Mini-project 1: Emoticon Classification

Varun Tandon
Roll No. 211151

Gary Derrick Anderson J
Roll No. 210383

Bhavesh Shukla
Roll No. 210266

1 TASK 1

Our first task was to train some binary classification models on each of the 3 datasets and identify the best possible model for each one. Our notion of "best" will be based on two aspects:

- Achieving high accuracy of the trained model on the validation/test set
- Using a low amount of training data to train the model.

We experimented with varying the number of training examples, using the first 20 %, 40%, 60%, 80%, and 100% of the training examples from each training dataset, and reported how accuracy varied on the validation sets. Once we had identified the best models for each of the three datasets, we used them to predict the labels for the corresponding test sets.
The 3 datasets have the same number of training examples and they represent the same raw training data but are only different in the features used to represent each input

1.1 Dataset 1: Emoticons as Features Dataset

The features of each input in this dataset are given in the form of 13 emoticon sequences.

- At first we used **Bert Embeddings** to map the emojis and develop features based on the **sum, mean, and standard deviation** of the sequences formed by the embeddings, however, we found that they had practically no correlation with the labels

Correlation matrix for df1:

	mean_embedding	sum_embedding	std_embedding	label
mean_embedding	1.000000	1.000000	-0.033465	0.012419
sum_embedding	1.000000	1.000000	-0.033465	0.012419
std_embedding	-0.033465	-0.033465	1.000000	-0.026337
label	0.012419	0.012419	-0.026337	1.000000

Figure 1: Correlation matrix for training dataset(df1) for the engineered features

- We found that the input data was the sequence of emoticons so we decided to use **the Long-short-memory (LSTM) RNN framework** for feature extraction. For this first, we tokenized the emojis and then extracted the features
- We trained the extracted features using **SVM** and **Random Forest classifier** and found that Random Forest classifier was giving better validation accuracies
- The validation accuracies for the training data fractions were as follows
  - 20% : 0.88
  - 40% : 0.90
  - 60% : 0.92

