DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING (ELEC0141) 23 REPORT

SN: 22082567

ABSTRACT

Sentiment analysis, a vital application within natural language processing (NLP), has attracted widespread attention due to the increasing volume of text data generated by social media platforms (such as Twitter). This project aims to categorize English Twitter messages into six basic emotions: 'sadness', 'joy', 'love', 'anger', 'fear' and 'surprise'. Four models were used as sub-projects for supervised learning classification with labelled data. Sub-project A used a traditional machine learning model, Decision Tree and achieved an accuracy of 84.20%. Whereas sub-projects B, C, and D employed more complex artificial neural network (ANN) models built based on TensorFlow, including Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BILSTM). The test accuracies of the three models are increasing in order, with accuracies of 87.60%, 88.70%, and 91.25%, respectively. The results showed that the ANN models had higher accuracies than traditional machine learning models. The experiments further demonstrated that BILSTM benefited from the reverse LSTM layer, which can better capture the long-term dependencies in the sequence, and achieved the highest accuracy in sentiment classification. The link to this project can be found on GitHub. [1]

***Index Terms*—** NLP, Sentiment Analysis, Decision Tree, MLP, LSTM, BILSTM

1. INTRODUCTION

With the significant increase in the frequency of social media usage in recent years, a corresponding surge in data (such as tweets) is generated. Tweets refer to text-based messages posted by users on Twitter. Analyzing the sentiment of tweets based on user emotions has become an essential task for various organizations and companies to improve service quality [1]. The primary purpose of sentiment analysis is to detect the subjectivity of a text, usually with a three-category classification, to determine whether the text expresses a positive, negative, or neutral sentiment [2]. However, in this project, tweets are classified into six emotions rather than the traditional three classifications to achieve a more diverse and accurate classification. The more fine-grained classification enables enterprises to achieve greater precision in market research and public opinion analysis while also providing a means to more rigorously evaluate the model's performance [3].

This study employed multiple machine learning algorithms for sentiment analysis with a six-class classification task, namely Decision Tree, Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BILSTM). The purpose was to explore the accuracy of these models in detecting emotions of 'sadness', 'joy', 'love', 'anger', 'fear' and 'surprise'. The dataset was the 'Emotion Dataset for Emotion Recognition Tasks' obtained from Kaggle [4], which contained 20,000 tweets exhibiting six different emotions. A training set of 16,000 tweets was utilized for model training, while a separate set of 2,000 tweets was used for validation and parameter tuning. Additionally, a set of 2,000 tweets was employed to identify the most effective model for the given task.

The structure of the paper is presented as follows. Section 2 offers a literature review of conventional and state-of-the-art NLP algorithms relevant to the project. In Section 3, the algorithms and principles of four models employed in the sub-projects are presented. Section 4 details the model development process in Python, including data cleaning and parameter tuning. Section 5 exhibits the experimental results with accompanying analysis. Lastly, Section 6 summarizes the project findings and describes potential research directions for future improvement.

2. LITERATURE SURVEY

MLP, as a representative neural network model, was first proposed by Gardner et al. in 1998 and was proven to be a powerful tool for handling prediction, function approximation, and classification tasks [5]. The LSTM model was initially developed to tackle tasks involving time series prediction, music composition, and voice processing. It effectively addresses challenging artificial problems characterized by long-term dependencies previously unsolvable by earlier recurrent network algorithms [6].

In 2016, Muhammad et al. conducted a study on sentiment mining of Roman-Urdu using Naïve Bayesian, KNN, and decision trees for binary classification. The research revealed that Naïve Bayesian benefited from the zero-one loss function and achieved a testing accuracy of 97.50%, higher than that of decision trees (92.50%) and KNN (92.50%). Additionally, due to the absence of a calculation process in

[1] https://github.com/Ys1ong/DLNLP_23_SN22082567

decision trees, it is the fastest training model [7]. Moreover, Yun et al. used several traditional machine learning models, including Naïve Bayesian, SVM, Bayesian networks, C4.5 decision tree, and random forest algorithm, to classify Twitter data for airline services analysis into positive, negative, neutral, and irrelevant categories. In contrast, in the more complex four-class classification task, decision trees had higher accuracy (83.70%) than Naïve Bayesian (82.40%), SVM (77.80%), and Bayesian networks (82.20%). Yun et al. also concluded that the ensemble method proposed was superior to the individual classifiers with accuracy 84.20% [8].

Betül et al. experimented with various artificial neural network models for classifying movie reviews, including MLP, CNN, LSTM, and BILSTM. The results indicated that the accuracy of BILSTM was higher than that of LSTM and MLP, and the ensemble model of CNN-LSTM had the highest performance with an accuracy of 98.07% [9]. Similarly, Beakcheol et al. presented the architecture of CNN (Convolutional Neural Network), which includes convolutional layers and pooling layers employed to extract more advanced features, while LSTM models can capture long-term dependencies among word sequences, making them more appropriate for text classification applications. In addition, the BiLSTM+CNN attention hybrid model proposed by Beakcheol et al. outperformed existing ANN models regarding accuracy on the IMDB (Internet Movie Database) movie review dataset [10].

For more complex multi-class sentiment analysis, Muhammad utilized the same dataset as this project and achieved an accuracy of 97.50% on 2,000 test data by adding word weighting TF-IDF (Term Frequency–Inverse Document Frequency) to the LSTM model, while the accuracy using LinearSVC was 89.00% [11]. Similar findings were also reported by Abdulqahar et al., who attempted various methods of word embeddings, including Word2Vec and GLoVE (Global Vectors for word representation). Abdulqahar et al. found that using GLoVE with CNN, LSTM, and BILSTM on the same dataset achieved accuracies of 55.00%, 95.00%, and 86.00%, respectively [12].

Compared to traditional machine learning models like SVM and decision trees, ANN models have demonstrated a higher degree of effectiveness in sentiment analysis tasks. However, ANN models often demand considerable human intervention, especially in data labelling and pre-processing tasks. These are crucial in identifying the most compelling features for training complex models that require longer processing times [13]. Currently, supervised learning is the most commonly used technique for classification in this field due to its simplicity and high accuracy. However, in the vast majority of datasets consisting of the English language, unsupervised learning and other natural language sentiment analysis remain a challenging open problem [14].

