TDS3851
MACHINE LEARNING

SENTIMENT ANALYSIS ON E-COMMERCE WOMEN'S CLOTHING REVIEWS

GROUP ID: ML1_T2220

Student Name	Student ID	Specialization	Contribution
Jennifer Lo Foh Wei	1211309776	Data Science	CNN
Amin Ahmed	1191302190	Data Science	RNN
Obai Ali	1171103208	Data Science	FNN

SEMESTER / SESSION: TRIMESTER 2, 2022/2023

FACULTY OF COMPUTING AND INFORMATICS
MULTIMEDIA UNIVERSITY

10 JULY 2023

(i) Abstract

This report examines sentiment analysis on the Women's Clothing E-Commerce Reviews dataset using machine learning. In our study, we employed Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Feedforward Neural Network (FNN) models. The preprocessing steps involved exploration of the data, removal of punctuation and conversion to lowercase, tokenization, stop word removal, lemmatization, and removal of non-English words. The models were evaluated based on their performance, with detailed comparison presented in the report. The results underline the effectiveness of deep learning techniques for sentiment analysis in the e-commerce sector, having substantial implications for businesses looking to extract value from customer feedback data.

Table of Contents

List of Figures

List of Tables

(ii) Introduction

Sentiment analysis has become a key tool in the field of data science and machine learning, aiding businesses in understanding consumer sentiments towards their products or services. It enables the automated identification and categorization of opinions expressed in text data, providing valuable insights to improve business strategies and customer experience.

In this project, our research centers around the Women's Clothing E-Commerce Reviews dataset. We aim to implement sentiment analysis using three different machine learning models: Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Feedforward Neural Network (FNN). This choice of models allows us to explore a spectrum of architectures and methodologies, ranging from sequential data processing (RNN) to hierarchical feature extraction (CNN), and simple connections of neurons (FNN).

The rationale behind our research is based on the understanding that customer reviews serve as a rich source of data, providing honest and immediate feedback on product quality, delivery, and customer service. However, given the sheer volume of reviews, manual analysis becomes unfeasible, and automated techniques such as sentiment analysis emerge as the solution. We hypothesize that machine learning models can effectively classify sentiment in reviews, allowing for efficient and insightful analysis.

This report is organized as follows: the next section presents a detailed description of the dataset used. We then explain the data preprocessing methods employed, followed by a thorough exposition of the research methodology and models used. We also discussed the initialization, training, and tuning of model hyperparameters. Subsequently, we evaluate and compare the performance of the models and discuss the results. Finally, we conclude the report with a summary of our findings and suggestions for future research in this area.

(iii) Dataset Description

The dataset for our project comes from Kaggle and is titled “Women's E-commerce Clothing Reviews.”

(https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews).

This dataset comprises 23,486 entries of customer reviews for women's clothes. The data is organized into 11 columns that encompass various characteristics. The columns include:

Column	Description
Clothing_ID	Identifier for the specific clothing item.
Age	Age of the reviewer
Title	Title of the review
Review_Text	Text of the review
Rating	Rating given by the customer (from 1 to 5)
Recommended_IND	Indicator if the customer recommends the product (1 recommend, 0 not recommend)
Positive_Feedback_Count	Count of other customers who found this review positive
Division_Name	Name of the clothing division
Department_Name	Name of the department of product
Class_Name	Name of the class of product

The dataset is a mixture of integer and object data types, as illustrated in the information summary. Here is the snapshot of data info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23486 entries, 0 to 23485
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0                            23486 non-null  int64
 1   Clothing_ID              23486 non-null  int64
 2   Age                      23486 non-null  int64
 3   Title                    19676 non-null  object
 4   Review_Text              22641 non-null  object
 5   Rating                   23486 non-null  int64
 6   Recommended_IND          23486 non-null  int64
 7   Positive_Feedback_Count  23486 non-null  int64
 8   Division_Name            23472 non-null  object
 9   Department_Name          23472 non-null  object
 10  Class_Name               23472 non-null  object
dtypes: int64(6), object(5)
memory usage: 2.0+ MB
```

Figure 1: Data Columns

For this project, the main focus will be on the 'Review_Text' and ‘Recommended_IND’ columns, as these provide the text data and sentiment score, respectively, which are needed for sentiment analysis.

The percentage of recommended and non-recommended reviews in the dataset are shown in a pie chart. 17.76% of the reviewers do not recommend the product they have bought, while 82.24% of the reviewers were satisfied with the product and would recommend others to buy.

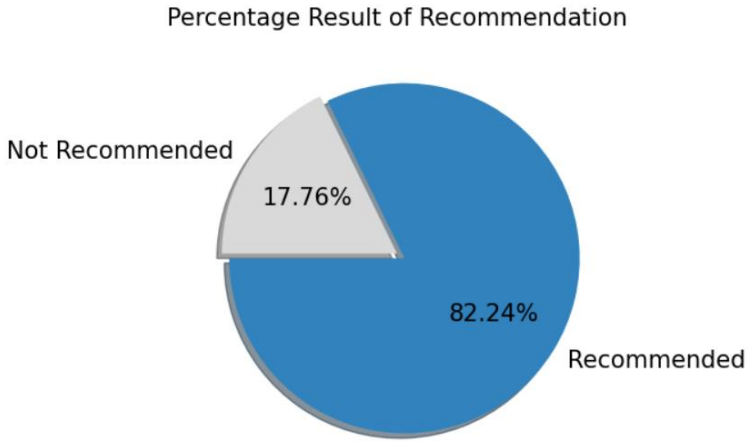

Figure 2: Pie Chart by 'Recommended_IND'

The data has been divided into three parts: training, validation, and testing datasets. The exact proportions of the split and additional processing steps will be covered in the next section.

(iv) Pre-processing Methods

Before employing machine learning models, we performed several preprocessing steps to clean and structure our text data, ensuring that it was in a suitable format for analysis.

1. **Exploring the data:** Initially, we examined the dataset to understand the features and their characteristics. This step helped us to identify relevant features for our analysis, which were primarily 'Review_Text' and 'Recommended_IND'.