- 80% : 0.95
- 100% :0.95

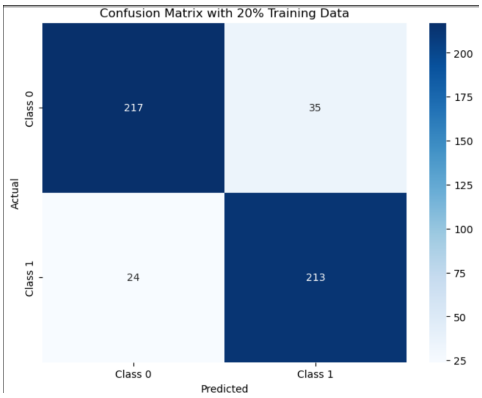

Figure 2: Confusion Matrix for 20% training data

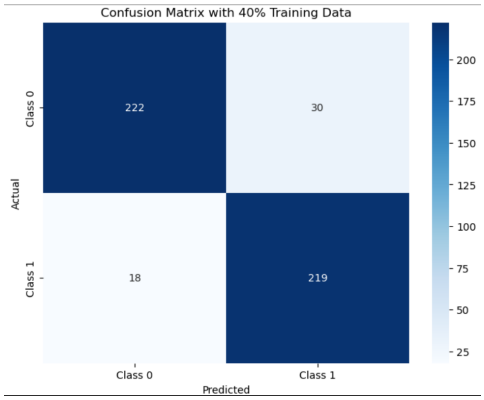

Figure 3: Confusion Matrix for 40% training data

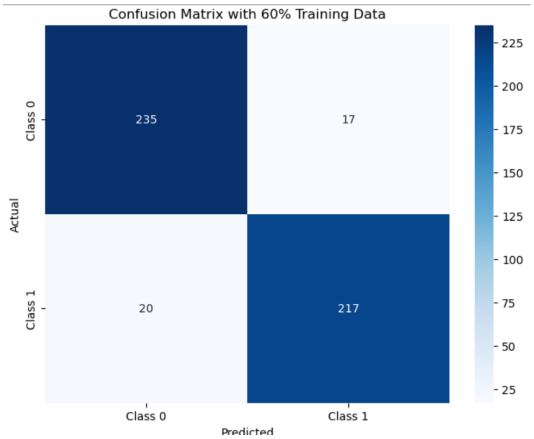

Figure 4: Confusion Matrix for 60% training data

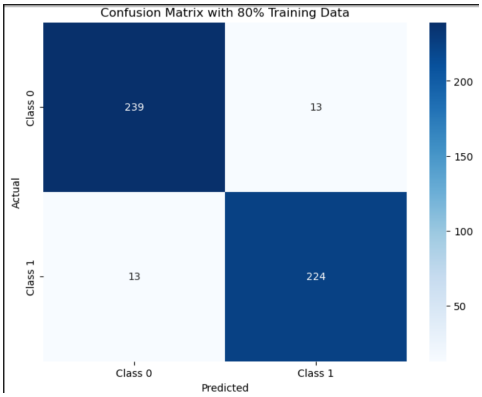

Figure 5: Confusion Matrix for 80% training data

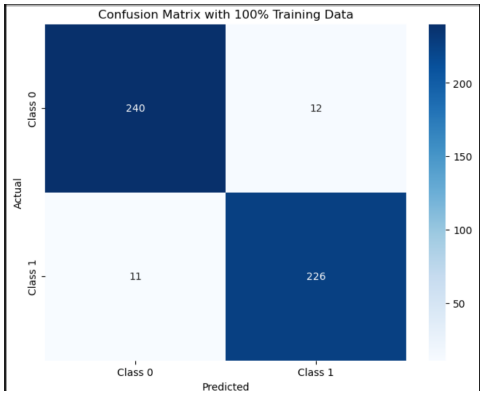

Figure 6: Confusion Matrix for 100% training data

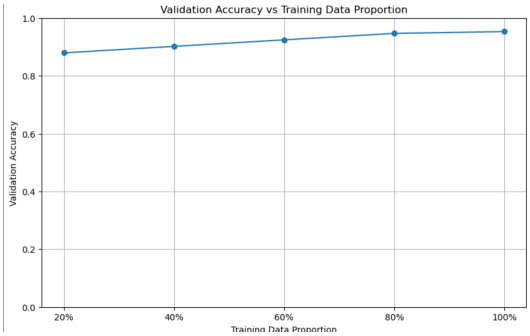

Figure 7: Validation accuracy vs fraction of training Data

- We found that the best model was for the 80% of the training data giving 0.95 accuracy on validation data.
- Trainable params: 7,513

1.2 Dataset 2: Deep Features Dataset

The features of each input are extracted using a simple deep neural network. This resulted in each input being represented using a 13×786 matrix which basically consists of 786-dimensional embeddings of each of the 13 emoticon features Initially, we tried a simple **KNN**. We chose `n_neighbors=1` based on the elbow method. It returned a very low score, so we decided to consider the input as a sequence of vectors and used models that learn patterns and dependencies among the feature vectors.

- We tried **LSTMs** because they are popular for sequence modeling. However, it gave accuracies of about 80% because its ability to capture long-range dependencies was not utilized effectively in this case.
- We also tried **CNNs** because they capture local patterns well. It didn't give great results, possibly because we couldn't get the right vector embeddings.
- Finally, we tried **TCNs**.
- **TCNs** are designed to capture temporal dependencies in sequences using convolutional layers, making them suitable for this task where order matters. TCN uses residual connections, which help in training deeper networks. *This allows the model to effectively model more complex sequential patterns without suffering from vanishing gradients.*

Observations:

1. Even when we used a fraction of the 768 dimensions of the vector, we did not observe much drop in the accuracy. Thus, we concluded that many features were redundant. To reduce the number of parameters in the model, we used only the first half of the dimensions.
2. Many columns had the same vector in all the training inputs. These columns harm the classifier because they have the same values in both classes, thus not improving inter-class variance. Columns [1, 7, 12] were the only columns that did not have single unique values. Removing these redundant columns is beneficial as they do not help in separating the classes.

Results:
The validation accuracies for different fractions of training data were

- 20% : 95.71%
- 40% : 97.14%
- 60% : 98.57%

- 80% : 97.14%
- 100% : 98.16%
- We found that the best model was for the 60% of the training data giving 98.57% accuracy on validation data.