3. DESCRIPTION OF MODELS

3.1. Task A: Decision Tree Classification

The decision tree is a commonly used traditional machine learning classification algorithm, which classifies data by constructing a tree structure. The tree structure is shown in Figure 1.

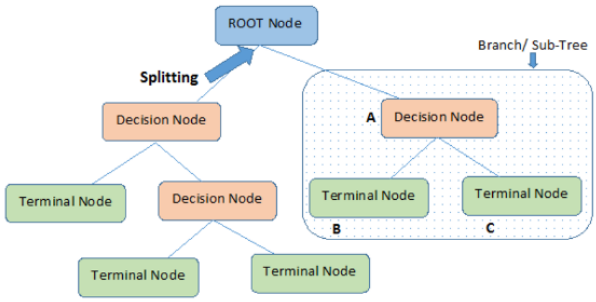


Fig. 1: Tree Structure [15]

A decision tree typically consists of four components: root node, decision node, leaf node, and branch:

1. The root node serves as the starting point of the decision tree and represents the entire input dataset.

2. The decision node follows the root node and represents the judgment on a specific feature.

3. The leaf node, also known as the terminal node, represents the final classification result of the data.

4. The branch is the link between nodes and represents the range of different feature values.

Pruning is a technology used to prevent overfitting. The decision tree's complexity is reduced by deleting some unnecessary decision nodes or leaf nodes. The pruning technique is divided into two types: pre-pruning and post-pruning. Pre-pruning occurs when the decision tree is built and judged before the node split. The split will be stopped if the split does not improve the performance. Post-pruning occurs after building a decision tree; if the performance is not affected on the validation dataset after pruning, the node is set to a leaf node. Pruning can improve the generalization ability of decision trees, reduce overfitting, and improve the accuracy of the decision tree on test data [16].

Overfitting can be effectively prevented by limiting the minimum number of splits in a decision tree. Furthermore, as one of the traditional machine learning algorithms, decision trees have been chosen as one of the first methods for sentiment classification due to their higher accuracy compared to SVM and Naïve Bayesian [8].

3.2. MLP Classification

An MLP is a type of ANN that takes input data and processes it through multiple layers of neurons to generate a set of output values. The usual topology of an MLP consists of interconnected nodes organized into multiple layers, including input, hidden, and output layers, as shown in Figure 2. Each neuron in the network receives input signals from neurons in the preceding layer, performing a weighted sum of these inputs. Then, a nonlinear activation function is applied to distinguish data that cannot be linearly separated [17].

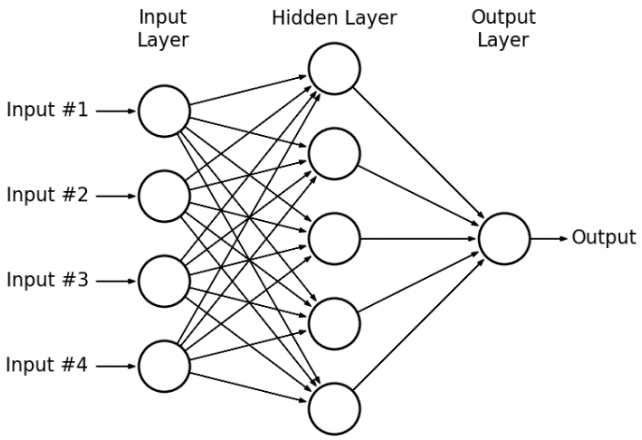


Fig. 2: MLP Structure [18]

The output activation α^{l+1} at layer $l+1$ is obtained by:

$$\alpha^{l+1} = \sigma(w^l \alpha^l + b^l) \quad [17] \tag{1}$$

where w^l and b^l are the weight and bias at previous layer l. σ refers to a nonlinear activation function, which can be selected from diverse options such as 'sigmoid', 'tanh', or 'ReLU'.

For an MLP with m layers, the first input layer is $a^1 = x$, while the final output layer is $h_{w,b}(x) = a^m$. The discrepancy between the predicted outputs of the model and the valid label is minimized:

$$J(W, b; x, y) = \frac{1}{2}||h_{w,b}(x - y)||^2 \quad [17] \tag{2}$$

As a non-linear ANN model, MLP has a significant advantage in handling non-linear relationships in data, achieved through non-linear activation functions and interconnected layers of neurons. Furthermore, MLP has high scalability, as the complexity of the model can be adjusted by increasing or decreasing the number of hidden layers or neurons to better adapt to different datasets. Therefore, MLP was selected as the first ANN model to be trained after traditional machine learning decision tree algorithms.

3.3. LSTM Classification

It can be observed from Figure 3, LSTM unit consists of one cell state and three gates: an input gate, a forget gate, and an output gate. At each time step, the input data is updated by the input gate. The forget gate controls the update of the cell state from the previous time step, and the output gate updates the output data. The yellow circle is the 'sigmoid' activation function, whereas the pink circle expresses the 'tanh' activation function.

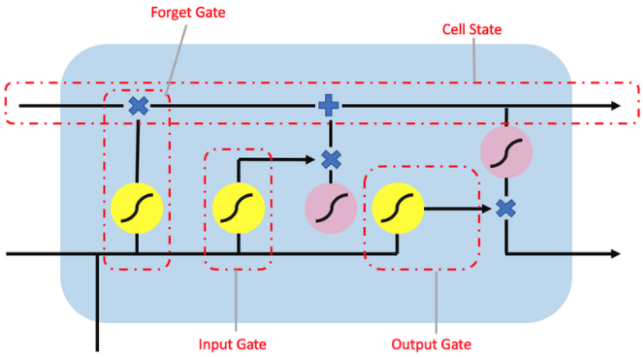


Fig. 3: LSTM Unit [19]

The update of LSTM model is:

$$\begin{aligned} f_t &= \sigma_g(W_f \cdot x_t + U_i \cdot h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i \cdot x_t + U_f \cdot h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \\ \tilde{C}_t &= \tanh(W_C \cdot x_t + U_c \cdot h_{t-1} + b_C) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ h_t &= o_t \odot \tanh(C_t) \end{aligned} \tag{3}$$

where σ_g is the 'sigmoid' activation function, $\odot$ represents element-wise multiplication, i_t, f_t, o_t and c_t represents input gate, forget gate, output gate and activation vector, W_f, W_i, W_C and W_o are weight matrices, b_f, b_i, b_C and b_o are bias vectors and U_f, U_i, U_C and U_o are trainable parameters. h_t is the implied state output at the time step [20].