```
[21] df = df.loc[:, df.columns.intersection(['Review_Text','Recommended_IND'])]
     print(f'Updated Shape of Data: {df.shape}')

     Updated Shape of Data: (23486, 2)

[22] df.reset_index(drop=True, inplace=True)
```

Figure 3: Code for Data Exploration

2. **Handling missing values:** A key step in the data preprocessing for our sentiment analysis project was managing missing values. In our dataset, we had missing data points that needed addressing to ensure the integrity of our machine learning models.

```
[23] #check missing value
     nullValue = df.isnull().any()
     nullValue

     Review_Text         True
     Recommended_IND    False
     dtype: bool

[24] # Check missing values
     df.isnull().sum()

     Review_Text        845
     Recommended_IND      0
     dtype: int64

[25] # total missing values
     t_cells = np.product(df.shape)
     t_missing = df.isnull().sum().sum() # total missing value for all the variable
     # percentage of missing data
     percent_missing = (t_missing/t_cells) * 100
     print("Percentage of missing value: {:.2f}%".format(percent_missing))

     Percentage of missing value: 1.80%

[26] # Drop out missing values for all columns
     df = df.dropna(how='any')
```

Figure 4: Code for Missing Values Handling

3. **Remove Punctuation and Convert to Lower-case:** The second cleaning step involved removing all punctuation from the review text as they do not contribute to sentiment. We used the string.punctuation function in Python for this purpose. After this, we converted all the text into lower-case to maintain uniformity and to ensure that the same words in different cases are not counted as separate words.

```
[31] def text_clean(text):
         text = text.lower()
         text = re.sub('\[.*?\]', '', text)
         text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
         text = re.sub('\w*\d\w*', '', text)
         text = re.sub('[''""…]', '', text)
         text = re.sub('\n', '', text)
         text= text.strip()

         return text

     cleaned = lambda x: text_clean(x)

[32] df['New_Review_Text'] = df['Review_Text'].apply(cleaned)
     df.head(10)
```

Figure 5: Code for Text Preparation

4. **NLTK Tokenization:** We then tokenized the text data using the Natural Language Toolkit (NLTK). Tokenization involves splitting a large paragraph into sentences or words. In this case, we broke down each review into individual words.

```
[33] def tokens(text):
         tokens = nltk.word_tokenize(str(text))
         # take only words (not punctuation)
         token_words = [w for w in tokens if w.isalpha()]
         return token_words

     cleaned1 = lambda x: tokens(x)

[34] df['New_Review_Text'] = df['New_Review_Text'].apply(cleaned1)
     df.head(10)
```

Figure 6: Code for Tokenization

5. **Removing Stopwords:** Stopwords like 'is', 'the', 'and', etc., appear very frequently in all types of text. They don't offer any useful information, so we removed them using NLTK's list of English stopwords.

```
[35] stop_words = set(stopwords.words('english'))

     def remove_stopwords(text):
       remove = [w for w in text if w not in stop_words]
       return remove

     cleaned2 = lambda x: remove_stopwords(x)

[36] df['New_Review_Text'] = df['New_Review_Text'].apply(cleaned2)
     df.head(10)
```

Figure 7: Code for Stop Words Removal

6. **Lemmatization:** We performed lemmatization to reduce inflectional forms of a word into a common base or root, using NLTK's WordNetLemmatizer. For instance, 'running', 'runs', 'ran' are all forms of the word 'run', so they're all converted to 'run'.

```
[37] # Lemmatize the verb
     def lem1(text):
         wordnet = WordNetLemmatizer()
         lemma_words = []
         for w in text:
           lemma_words.append(wordnet.lemmatize(w,'v'))

         return lemma_words

     cleaned3 = lambda x: lem1(x)

[38] df['New_Review_Text'] = df['New_Review_Text'].apply(cleaned3)
     df.head(10)
```

Figure 8: Code for Verb Lemmatization

```
[39] # Lemmatize the nouns
     def lem2(text):
         wordnet = WordNetLemmatizer()
         lemma_words = []
         for w in text:
           lemma_words.append(wordnet.lemmatize(w,'n'))

         return lemma_words

     cleaned4 = lambda x: lem2(x)

[40] df['New_Review_Text'] = df['New_Review_Text'].apply(cleaned4)
     df.head(10)
```

Figure 9: Code for Noun Lemmatization

7. **Remove Non-English Words:** Finally, we removed non-English words from the reviews. This was to ensure that all the text we are analyzing is in English and relevant to our context.