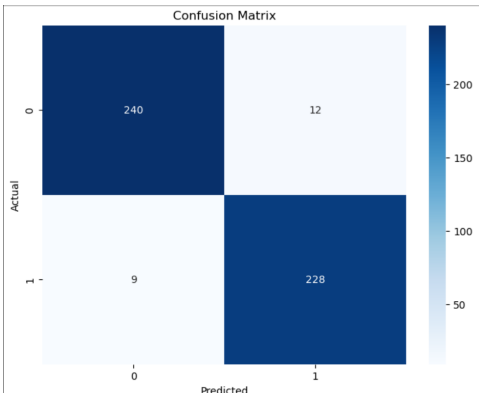

Figure 8: Confusion Matrix for 20% training data

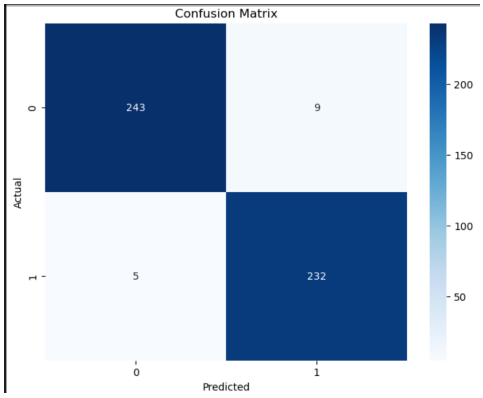

Figure 9: Confusion Matrix for 40% training data

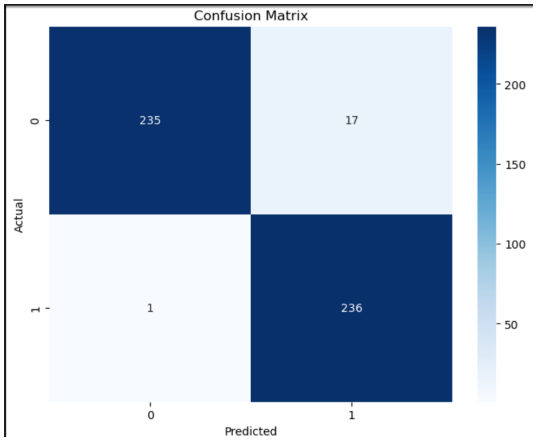

Figure 10: Confusion Matrix for 60% training data

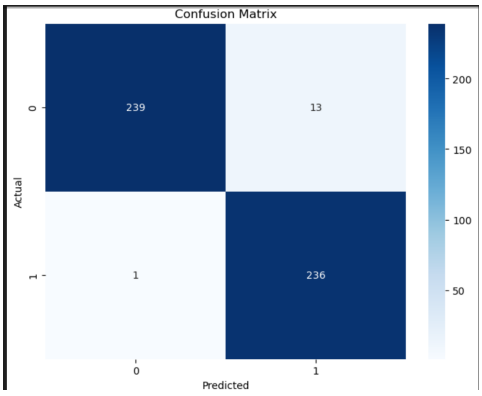

Figure 11: Confusion Matrix for 80% training data

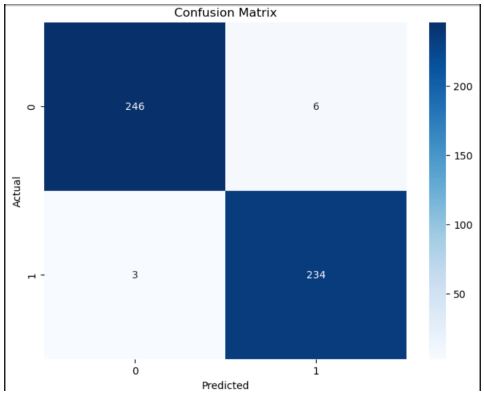

Figure 12: Confusion Matrix for 100% training data

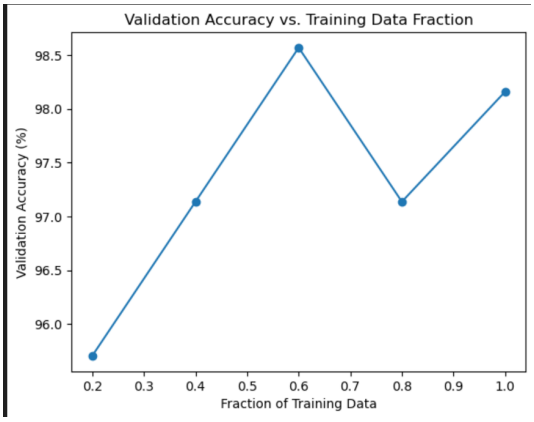

Figure 13: Validation accuracy vs fraction of training Data

- Trainable params: 9,222

1.3 Dataset 3: Text Sequence Dataset

The features of each input are given in form of a length 50 string consisting of 50 digits