Using gating mechanisms to control information flow and retention over time, LSTM can better capture long-term dependencies in text sequences, improving classification accuracy. Therefore, LSTM was chosen for sentiment classification after MLP.

3.4. BILSTM Classification

BILSTM, as an extension of LSTM, employs both forward and backward processing units in a bidirectional RNN model. Compared to LSTM uses a single LSTM unit to propagate information in a forward direction at each time step, BILSTM utilizes another LSTM unit to propagate information in

a backward direction. As a result, BILSTM is capable of capturing contextual information in the sequence by leveraging both past and future information. The outputs of the forward and backward LSTMs are concatenated together to form the final output of the current time step. The structure of BILSTM is shown in Figure 4.

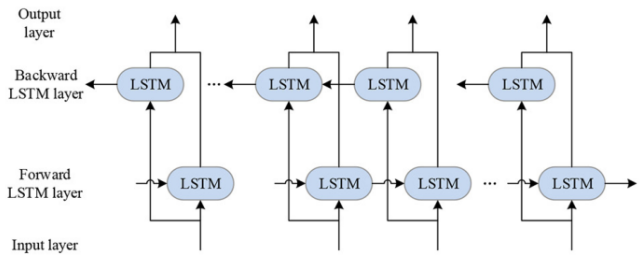


Fig. 4: BILSTM Structure [21]

The forward propagation is the same as LSTM model in Equation 3. The backward propagation is the inverse of Equation 3:

$$
\begin{aligned}
\overleftarrow{f_t} &= \sigma_g(\overleftarrow{W_f} \cdot x_t + \overleftarrow{U_i} \cdot \overleftarrow{h_{t-1}} + \overleftarrow{b_f}) \\
\overleftarrow{i_t} &= \sigma_g(\overleftarrow{W_i} \cdot x_t + \overleftarrow{U_f} \cdot \overleftarrow{h_{t-1}} + \overleftarrow{b_i}) \\
\overleftarrow{o_t} &= \sigma_g(\overleftarrow{W_o} \cdot x_t + \overleftarrow{U_o} \cdot \overleftarrow{h_{t-1}} + \overleftarrow{b_o}) \\
\overleftarrow{\tilde{C}_t} &= \tanh(\overleftarrow{W_C} \cdot x_t + \overleftarrow{U_c} \cdot \overleftarrow{h_{t-1}} + \overleftarrow{b_C}) \\
\overleftarrow{C_t} &= \overleftarrow{f_t} \odot \overleftarrow{C_{t-1}} + \overleftarrow{i_t} \odot \overleftarrow{\tilde{C}_t} \\
\overleftarrow{h_t} &= \overleftarrow{o_t} \odot \tanh(\overleftarrow{C_t})
\end{aligned}
\tag{4}
$$

In task 3, LSTM has been able to capture long-term dependencies in sequences. However, through forward propagation, LSTM only processes input sequences in a forward direction, from past to future. Therefore, in task 4, BILSTM is introduced to connect the outputs of forward and backward LSTMs, capturing both past-to-future and future-to-past sequences and addressing the shortcomings of LSTM in information capture. Moreover, the differences in algorithms and performance can be further compared by implementing the two models.

4. IMPLEMENTATION

4.1. Dataset

The dataset used in this project is the 'Emotion Dataset for Emotion Recognition Tasks', which was obtained from Kaggle [4]. The dataset comprises three 'csv' files ('training.csv', 'validation.csv', and 'test.csv'), containing 16,000, 20,000, and 2,000 tweets, respectively, that were employed for training, validation, and test. The proportion of training, validation, and testing data is 8:1:1.

Each 'csv' file contains tweets of varying lengths and six basic emotions are labelled using integers from '0' to '5', where '0' represents 'sadness', '1' represents 'joy', '2' represents 'love', '3' represents 'anger', '4' represents 'fear', and '5' represents 'surprise'. The distribution details of the data labels are presented in Figure 14, 15 and Table 1.

	Training.csv	Vlidation.csv	Test.csv
joy	5,362 (33.5%)	704 (35.2%)	695 (34.8%)
sadness	4,666 (29.2%)	550 (27.5%)	581 (29.1%)
anger	2,159 (13.5%)	275 (13.8%)	275 (13.8%)
fear	1,937 (12.1%)	212 (10.6%)	224 (11.2%)
love	1,304 (8.2%)	178 (8.9%)	159 (8.0%)
surprise	572 (3.6%)	81 (4.1%)	66 (3.3%)
total	16,000	2,000	2,000
max_len	66	61	61

Table 1: Distribution of Training, Validation and Test Data

It can be observed from the dataset that the distribution of the labels is unbalanced, where the majority of tweets belong to the categories of 'joy' and 'sadness', while a tiny portion of tweets belongs to the 'surprise'. However, the label distributions are similar across all three datasets, with the proportions of emotions being in the order of 'joy' > 'sadness' > 'anger' > 'fear' > 'love' > 'surprise'.

4.2. Pipline

The pipelines for the four tasks are as follows. The traditional machine learning model, decision tree, use frequency-inverse document frequency (TF-IDF), while the ANN models use word-to-index mapping (word2id).

1. Data Pre-processing: Clean each tweet in the datasets by removing URLs and punctuation and converting all text to lowercase.

2. Embedding: Create a word embedding using TF-IDF or word2id and convert each document to a fixed-length vector representation

   (a) For TF-IDF: Compute the sparse matrix for each dataset in the corpus.

(b) For word2id: Get corpus as all unique words across training and validation datasets, map each word in the corpus to a unique integer index, and replace each word in the document with its corresponding integer index and pad or truncate to a fixed length.

3. Training: Feed the sparse matrix or vector representations and labels to a machine learning model to train a machine learning model.