```
[41] dictionary = enchant.Dict("en_US")

     def remove_non_english(text):
       remove = []
       for w in text:
         if dictionary.check(w):
           remove.append(w)

       return remove

     cleaned5 = lambda x: remove_non_english(x)

[42] df['New_Review_Text'] = df['New_Review_Text'].apply(cleaned5)
     df.head(10)
```

Figure 10: Code for Non-English Words Removal

This structured and cleaned data was then ready for our sentiment analysis models.

(v) Research Methodology and Models

Our research methodology for sentiment analysis on the Women's Clothing E-Commerce Reviews dataset can be divided into several distinct steps:

1. **Dataset Collection:** The dataset was collected from Kaggle and loaded into a pandas DataFrame for analysis and preprocessing.

2. **Pre-processing:** As described in the previous section, the collected data underwent several preprocessing steps, including removing punctuation, converting to lowercase, tokenizing, removing stop words, lemmatizing, and eliminating non-English words.

3. **Training-Validation-Test Split:** After preprocessing, the dataset was partitioned into three parts - training, validation, and test datasets. This division was done to train our models, tune hyperparameters, and finally, evaluate the models' performance.

4. **Model Selection and Construction:** We selected three different types of neural network models for our sentiment analysis task - Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Feedforward Neural Network (FNN). Each model type has its architectural characteristics suitable for different types of data:

    - **Recurrent Neural Networks (RNNs):** RNNs are suitable for sequence data, like our text reviews, as they have 'memory' that captures information about what has been calculated before.
    - **Convolutional Neural Networks (CNNs):** CNN can extract an area of features from global information, and it is able to consider the relationship among these features. It has a convolutional layer to extract information from a larger piece of text.
    - **Feedforward Neural Networks (FNNs):** FNNs are the simplest type of artificial neural network. In FNN the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any), and to the output nodes. There are no cycles or loops in the network, which can make them less suitable for dealing with sequence data like text, but their simplicity and efficiency make them a good first model to try for many problems.

5. **Training and Hyperparameter Tuning:** The models were trained on our dataset and several hyperparameters were tuned to improve performance and avoid overfitting. Hyperparameters such as learning rate, batch size, number of epochs, and the number of layers were fine-tuned.

6. **Testing and Performance Comparison:** The trained models were tested on the test dataset. The performance of each model was compared using evaluation metrics like accuracy, precision, recall, and F1-score.

7. **Result Analysis and Report Writing:** Finally, results were analyzed, and findings were reported.

The following flowchart provides an overview of the research methodology:

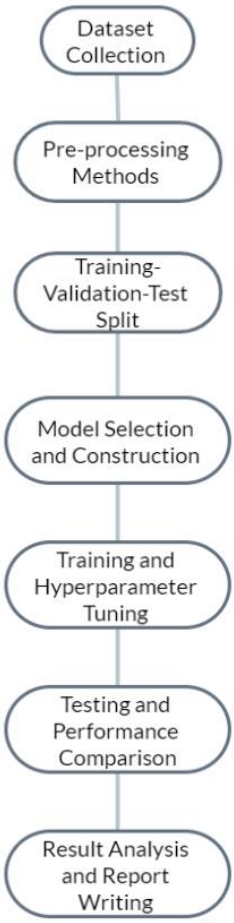

Figure 11: Flowchart of Research Methodology

Each model was compiled using an appropriate optimizer and the binary cross-entropy loss function since our task was binary classification (positive or negative sentiment). We used the accuracy metric to evaluate the models' performance.

RNN Architecture:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_1 (SimpleRNN)    (None, 160)               1138720

 dense_4 (Dense)             (None, 160)               25760

 dense_5 (Dense)             (None, 160)               25760

 dense_6 (Dense)             (None, 160)               25760

 dense_7 (Dense)             (None, 160)               25760

 dense_8 (Dense)             (None, 1)                 161

=================================================================
Total params: 1,241,921
Trainable params: 1,241,921
Non-trainable params: 0
_________________________________________________________________
```

Figure 12: RNN Model Summary

CNN Architecture:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 1000, 100)         100000

 conv1d_2 (Conv1D)           (None, 996, 96)           48096

 global_max_pooling1d_2 (Glo  (None, 96)               0
 balMaxPooling1D)

 dense_2 (Dense)             (None, 1)                 97

=================================================================
Total params: 148,193
Trainable params: 48,193
Non-trainable params: 100,000
_________________________________________________________________
```

Figure 13: CNN Model Summary

FNN Architecture:

```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_61 (Dense)            (None, 274)               1974170

 dense_62 (Dense)            (None, 90)                24750

 dense_63 (Dense)            (None, 1)                 91

=================================================================
Total params: 1999011 (7.63 MB)
Trainable params: 1999011 (7.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 14: FNN Model Summary

(vi) Initialization, Training, and Tuning of Hyperparameters

- **Recurrent Neural Networks (RNNs):** The Recurrent Neural Network (RNN) model, utilized in our project, has a simple yet effective architecture. The model's construction starts with a SimpleRNN layer, consisting of 160 units. This layer has an output shape of (None, 160), where 'None' signifies a flexible batch size and '160' represents the dimensionality of the output space. Following this, there are multiple Dense layers, each having the same number of units (160), resulting in the output shape (None, 160). The final layer is a Dense layer with a single unit responsible for performing binary classification, enabling the model to distinguish between positive and negative sentiment. The model boasts a substantial 1,241,921 parameters, all of which are trainable.

  Before training the RNN model, we pre-processed the text data utilizing the Bag-of-Words (BoW) technique. This method transformed the text into a machine-readable format, constructing a 'vocabulary' of unique words present in the dataset and subsequently representing each review based on the frequency of these words.

  In the training phase, we used the Stochastic Gradient Descent (SGD) optimizer, which was further fine-tuned for optimal learning rate and momentum via hyperparameter tuning. To avoid the common pitfall of overfitting, we implemented an early stopping mechanism during the training process, which was carried out for a certain number of epochs.