- The key observation is that this dataset is similar to the emoticon dataset. Every example has a padding of zeros at the beginning, and every emoticon is encoded as a sequence of 3–5 digits in this dataset.
- Therefore, similar to the first dataset, we chose to train an **LSTM** model since it would learn both the local patterns (individual emoticons) and the long-term memory (the sequence of emoticons).
- Each digit is embedded as a 16-dimensional vector.
- The model is expected to have lower accuracy compared to the emoticon dataset because there is a loss of information in learning the encoding of the emoticons.

Other ML methods tried:

- Tried kernelized **SVM** with RBF kernel, polynomial kernel (degree = 3), and linear kernel but was consistently getting accuracy less than 70%. This prompted us to treat the input digits as a sequence rather than individual features.
- Also tried **n-gram** feature extraction (n = 3–6), thinking it would learn the digit encodings of each emoticon, but we could not achieve reasonable results.

Results
The validation accuracies for different fractions of training data were

- 20% : 67.89%
- 40% : 75.46%
- 60% : 82.21%
- 80% : 84.66%
- 100% : 88.14%
- We found that the best model was for the 100% of the training data giving 88.14% accuracy on validation data.

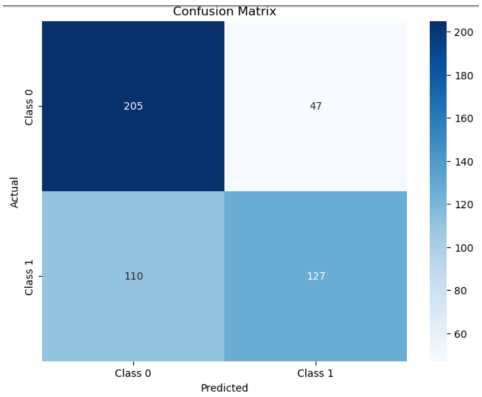

Figure 14: Confusion Matrix for 20% training data

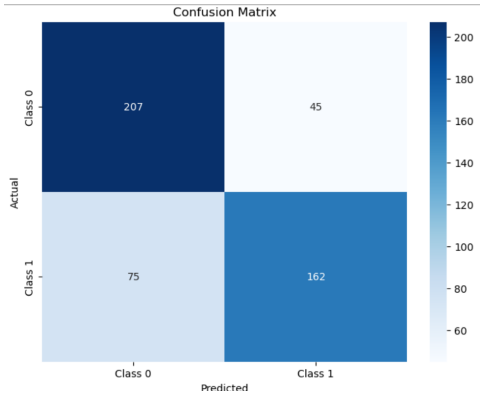

Figure 15: Confusion Matrix for 40% training data

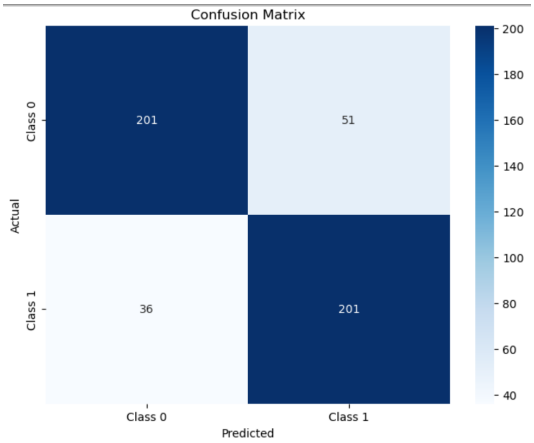

Figure 16: Confusion Matrix for 60% training data

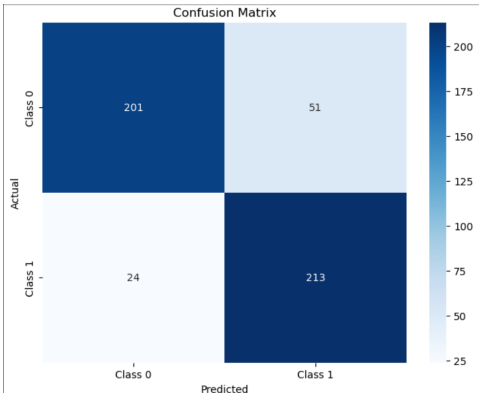

Figure 17: Confusion Matrix for 80% training data

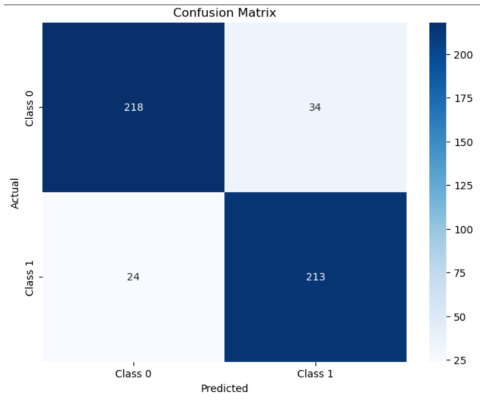

Figure 18: Confusion Matrix for 100% training data

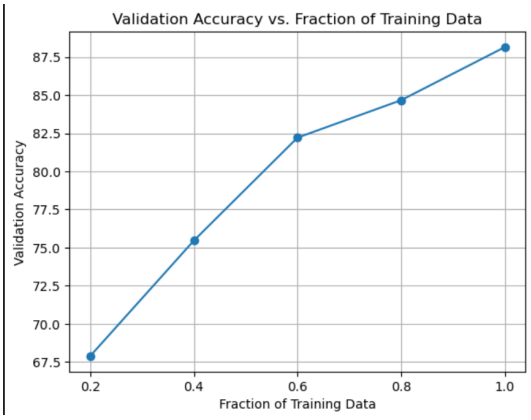

Figure 19: Validation accuracy vs fraction of training Data