4. Predicting: Use the trained model to predict the sentiment of a new text by first converting the text to a vector representation using the same embedding technique used during training and then feeding the vector to the trained model.

5. Evaluating: Evaluate the performance of the predicted results with true labels using the accuracy and confusion matrix.

4.2.1. Data Pre-processing

Data pre-processing aims to standardise and clean the text data so that downstream models can analyse it more efficiently and accurately. The cleaning process is shown as follows.

1. Remove URLs using regular URL expression `https?:\/\/\S+`.

2. Remove punctuation from `string.punctuation` using the `translate()`.

3. Convert all text to lowercase using the `lower()`.

4.2.2. Embedding TF-IDF

The TF-IDF embedding uses the `TfidfVectorizer` imported from `sklearn.feature_extraction.text` to transform the raw text data into numerical feature vectors. The `fit_transform()` is called on the combined training and validation datasets to learn the vocabulary dictionary and train the vectorizer. Then, the `transform()` is applied separately on the training and validation datasets to transform the raw text into feature vectors based on the learned vocabulary from the combined dataset and return the sparse document-term matrices.

4.2.3. Embedding word2id

The word2id embedding employs `TweetTokenizer` from the `nltk.tokenize` library to tokenize each tweet into a list of words. Then, the `get_vocab()` function creates a word-to-id mapping for all unique words across the training and validation datasets. The `get_vocab()` function stores the maximum tweet length and iterates through each pre-processed tweet, adding each unique word to a dictionary and mapping the word to a unique integer index. Three special tokens ('0' for '`<PAD>`', '1' for '`<UNK>`', and '3' for '`<EOS>`') are added to the dictionary with fixed indices at the beginning. '`<PAD>`' is used to pad sequences to a fixed length, '`<UNK>`' represents unknown words that are not present in the learned vocabulary, and '`<EOS>`' is the end of the sentence. The resulting dictionary word-to-id mapping and maximum tweet length are returned.

Then, the `tokenization()` function converts each tweet into a fixed-length sequence of integers using word-to-id mapping. The `tokenization()` function numerically tokenizes each word in the tweet using the word-to-id mapping returned from `get_vocab()` function. If a word is not found in the mapping, it is replaced with the '`<UNK>`' token. The resulting sequence is then padded with '`<PAD>`' tokens if it is shorter than the maximum tweet length or truncated if it is longer.

4.3. Task A: Decision Tree Classification

The decision Tree, as a representative of traditional machine learning, was chosen as the first method for sentiment classification. The decision Tree was constructed using the `tree` module imported from `sklearn`.

The Classifier of a single decision tree can be tuned by adjusting the parameter of the minimum number of samples required to split a node. In this task, the minimum number of samples required to split a node was tested at intervals of 5 from 55 to 300. The training data was trained using these 50 different parameters, and the accuracy was evaluated using the validation dataset. The resulting accuracy is shown in Figure 5.

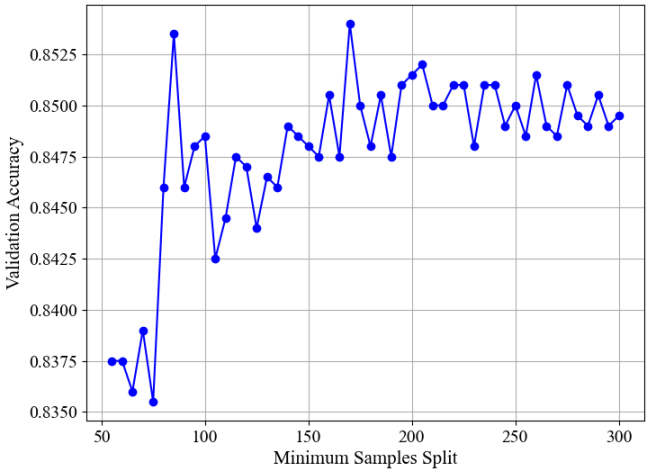


Fig. 5: Validation Accuracy of Different Minimum Samples Split of Decision Tree

Figure 5 demonstrates that validation accuracy has the highest value when the sample split is 170. The learning

curve for a single decision tree with a sample split of 170 was calculated using `learning_curve` imported from `sklearn.model_selection` and plotted in Figure 6. The training accuracy of the decision tree with 170 samples split is 89.16%, and the validation accuracy is 86.06%.

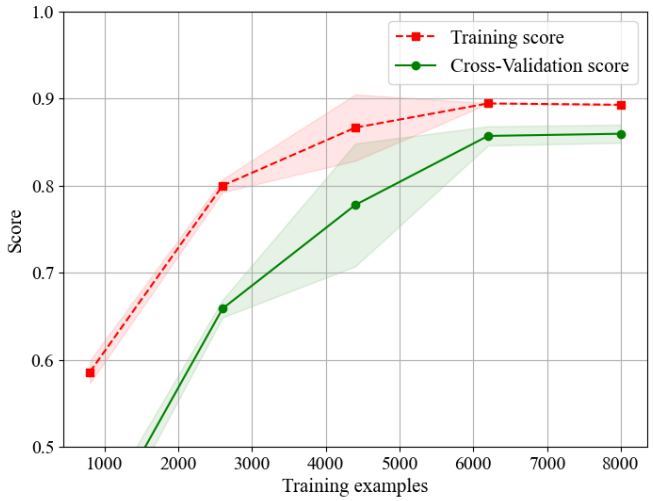


Fig. 6: Learn Curve of Decision Tree

4.4. Task B: MLP Classification

The MLP model was constructed using the `layers` and `models` imported from the `tensorflow.keras` library.

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 66, 100)	1618900
global_average_pooling1d (GlobalAveragePooling1D)	(None, 100)	0
dense (Dense)	(None, 128)	12928
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 6)	774

Total params: 1,632,602

Trainable params: 1,632,602

Non-trainable params: 0

Table 2: Structure of MLP Model

Table 2 displays the structure of the MLP, including five layers. The embedding layer is used to embed words in a text sequence into a high-dimensional vector space containing three adjustable parameters, where `input_dim` is the size of the corpus, `output_dim` represents the dimension of the embedding vector, and `input_length` is the length of the input sequence. The pooling layer reduces the dimensionality of the two-dimensional tensor output from the embedding layer. The first fully connected layer uses 'ReLU' as the activation function. The dropout layer is set with a dropout rate of 0.2, randomly setting some neurons to 0 with a probability of 0.2. The last fully connected layer serves as the output layer, using the activation function 'softmax' to map the output to a vector of size 6.

