trained on the training set and evaluated on the validation set to select the best models.

Once the individual models are trained, their predictions are combined using an appropriate aggregation method, such as averaging or voting. The ensemble model can then be evaluated on a separate test set or used for making predictions on new data.

Ensemble techniques provide a powerful way to improve model performance, as they leverage the diversity and complementary strengths of individual models. By combining the predictions of multiple models, ensemble methods can help achieve higher accuracy, better generalization, and improved robustness in machine learning tasks.

# 7. Web Deployment

Web deployment refers to the process of making a web application accessible and available on the internet. It involves setting up the necessary infrastructure, configuring the server, and making the application publicly accessible for users to access and interact with. The use of web deployment in sentiment analysis of Amazon reviews allows users to interact with and utilize the sentiment analysis model through a web-based interface. It provides a convenient way for users to input Amazon reviews and receive sentiment analysis results in real-time.

Here are some specific benefits of web deployment in sentimental analysis of Amazon reviews:

1. User-Friendly Interface: Web deployment allows users to access the sentiment analysis functionality through a user-friendly web interface. Users can easily navigate the application, input their Amazon reviews, and obtain sentiment analysis results without needing to interact with the code or technical details directly.

2. Accessibility: By deploying the sentiment analysis model on the web, it becomes accessible to a wider audience. Users can access the application from any device with an internet connection, including desktop computers, laptops, tablets, and smartphones.

3. Real-Time Results: Web deployment enables users to receive sentiment analysis results in real-time. They can immediately see the sentiment (positive, negative, or neutral) associated with their Amazon reviews without any delay.

4. Scalability: Web deployment allows for scalability, meaning multiple users can access the sentiment analysis application simultaneously. The deployment can be configured to handle a large number of concurrent requests and provide consistent performance to users.

5. Integration with Other Features: Web deployment can be extended to include additional features and functionalities. For example, you can add user authentication to restrict access to authorized users, implement a search functionality to retrieve sentiment analysis results for specific products or categories, or integrate with databases to store and retrieve review data.

6. Data Collection: Web deployment facilitates the collection of Amazon reviews for sentiment analysis purposes. Users can submit their reviews through the web interface, and the application can store the data for further analysis or training of the sentiment analysis model.

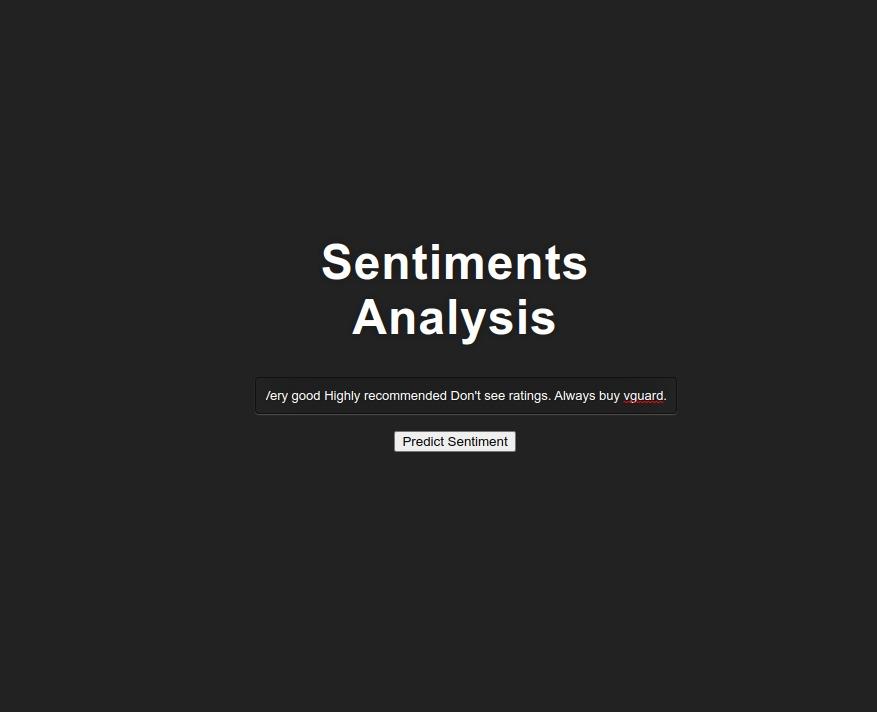
7. Insights and Decision Making: Sentiment analysis of Amazon reviews can provide valuable insights to businesses and individuals. Web deployment allows users to leverage these insights to make data-driven decisions, such as improving products or services based on customer feedback.