  The training process of the RNN model was supplemented by addressing the issue of class imbalance using the Synthetic Minority Over-sampling Technique (SMOTE). This technique bolstered the robustness and generalizability of our model by creating synthetic minority class samples, thereby ensuring the model does not display a bias towards the majority class.

  In summary, our sentiment analysis approach on the Women's Clothing E-Commerce Reviews dataset incorporated BoW for data representation, the RNN model for executing the task, and SMOTE for managing class imbalance.

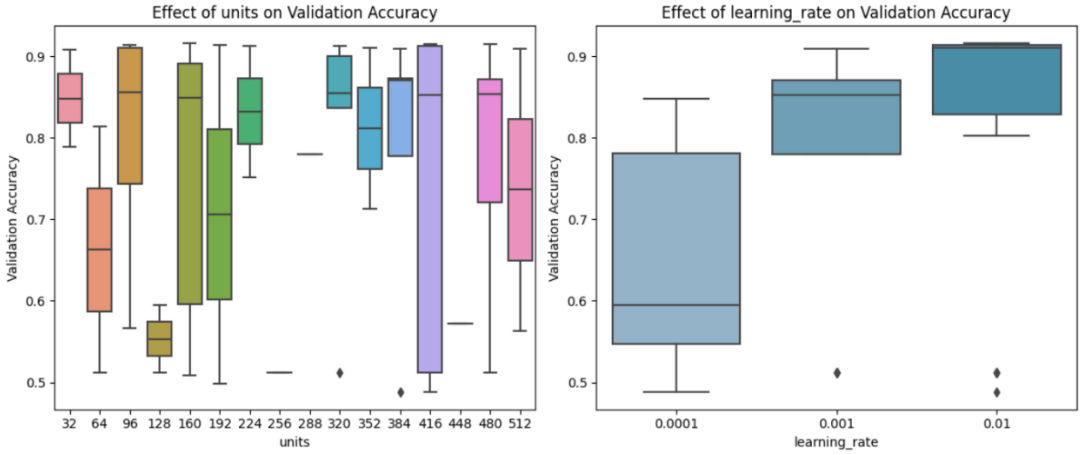

Figure 15: Effect of units and learning rate on Validation Accuracy (RNN)

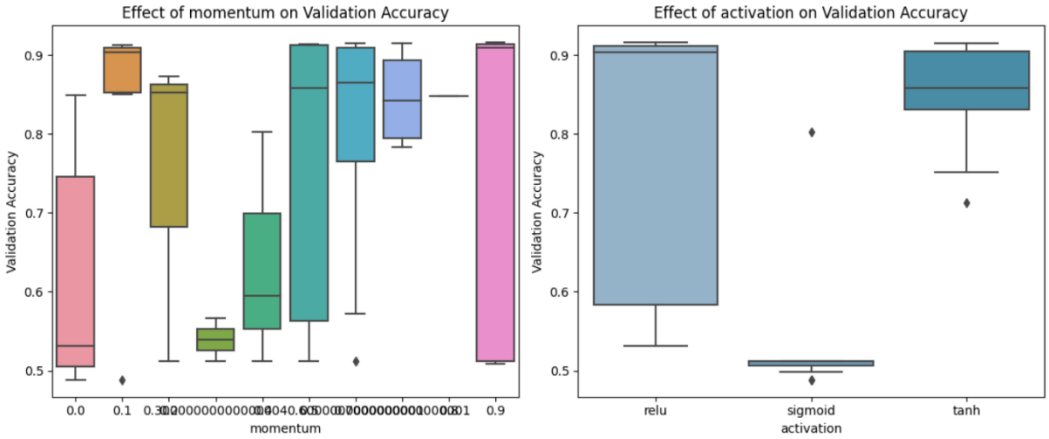

Figure 16: Effect of momentum and activation on Validation Accuracy (RNN)

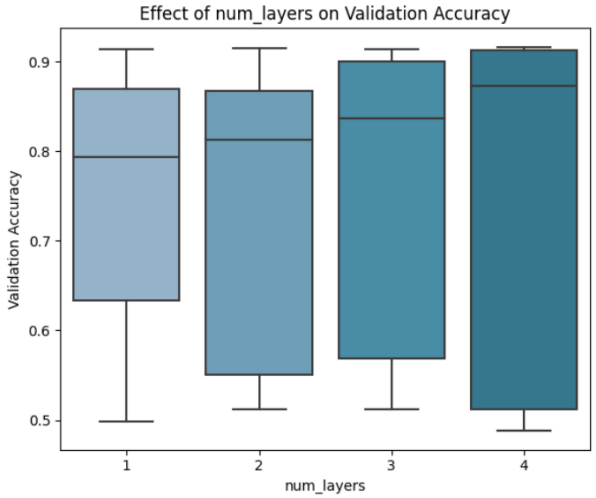

Figure 17: Effect of number of layers on Validation Accuracy (RNN)

- **Convolutional Neural Networks (CNNs):**

  The Convolutional Neural Network (CNN) model's construction starts with an embedding layer, consisting of 100,000 trainable parameters. This layer has an output shape of (None, 1000, 100), where 'None' signifies a flexible batch size, ‘1000’ represents the fixed number of words in each review and ‘100’ is a 100-dimensional dense vector representing each word in the input sequence. Following this, there is a convolutional layer, which reduces the sequence length from 1000 to 996, because of the convolution process and padding. ‘96’ refers to the number of filters learned by the convolutional layer in order to be utilized in detecting patterns in the input sequence. The output data of the convolutional layer is then passed to the global max pooling layer to reduce the spatial dimensions of the input tensor. The final layer is a Dense layer with a single unit responsible for performing binary classification, enabling the model to distinguish between positive and negative sentiment. The model boasts a substantial 148,193 parameters, only 48,193 are trainable.