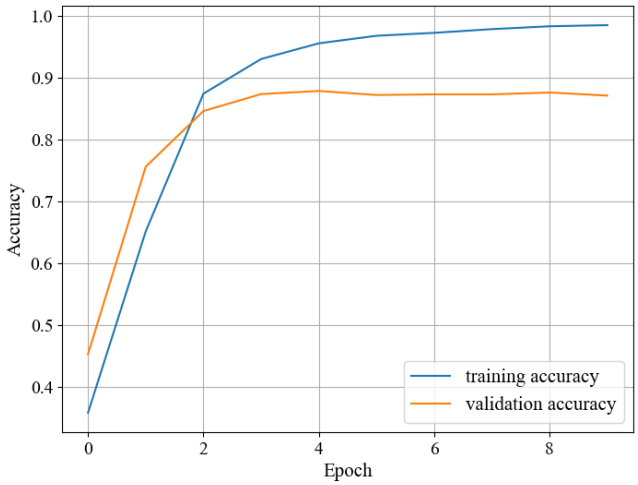


Fig. 7: Learn Curve of MLP

The learning rate was set to 0.001 using `keras.optimizers.Adam(0.001)`, and the default batch size of 32 was used. The model was trained for a maximum of 50 epochs, but training was stopped early using callbacks `EarlyStopping` when no increase in validation loss was observed in 5 consecutive epochs. The learning curve of the MLP was plotted by the training and validation accuracy histories using the `matplotlib.pyplot` as shown in Figure 7. The training and validation accuracy of MLP increased significantly in the first three epochs and remained stable after the fourth epoch. The training accuracy was 0.9844, and the validation accuracy was 0.8705, stopped at the 10th epoch with 15.72s training time.

4.5. Task C: LSTM Classification

Similar to Task B, the LSTM model was built using the `layers` and `models` imported from the `tensorflow.keras` library.

The LSTM model consists of five layers, including an embedding input layer (same as MLP), two LSTM layers, one dropout layer (with a dropout rate of 0.2), and one fully connected layer (with a 'softmax' activation function). The first LSTM layer contains 64 LSTM units, while the second LSTM layer contains 32 LSTM units. The structure of the LSTM model is shown in Table 3.

Model: “sequential”

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 66, 100)	1618900
lstm (LSTM)	(None, 66, 64)	42240
dropout (Dropout)	(None, 66, 64)	0
lstm_1 (LSTM)	(None, 32)	12416
dense (Dense)	(None, 6)	198

Total params: 1,673,754

Trainable params: 1,673,754

Non-trainable params: 0

Table 3: Structure of LSTM Model

To ensure consistency of the control experiment, the learning rate and batch size were set to 0.001 and 32, respectively, the same as used for MLP, and the `patience` parameter of `EarlyStopping` was set to 5.

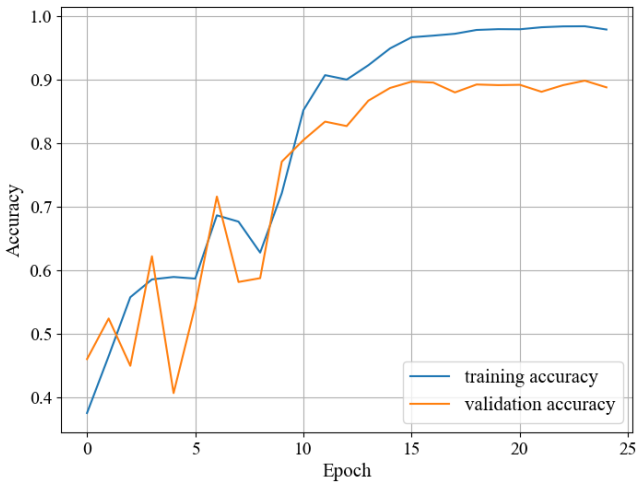


Fig. 8: Learn Curve of LSTM

The learning curve of LSTM is shown in Figure 8. The training and validation accuracies fluctuated, increasing in the first ten epochs and remaining stable after 15. LSTM took the longest training time (117.54s) among all models and stopped after 25 epochs with a training accuracy of 97.96% and a validation accuracy of 88.85%.

4.6. Task D: BILSTM Classification

Similar to Task B and C, with the `tensorFlow` library, the architecture of the BILSTM model is shown in Table 4.

Model: “sequential”

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 66, 100)	1618900
dense (Dense)	(None, 66, 128)	12928
bidirectional (Bidirectional)	(None, 66, 128)	98816
dropout (Dropout)	(None, 66, 128)	0
bidirectional_1 (Bidirectional)	(None, 64)	41216
dense_1 (Dense)	(None, 6)	390

Total params: 1,772,250

Trainable params: 1,772,250

Non-trainable params: 0

Table 4: Structure of BILSTM Model

The BILSTM model consists of 6 layers, with the input layer, output layer, and dropout layer identical to layers in the MLP and LSTM models. An additional fully connected layer is added after the input layer, containing 128 neurons and utilizing L2 regularization. The first BILSTM layer, with 64 LSTM units, is used to process sequential data, while the second BILSTM layer contains 32 LSTM units.