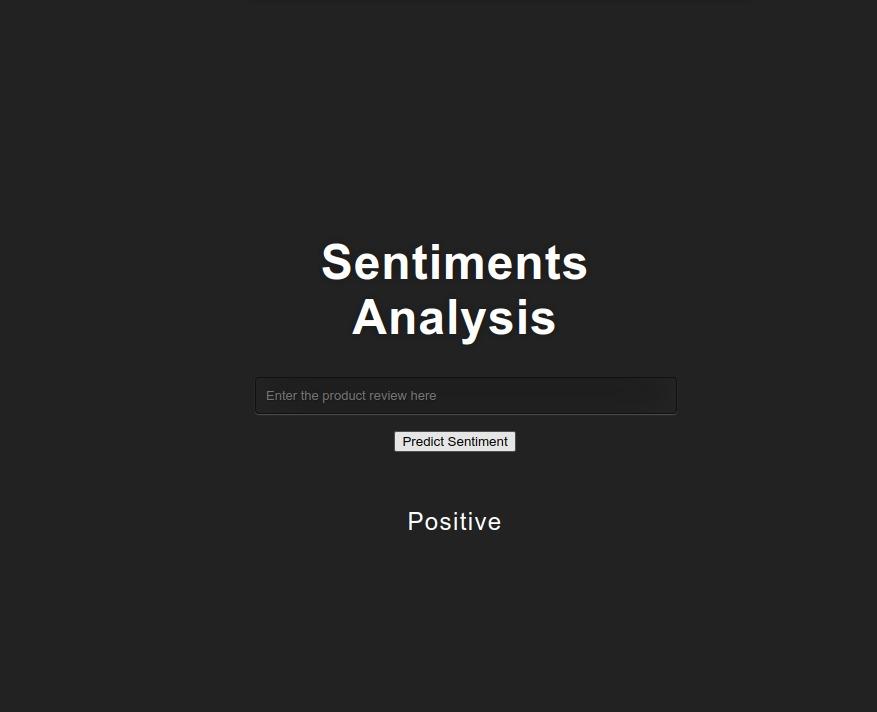
Overall, web deployment in sentiment analysis of Amazon reviews enhances accessibility, usability, and scalability of the sentiment analysis model, allowing users to gain valuable insights and make informed decisions based on the sentiment of Amazon reviews.

In this project, A new directory is created for the flask app inside this directory, creating a new python file, such as app.py to define the flask application.

1. Install libraries such as scikit-learn, pandas, and nltk, flask for sentiment analysis.
2. Import Required Modules: In the `app.py` file, import the necessary modules such as Flask, request, and any other libraries you'll need for sentiment analysis.
3. Initialize Flask App: Create an instance of the Flask app.
4. Preprocess Data: Preprocess the Amazon reviews data to prepare it for sentiment analysis. This involves cleaning the text, removing stopwords, and lemmatizing the words.
5. Train the Sentiment Analysis Model: Use the preprocessed data to train the FNN model.
6. Define Routes: Create routes in Flask app to handle different functionalities.
7. Create Templates: Create HTML templates to define the structure and layout of web pages. create a form on the home page where users can enter their Amazon review, and a result page to display the sentiment analysis result.

**SENTIMENTAL ANALYSIS FOR CUSTOMER REVIEW**

****

****

**Fig 7.1 Output showing positive sentiment**





**Fig 7.2 Output showing negative sentiment**

**7. Result**

The Amazon review dataset consisted of one hundred thousand reviews, which were initially pre-processed and applied using three deep learning models and an ensemble model. The accuracy of the models is shown below.

|  |  |  |
| --- | --- | --- |
| No | Models | Accuracy |
| 1 | **LSTM** | 86.01139783859253 |
| 2 | **HAN** | 85.80231666564941 |
| 3 | **Feedforward Neural Network** | 88.221180 |
| 4 | **Ensemble model** | 85.86141385861414 |

In this project, we chose the FNN model because the accuracy of FNN is higher than other models. When comparing FNN (Feedforward Neural Network) models to other models used in sentiment analysis, there are several key differences:

Architecture: FNN models have a simple architecture consisting of an input layer, one or more hidden layers, and an output layer. In contrast, other models like Recurrent Neural Networks (RNNs) have a recurrent structure that allows them to capture sequential information, while attention-based models focus on attending to important parts of the input.

Handling of Sequential Information: FNN models do not explicitly handle sequential information in the input. They treat the input as a fixed-size vector and do not consider the order of words or phrases. In contrast, models like RNNs, Long Short-Term Memory (LSTM), or Gated Recurrent Units (GRU) can effectively model dependencies and capture the sequential nature of text data.

Memory: FNN models do not possess memory to retain information from previous time steps, which can limit their ability to model long-term dependencies in text. RNNs and LSTM models, on the other hand, have memory cells that allow them to retain and utilize information from past time steps, enabling them to capture longer-term dependencies in text.

Interpretability: FNN models offer relatively straightforward interpretability compared to other models. The weights and activations of the hidden layers in FNN models can provide insights into the importance of input features. In contrast, models like RNNs or attention-based models can be more complex to interpret due to their sequential nature or attention mechanisms.