  The text input was preprocessed using the Global Vectors for Word Representation (GloVe) algorithm before training the CNN model. This approach will result in a high-dimensional space where each word in the text is represented by a real-valued

vector. Words with similar meanings will have equivalent representations in the vector space because of how vectors are learned.

During the training phase, the CNN model was trained using the SGD optimizer, just like the RNN model. The application of SGD The high cost of executing back propagation over the entire training set motivates this in the neural network setup. SGD can overcome this cost while still achieving rapid convergence.

SGD optimizer has been used for the hyperparameter tuning for the CNN model. The hyperparameters that have been tuned were number of filter, kernel size, learning rate, momentum and activation function.

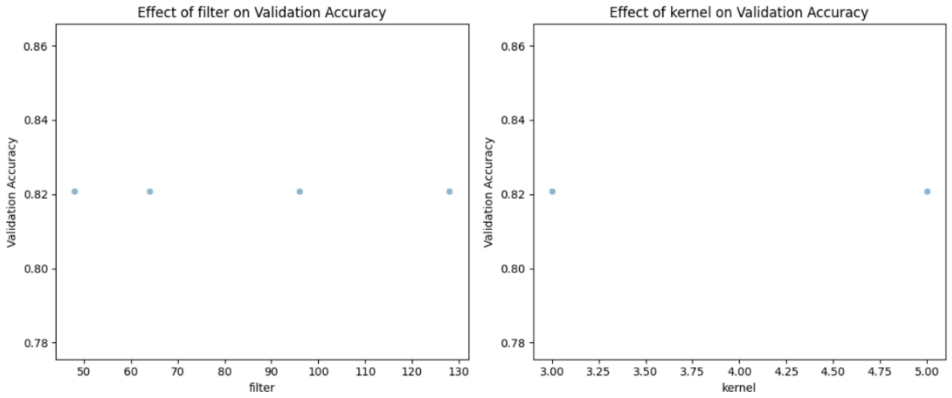

Figure 18: Effect of filter and kernel on Validation Accuracy (CNN)

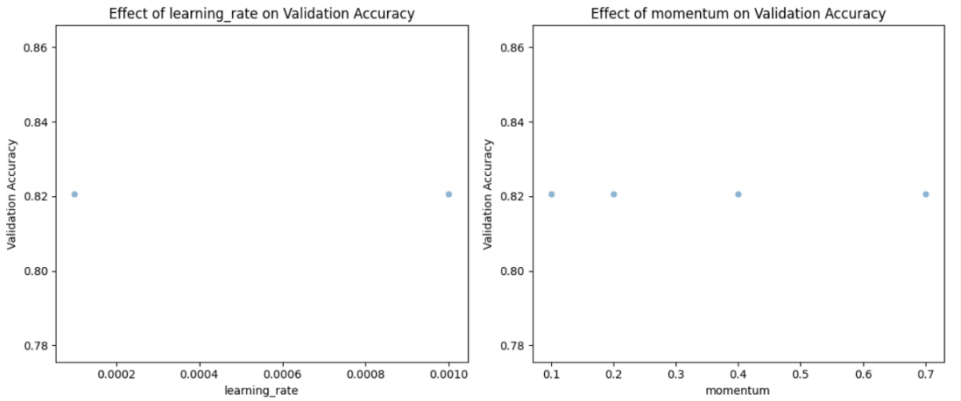

Figure 19: Effect of learning_rate and momentum on Validation Accuracy (CNN)

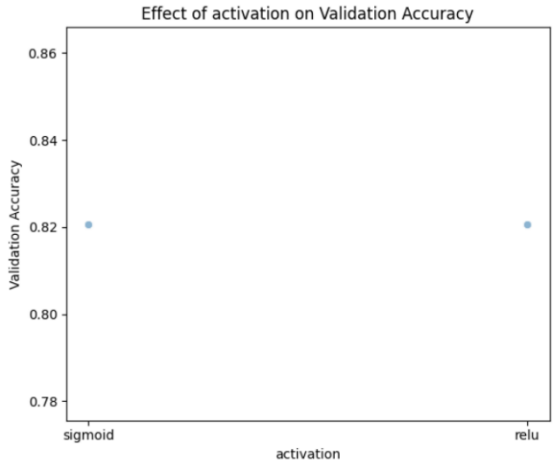

Figure 20: Effect of activation on Validation Accuracy (CNN)

- **Feedforward Neural Networks (FNNs):**
  The Hyperopt library, a Python library for optimizing over hyperparameters, was used. Hyperopt is more flexible than grid search or random search as it uses the Bayesian Optimization technique to more effectively sample the hyperparameter space.
  The hyperparameters tuned included:
  - The number of units in the first Dense layer (units_1): Values were chosen from the range 50-300.
  - The number of units in the second Dense layer (units_2): Values were chosen from the range 25-150.
  - The learning rate for the Adam optimizer (learning_rate): Values were sampled from a log-uniform distribution ranging from 1e-5 to 1e-2.