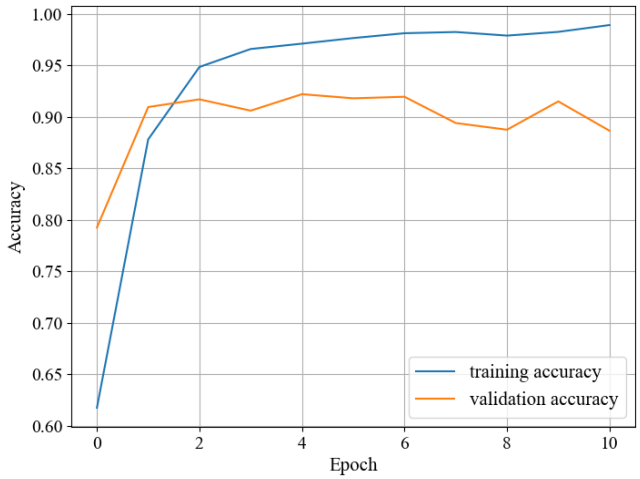


Fig. 9: Learn Curve of BILSTM

The learning curve of the BILSTM model shown in Figure 9 demonstrates faster learning efficiency than LSTM, as the validation accuracy exceeded 90% in the second epoch and remained stable. The training was stopped by early stopping after 11 epochs, achieving a training accuracy of 98.91% and a validation accuracy of 88.65%. The training time was less than that of LSTM, taking 88.38s.

5. EXPERIMENTAL RESULTS AND ANALYSIS

5.1. Accuracy Evaluation

Task	Model	Train Acc	Val Acc	Test Acc
A	Decision Tree	0.8930	0.8597	0.8420
B	MLP	0.9844	0.8705	0.8760
C	LSTM	0.9796	0.8885	0.8870
D	BILSTM	0.9891	0.8865	0.9125

Table 5: Training, Validation and Test Accuracy for Task A, B, C and D

The four models' training, validation and test accuracies can be observed in Table 5. The accuracy was calculated using the `accuracy_score` function imported from `sklearn.metrics`. As expected, the test accuracy gradually increased with the increasing complexity of the models, with the performance ranking as BILSTM $>$ LSTM $>$ MLP $>$ Decision Tree.

However, overall, the test accuracy did not increase significantly, with the highest-performing BILSTM (91.25%) only showing a 7.05% improvement over the lowest-performing Decision Tree (84.20%). Several potential reasons could cause the resulting phenomenon:

- Tweets in the dataset are too short. Even the longest tweet after data cleaning only contained 66 words, and the vast majority of tweets in the dataset were only one sentence in length. Short tweets may not provide sufficient information for the models, which explains why LSTM did not perform much better than MLP (only a 1.1% improvement).

- The limited size of the corpus. The vocabulary size of the training and validation sets is only 16,189 words. Therefore when mapping unknown words to the UNK (Unknown Word) index in the vocabulary during testing on an unseen dataset, an excessive number of unknown words causes complex models not to be significantly better than simpler ones on the test set.

- The label distribution is imbalanced. According to the data distribution presented in Section 4, most of the training data labels are 0 ('sadness') or 1 ('joy'), which resulting the models being more accurate in predicting corresponding two emotions and less sensitive to other emotions, resulting in more minor differences between four models.

Additionally, with the aid of the 'time' library, the training time for the four models was 5.94s for the decision tree, 15.72s for the MLP, 117.54s for the LSTM, and 88.38s for the BILSTM.

5.2. Confusion Matrix

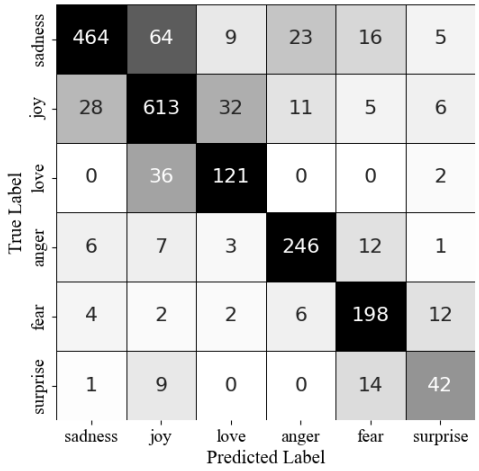


Fig. 10: Confusion Matrix of Decision Tree

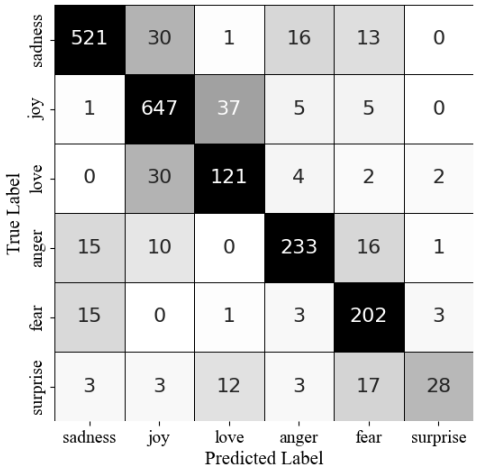


Fig. 11: Confusion Matrix of MLP

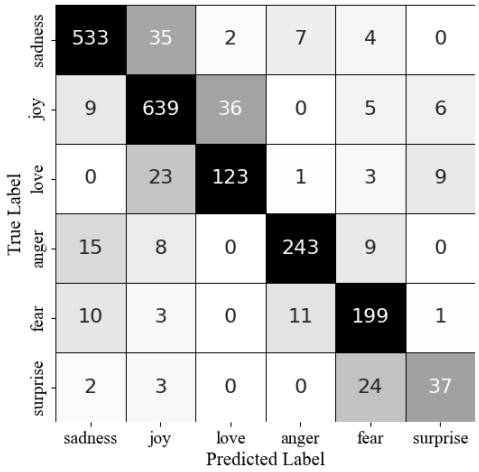


Fig. 12: Confusion Matrix of LSTM

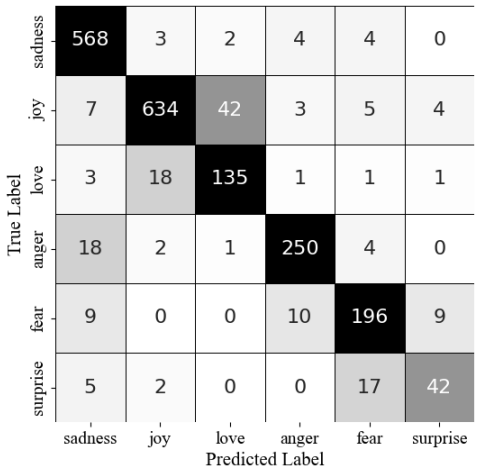


Fig. 13: Confusion Matrix of BILSTM

Figure 10, 11, 12, and 13 depict the confusion matrices of the four models. Compared to ANN models, the traditional Decision Tree misclassified the most tweets with the emotion 'sadness', where 64 tweets were predicted as the opposite 'joy' emotion. In contrast, other ANN models improved predicting the 'sadness' emotion. On the other hand, the ANN models performed poorly in predicting the 'joy' emotion and even had higher error rates than the decision tree. Moreover, all three ANN models tended to misclassify the 'joy' emotion as the 'love' emotion.