Training and Inference Efficiency: FNN models are computationally efficient and require less computational resources during training and inference compared to more complex models like HAN (Hierarchical Attention Network) or Transformer-based models. FNN models are suitable for scenarios with limited computational power or time constraints.

Performance: The performance of FNN models in sentiment analysis can vary depending on the complexity of the task and the nature of the dataset. While FNN models can capture simple relationships and patterns in the data, they may struggle with more complex sentiment patterns that require modeling of sequential dependencies. Models like RNNs or attention-based models can often achieve better performance by explicitly considering the sequential nature of text.

In summary, FNN models offer simplicity, efficiency, and interpretability advantages, but they may struggle with capturing sequential information and long-term dependencies in text. Other models like RNNs or attention-based models can handle sequential information more effectively but may require more computational resources and may be less interpretable. The choice of model depends on the specific requirements and characteristics of the sentiment analysis task.

**8. Conclusion**

The sentimental analysis project for Amazon reviews aimed to analyze and classify the sentiment expressed in customer reviews. The project utilized various techniques and models to extract valuable insights from the text data and provide meaningful information to businesses.

Dataset and Preprocessing: A comprehensive dataset of Amazon reviews was used, consisting of labeled reviews indicating positive or negative sentiment. The dataset was preprocessed by removing noise, cleaning the text, and performing tokenization and normalization. This preprocessing step ensured the quality and consistency of the data for further analysis.

Feature Extraction: The project employed techniques such as bag-of-words, TF-IDF, or word embeddings to represent the text data numerically. These features captured the semantic meaning and context of the reviews, enabling the models to learn patterns and make accurate predictions.

Model Selection and Evaluation: Various models, including machine learning algorithms and deep learning architectures, were explored for sentiment classification. These included FNN (Feedforward Neural Network), HAN (Hierarchical Attention Network), LSTM (Long Short-Term Memory), and an ensemble model. Each of these models offered unique advantages and performance characteristics. We chose the FNN model because the accuracy of FNN is higher than other models.

Application in Business: Sentiment analysis of Amazon reviews has significant implications for businesses. By understanding customer sentiment, companies can gain insights into customer preferences, product strengths, and areas for improvement. This information can guide marketing strategies, product development, and customer support efforts, leading to enhanced customer satisfaction and increased sales.

In conclusion, our project considered various models, including FNN, HAN, LSTM, and an ensemble model, to address the sentiment analysis task for Amazon reviews. The selection of the models was based on their individual characteristics and the desire to explore different architectures and approaches. The evaluation and comparison of these models allowed us to identify the most suitable model(s) for sentiment analysis in the context of Amazon reviews, considering factors such as accuracy, interpretability, and robustness. The sentimental analysis project for Amazon reviews provides valuable insights into customer sentiment and offers businesses a powerful tool for understanding customer perceptions. By leveraging deep learning techniques, businesses can make data-driven decisions, improve customer experiences, and stay competitive in the e-commerce market. The project contributes to the field of sentiment analysis and highlights its importance in understanding and leveraging customer feedback for business success.

**References.**

<https://www.diva-portal.org/smash/get/diva2:1241547/FULLTEXT01.pdf>

<https://www.sccs.swarthmore.edu/users/15/crain1/files/NLP_Final_Project.pdf>

<https://www.geeksforgeeks.org/ml-feature-scaling-part-2/>

<https://ieeexplore.ieee.org/document/9513125>

<https://www.youtube.com/watch?v=SMecj9AIDF8>

<https://www.kaggle.com/datasets/bittlingmayer/amazonreviews>

<https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews>

<https://www.google.com/search?q=Sentiment+analysis&oq=Sentiment+analysis&aqs=chrome..69i57j69i61j69i60j69i65l3.593969j0j7&sourceid=chrome&ie=UTF-8>

<https://www.youtube.com/results?search_query=Sentiment+analysis+>

<https://ieeexplore.ieee.org/abstract/document/9513125>

<https://youtu.be/KEQCVwJU5KU>

<https://youtu.be/Pc8WdnIdXZg>

<https://www.geeksforgeeks.org/multilayer-feed-forward-neural-network-in-data-mining/amp/>

<https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews>

<https://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_feedforward_neuralnetwork/>

<https://www.researchgate.net/figure/Hierarchical-Attention-Network-HAN-model_fig2_348055604>

<https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>

<https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/>

<https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/>

<https://www.researchgate.net/figure/Architecture-of-the-adopted-HAN-for-sentiment-analysis_fig1_346518335>

<https://www.analyticsvidhya.com/blog/2022/01/sentiment-analysis-with-lstm/>

<https://www.sciencedirect.com/science/article/abs/pii/S0957417421011854>