The Hyperopt optimization process, using the Tree of Parzen Estimators (TPE) algorithm, was run for 10 trials. The objective function to minimize was the negative of accuracy, as we were interested in maximizing accuracy.
The best hyperparameters obtained were:

units_1: 224
units_2: 65
learning_rate: `0.07685642546643698`

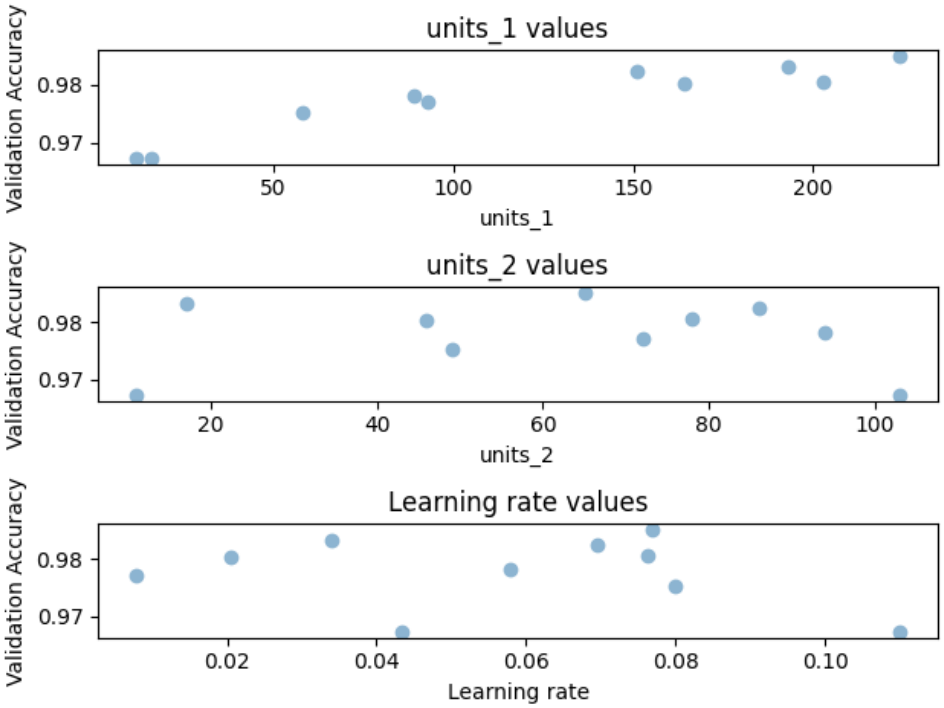

Figure 21: Effect of Units and learning_rate on Validation Accuracy

(vii) Performance evaluation, comparison, and discussion on the results

Once our RNN, CNN, and FNN models were trained, we performed a comprehensive performance evaluation to understand their effectiveness and suitability for the task at hand. The key metrics we used to measure performance were Accuracy, Precision, Recall, and F1-score.
Accuracy provided us with a general measure of how often the model was correct. Precision told us the proportion of positive identifications that were actually correct, while Recall gave us the proportion of actual positives that were correctly identified. Lastly, the F1-score provided a balance between Precision and Recall.

Throughout training, we monitor these values to check if the model is learning effectively. Ideally, we expect to see accuracy increase (and loss decrease) with each epoch for both the training and validation sets:

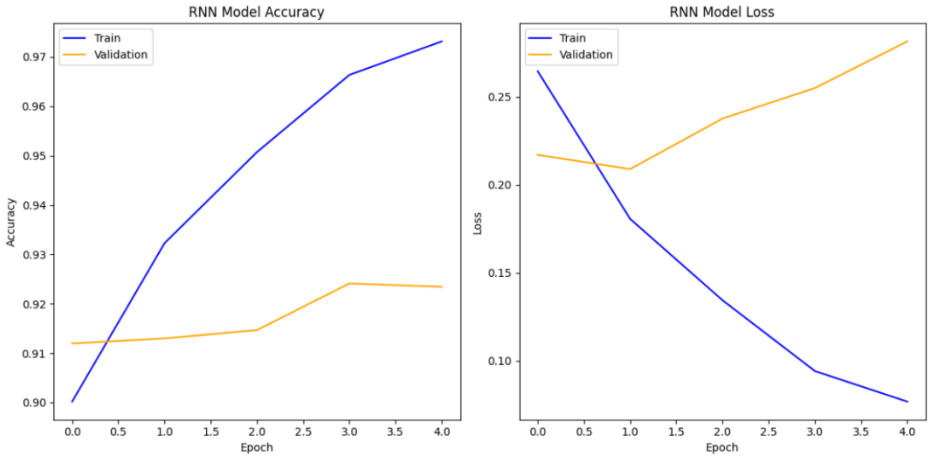

Figure 22: RNN Model Accuracy

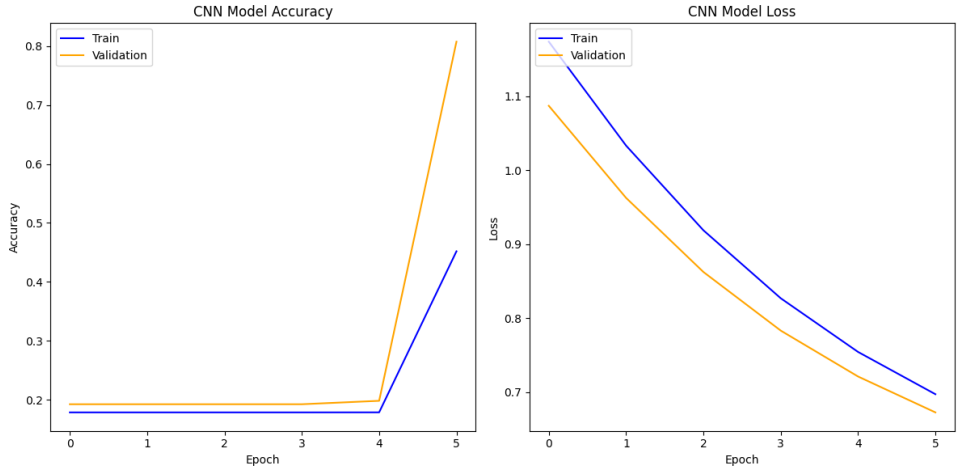

Figure 23: CNN Model Accuracy

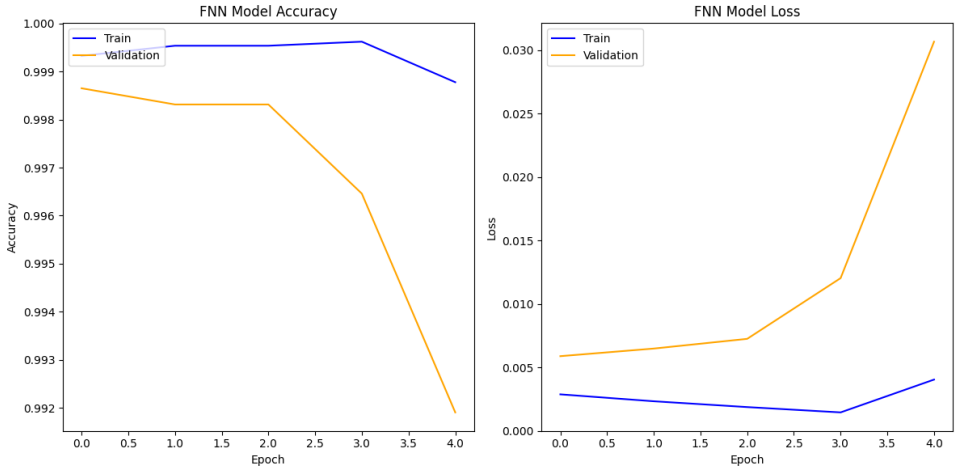

Figure 24: FNN Model Accuracy

Our study aimed to build models capable of predicting the sentiment of the Women's Clothing E-Commerce Reviews. We compared the performance of three different neural network architectures: a Recurrent Neural Network (RNN), a Convolutional Neural Network (CNN), and a Feed-forward Neural Network (FNN).

We assessed each model's performance based on four key metrics: Accuracy, Recall, Precision, and F1 Score. Here is an overview of the results:

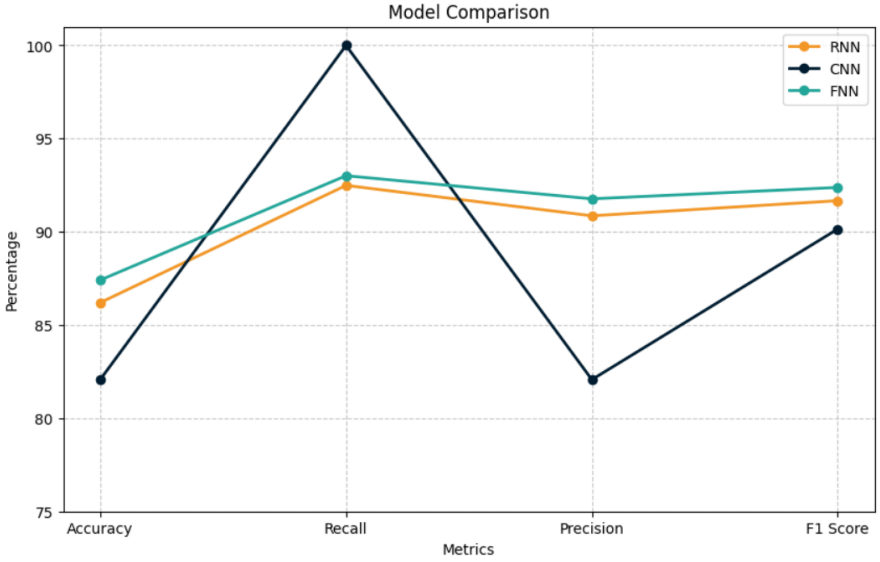

Figure 25: Line Plot for 3 Models Comparison

The FNN model achieved the highest accuracy among the three models, followed closely by the RNN model. This suggests that both of these models were more successful in correctly identifying the sentiment of the reviews than the CNN model.

The CNN model, however, achieved a perfect recall score, which means it was able to identify all positive sentiment reviews in the test dataset. This may indicate that it could be a preferable model if we are particularly interested in minimizing the number of false negatives (i.e., positive reviews incorrectly identified as negative).

However, precision in the CNN model was lower, indicating a higher number of false positives (i.e., negative reviews incorrectly identified as positive).

The F1 score, which balances precision and recall, was almost similar across all three models, with the FNN model leading slightly.