6. CONCLUSION

Based on the given dataset, the ANN models demonstrate higher accuracy in sentiment analysis than traditional machine learning models. Additionally, with the increasing complexity of the model, the accuracy improves slightly. The BILSTM model, the most complex ANN model, outperforms the other three models with the highest accuracy. However, due to the short length and the limited number of tweets in the training data, ANN models could not effectively capture the contextual relationships in the data, leading to only a marginal improvement compared to traditional machine learning models. On the other hand, for NLP problems, the size and quality of data are crucial factors that directly affect model performance.

Traditional machine learning algorithms suit small data sizes and corpus in real-world word applications. More extensive and diverse data should be provided for more complex ANN models to enhance the generalization capability. One potential direction for future development is to explore ensemble learning by integrating both ANN and traditional machine learning models to achieve superior performance.

7. REFERENCES

[1] Furqan Rustam, Imran Ashraf, Arif Mehmood, Saleem Ullah, and Gyu Sang Choi, "Tweets classification on the base of sentiments for us airline companies," *Entropy*, vol. 21, no. 11, pp. 1078, 2019.

[2] Iti Chaturvedi, Erik Cambria, Roy E Welsch, and Francisco Herrera, "Distinguishing between facts and opinions for sentiment analysis: Survey and challenges," *Information Fusion*, vol. 44, pp. 65–77, 2018.

[3] Mudasir Ahmad Wani, Nancy Agarwal, Suraiya Jabin, and Syed Zeeshan Hussain, "User emotion analysis in conflicting versus non-conflicting regions using online social networks," *Telematics and Informatics*, vol. 35, no. 8, pp. 2326–2336, 2018.

[4] "Emotion dataset for emotion recognition tasks," https://www.kaggle.com/datasets/parulpandey/emotion-dataset, Accessed: 2023-05-06.

[5] Matt W Gardner and SR Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[6] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] Muhammad Bilal, Huma Israr, Muhammad Shahid, and Amin Khan, "Sentiment classification of roman-urdu opinions using naïve bayesian, decision tree and knn classification techniques," *Journal of King Saud University-Computer and Information Sciences*, vol. 28, no. 3, pp. 330–344, 2016.

[8] Yun Wan and Qigang Gao, "An ensemble sentiment classification system of twitter data for airline services analysis," in *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, 2015, pp. 1318–1325.

[9] Betül Ay Karakuş, Muhammed Talo, İbrahim Rıza Hallaç, and Galip Aydin, "Evaluating deep learning models for sentiment classification," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 21, pp. e4783, 2018.

[10] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang-ug Kang, and Jong Wook Kim, "Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism," *Applied Sciences*, vol. 10, no. 17, pp. 5841, 2020.

[11] Muhammad Ibnu Alfarizi, Lailis Syafaah, and Merinda Lestandy, "Emotional text classification using tf-idf

(term frequency-inverse document frequency) and lstm (long short-term memory)," *JUITA: Jurnal Informatika*, vol. 10, no. 2, pp. 225–232, 2022.

[12] Abdulqahar Mukhtar Abubakar, Deepa Gupta, and Suja Palaniswamy, "Explainable emotion recognition from tweets using deep learning and word embedding models," in *2022 IEEE 19th India Council International Conference (INDICON)*. IEEE, 2022, pp. 1–6.

[13] Jurgita Kapočiūtė-Dzikienė, Robertas Damaševičius, and Marcin Woźniak, "Sentiment analysis of lithuanian texts using traditional and deep learning approaches," *Computers*, vol. 8, no. 1, pp. 4, 2019.

[14] Marouane Birjali, Mohammed Kasri, and Abderrahim Beni-Hssane, "A comprehensive survey on sentiment analysis: Approaches, challenges and trends," *Knowledge-Based Systems*, vol. 226, pp. 107134, 2021.

[15] Anshul Saini, "Decision tree algorithm – a complete guide, analytics vidhya," https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/, Accessed: 2023-05-10.

[16] John Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine learning*, vol. 4, pp. 227–243, 1989.

[17] Ce Zhang, Xin Pan, Huapeng Li, Andy Gardiner, Isabel Sargent, Jonathon Hare, and Peter M Atkinson, "A hybrid mlp-cnn classifier for very fine resolution remotely sensed image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 133–144, 2018.

[18] Hassan Mohamed, Abdelazim Negm, Mohamed Zahran, and Oliver C Saavedra, "Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: case study el burullus lake," in *International water technology conference*, 2015, pp. 12–14.

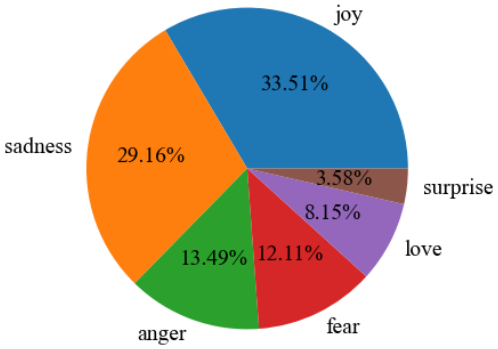
[19] Divish Rengasamy, Mina Jafari, Benjamin Rothwell, Xin Chen, and Grazziela P Figueredo, "Deep learning with dynamically weighted loss function for sensor-based prognostics and health management," *Sensors*, vol. 20, no. 3, pp. 723, 2020.

[20] Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang, "Cached long short-term memory neural networks for document-level sentiment classification," *arXiv preprint arXiv:1610.04989*, 2016.