- Trainable params: 9,481

2 Task 2

In Task 2, we combine all three datasets: **Emoticons dataset**, **Features dataset**, and **Text sequence dataset**.

- Our initial idea was to use the first dataset to find the digit encodings of each emoticon in the third dataset. This approach aimed to extract better features from the third dataset, but it was difficult to come up with a working algorithm.
- We trained an LSTM classifier on the first dataset, a TCN classifier on the second dataset, and another LSTM classifier on the third dataset. We reduced the parameters in each model considerably to ensure they adhered to the 10,000 parameter limit.
- We implemented a **Majority Voting** algorithm weighted by the validation accuracies of the individual models.
- The combined model is performing slightly worse than the best individual model, which is the TCN classifier for the second dataset.
- The Validation Accuracy of the combined voting model: 93.66%

```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
EmoticonLSTMClassifier                   --
├─Embedding: 1-1                         904
├─LSTM: 1-2                              1,728
├─Linear: 1-3                            38
├─Softmax: 1-4                           --
├─CrossEntropyLoss: 1-5                  --
==========================================================================
Total params: 2,670
Trainable params: 2,670
Non-trainable params: 0
==========================================================================
```

Figure 20: Model summary for Emoticon LSTM classifier

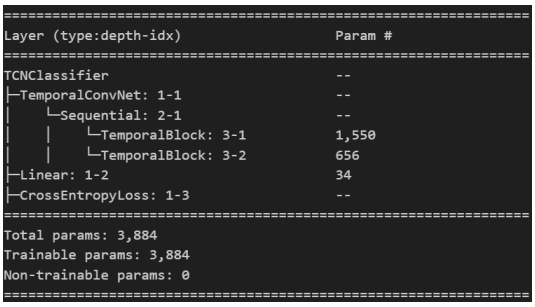

Figure 21: Model summary TCN Classifier

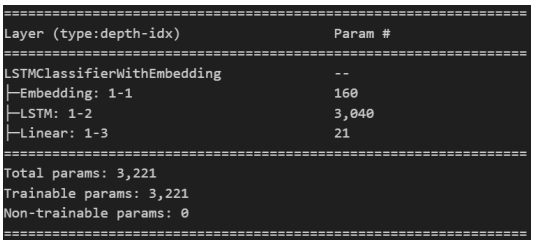

Figure 22: Model Summary for LSTM classifier with embeddings

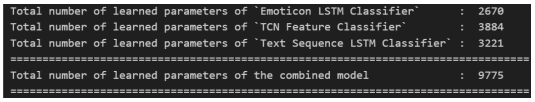

Figure 23: No. of Trainable parameters for combined model