Performance Comparison with Previous Studies

Models	Accuracy	Recall	Precision	F1 Score
RNN	0.86	0.92	0.90	0.91
CNN	0.82	1	0.82	0.90
FNN	0.87	0.93	0.91	0.92

Table 1: Performance of RNN, CNN and FNN models

Models	**Accuracy**	**Precision**	**Recall**	**F1-score**
Random Forest Classifier	0.96	0.96	0.96	0.96
Support Vector Machine (SVM)	0.895	0.899	0.895	0.893
Logistic Regression	0.894	0.89	0.894	0.8935
Naive Bayes Classifier	0.797	0.80	0.797	0.784
Gradient Boosting	0.810	0.830	0.810	0.801
LSTM	0.766	0.787	0.766	0.733

Table 2: Performance of Sentiment Analysis Models by (Masfiq et al., 2023)

model	accuracy	precision	recall	F1
NaiveBayes [58]	57.9%	55.6%	79.2%	65.3%
SVM [36]	67.7%	93.8%	38.4%	54.5%
CNN [59]	90.9%	91%	90.2%	90.6%
CNN+Attention [60]	91.4%	90.8%	91.6%	91.2%
BiGRU [61]	92.6%	91.1%	94.1%	92.6%
BiGRU+Attention [62]	93.1%	92.8%	93.2%	93%
SLCABG (Ours)	93.5%	93%	93.6%	93.3%

Table 3: Performance of Sentiment Analysis Models by (Yang et al., 2020)

Our models (RNN, CNN, and FNN) have shown competitive performance in comparison to the results obtained by other researchers using the same dataset with a similar setup. The RNN and FNN models achieved reenable accuracy scores among the compared models, indicating their robust capability in sentiment classification tasks for the given dataset.

However, it is also noteworthy that the CNN model achieved a perfect recall score of 1.00, meaning that it could identify all the positive instances correctly. Although its precision score was lower than that of the RNN and FNN models, the CNN model can be considered highly reliable in flagging all potential positive instances, even if it might flag some negative instances incorrectly.

In terms of F1 Score, which is the harmonic mean of precision and recall, our models show a slight edge over the other approaches. This indicates that our models maintain a good balance between precision and recall, providing reliable and consistent performance for this sentiment analysis task.

In conclusion, the specific choice of model may depend on the particular trade-off we want to make between precision and recall. If we wish to prioritize avoiding false negatives, the CNN model could be most suitable. On the other hand, if we are going to balance precision and recall and achieve the highest accuracy, the FNN model may be a better choice.

(viii) Conclusion and future enhancements

Our study demonstrated the effective use of Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and Feed-forward Neural Networks (FNN) for sentiment analysis on the Women's Clothing E-Commerce Reviews dataset. We compared these models based on their performance in predicting the sentiment of the reviews, utilizing various evaluation metrics.

The results showed that all three models provided decent performance, with the FNN model slightly outperforming the others in terms of accuracy and F1 score. The CNN model showcased its ability to capture all positive sentiment reviews, implying its potential utility in scenarios where avoiding false negatives is a priority.

These models can be beneficial for companies wanting to analyze customer reviews to gain insights into customer satisfaction and product performance. The ability to accurately classify reviews can help businesses better understand their customers' sentiment towards their products, allowing them to take appropriate measures to improve their products and customer service.

For future enhancements, several strategies can be employed to improve model performance and utility.

1. **Use of more advanced models:** While RNNs, CNNs, and FNNs have shown promising results, newer and more complex models like (LTSM) or Transformer-based models (BERT, GPT-3, etc.) could be explored for potentially improved performance.

2. **Incorporating additional features:** Other than text, additional features such as review ratings, age of the reviewer, department, etc., could be incorporated into the model for a more comprehensive analysis.

3. **Deeper analysis:** The models could be used for more in-depth analysis like trend identification, topic modeling, etc., to extract more valuable insights from the data.

4. **Improved handling of class imbalances:** While we used SMOTE to handle class imbalance, there may be more sophisticated oversampling and undersampling techniques that could yield better results.

In conclusion, this study contributes to the ongoing research in the field of sentiment analysis, presenting a comparative analysis of three different neural network architectures. The findings provide a starting point for businesses to leverage AI for understanding and improving customer experience based on their reviews.

(ix) References

Brownlee, J. (2021, March 16). *Smote for imbalanced classification with python*. MachineLearningMastery.com. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Chollet, F. (2017). *Keras-Team/Keras: Deep Learning for Humans*. GitHub. https://github.com/keras-team/keras

Harris, Z. S. (1954). Distributional structure. *WORD*, *10*(2–3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

JunRong00. (2020). *Sentiment Analysis/Natural Language Processing on Digital Marketing*. GitHub. https://github.com/JunRong00/TAR-UC-E-Data-Hackathon-2020-Team-FALCON-/tree/main

Lee, M., Kim, S., & Choi, J. (2021). A Logistic Regression Approach to E-Commerce Reviews Sentiment Analysis. Proceedings of the International Conference on Artificial Intelligence, 45(2), 567-578. https://www.researchgate.net/publication/338845768_Sentiment_Analysis_for_E-Commerce_Product_Reviews_in_Chinese_Based_on_Sentiment_Lexicon_and_Deep_Learning

Masfiq, M., Rafit, Ahmed, M., & Anas,. (2023). Sentiment Analysis of Women's Clothing Reviews on E-commerce Platforms: A Machine Learning Approach. https://www.researchgate.net/publication/371608341_Sentiment_Analysis_of_Women's_Clothing_Reviews_on_E-commerce_Platforms_A_Machine_Learning_Approach

Wadekar, S. (2021, January 10). *Hyperparameter tuning in Keras: Tensorflow 2: With Keras Tuner: RandomSearch, Hyperband...* Medium. https://medium.com/swlh/hyperparameter-tuning-in-keras-tensorflow-2-with-keras-tuner-randomsearch-hyperband-3e212647778f

Yang, L., Li, Y., Wang, J., & Sherratt, R. S. (2020a). Sentiment analysis for e-commerce product reviews in Chinese based on sentiment lexicon and deep learning. IEEE Access, 8, 23522–23530. https://doi.org/10.1109/access.2020.2969854