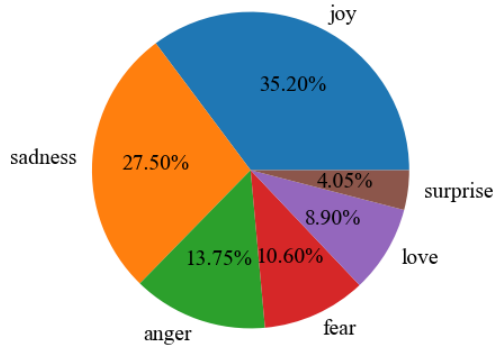
[21] Jiarui Zhong, Tangxian Chen, and Liuhan Yi, "Face expression recognition based on ngo-bilstm model," *Frontiers in Neurorobotics*, vol. 17, 2023.

Appendix

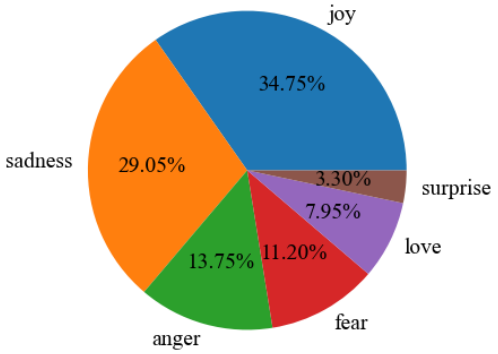
A. Data Distribution



(a) Training Data

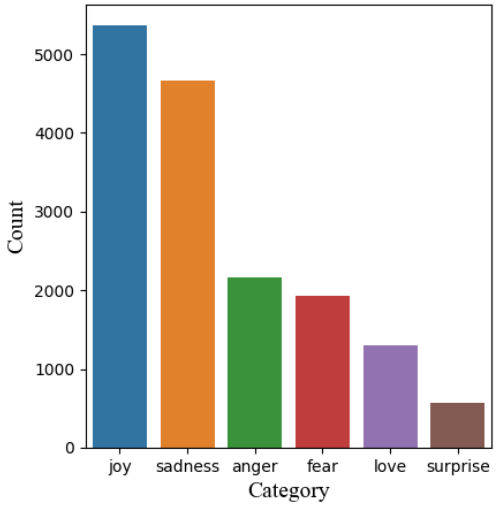


(b) Validation Data

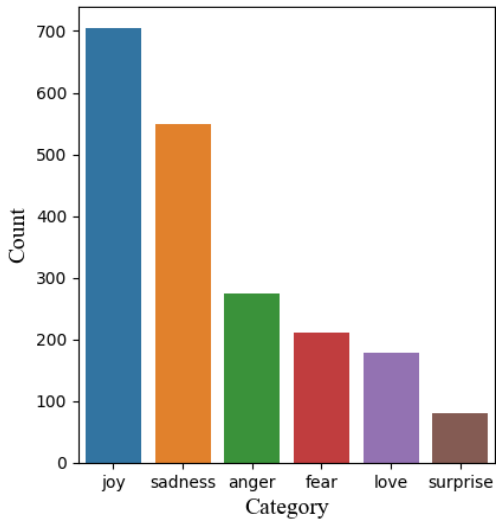


(c) Test Data

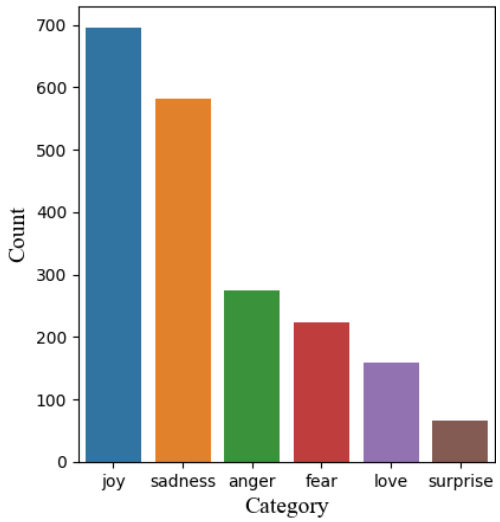
Fig. 14: Label Distribution Pie Chart



(a) Training Data



(b) Validation Data



(c) Test Data

Fig. 15: Label Distribution Bar Chart

B. Data visualization

	text	label	description
0	i didnt feel humiliated	0	sadness
1	i can go from feeling so hopeless to so damned...	0	sadness
2	im grabbing a minute to post i feel greedy wrong	3	anger
3	i am ever feeling nostalgic about the fireplac...	2	love
4	i am feeling grouchy	3	anger
...	...	...	...
15995	i just had a very brief time in the beanbag an...	0	sadness
15996	i am now turning and i feel pathetic that i am...	0	sadness
15997	i feel strong and good overall	1	joy
15998	i feel like this was such a rude comment and i...	3	anger
15999	i know a lot but i feel so stupid because i ca...	0	sadness

16000 rows × 3 columns

(a) Training Data

	text	label	description
0	im feeling quite sad and sorry for myself but ...	0	sadness
1	i feel like i am still looking at a blank canv...	0	sadness
2	i feel like a faithful servant	2	love
3	i am just feeling cranky and blue	3	anger
4	i can have for a treat or if i am feeling festive	1	joy
...	...	...	...
1995	im having ssa examination tomorrow in the morn...	0	sadness
1996	i constantly worry about their fight against n...	1	joy
1997	i feel its important to share this info for th...	1	joy
1998	i truly feel that if you are passionate enough...	1	joy
1999	i feel like i just wanna buy any cute make up ...	1	joy

2000 rows × 3 columns

(b) Validation Data

	text	label	description
0	im feeling rather rotten so im not very ambiti...	0	sadness
1	im updating my blog because i feel shitty	0	sadness
2	i never make her separate from me because i do...	0	sadness
3	i left with my bouquet of red and yellow tulip...	1	joy
4	i was feeling a little vain when i did this one	0	sadness
...	...	...	...
1995	i just keep feeling like someone is being unki...	3	anger
1996	im feeling a little cranky negative after this...	3	anger
1997	i feel that i am useful to my people and that ...	1	joy
1998	im feeling more comfortable with derby i feel ...	1	joy
1999	i feel all weird when i have to meet w people ...	4	fear

2000 rows × 3 columns

(c) Test Data

Fig. 16: Data visualization