CS771: Introduction to Machine Learning
Linear Depression

Aryan Singh
220229

Shashwat Agarwal
221004

Bhaumik Chawda
220292

Shubhanshu Mishra
221048

1 Introduction

In this mini-project, we were provided with three distinct binary classification datasets, all derived from a common raw dataset, but differing in terms of feature representations. The goal of the project was twofold:

1. Train binary classification models on each dataset individually, select the best model, and analyze its performance with varying amounts of training data.
2. Combine the three datasets to determine if learning from combined feature representations results in better performance.

The datasets included the **Emoticons as Features**, **Deep Features**, and **Text Sequence** datasets, each representing the same underlying task. We experimented with various machine learning models and approaches to optimize the performance on these datasets and report our findings in this document.

2 Dataset Analysis

2.1 Emoticons as Features Dataset

This dataset represents binary classification based on emoticon sequences. The preprocessing steps applied to the Emoticons dataset are as follows:

- **Tokenization:** Each emoticon sequence is tokenized using the `Tokenizer` class from Keras, converting the sequences into numerical arrays by mapping each unique emoticon to an integer.
- **Padding:** Sequences are padded to a uniform length of 13 emoticons using `pad_sequences()`. Shorter sequences are padded with zeros, and longer sequences are truncated to this length.

2.2 Deep Features Dataset

This dataset represents each input as a 13x786 matrix, where each row represents a 786-dimensional embedding of the corresponding emoticon feature. We used the following approaches to preprocess the data:

- **Reshaping:** Each input was originally in the shape (n_samples, 13, 786), but for model training, we flattened it into a (n_samples, 13 * 786) shape, effectively treating all 13 emoticon embeddings as one long vector.

- **Standardization:** To ensure the features had a mean of zero and unit variance, we applied StandardScaler on both the training and validation sets.

2.3 Text Sequence Dataset

In this dataset, each input is represented as a string of 50 digits. The digits correspond to the emoticon and deep feature representations in the other datasets. The following steps were applied to preprocess the Text Sequence Dataset:

- **String Processing:** Raw input strings were cleaned by removing unwanted leading characters. Additional processing steps removed specific substrings irrelevant to the task, resulting in a modified string.
- **Length Standardization:** Only strings with exactly 13 characters were retained to ensure consistency across all samples.

3 Machine Learning Models

For each dataset, we explored multiple machine learning models to find the best-performing model. Below is a summary of our chosen models and approaches for each dataset:

3.1 Emoticons as Features Dataset

For the Emoticons dataset, we experimented with several machine learning models, including Random Forest Classifier, Long Short-Term Memory, Support Vector Classifier and XGBoost Classifier. Below is a summary of the results and the best-performing models for this dataset.

3.1.1 Random Forest Classifier:

This is a bagging algorithm which seemed to understand the links between different features and make classification by understanding appropriate conditions. However considering the features individually did not appeared beneficial.

3.1.2 Long Short-Term Memory:

This is a type of RNN model which is used for sequential data and NLP works. This proved to generate best validation accuracy. This gives the idea that the emoticons store some sequential patterns and must not be used as individual features.

3.1.3 Support Vector Classifier:

The classifier was used with an idea that we might find some good hyperplane. Again the issue was that we were splitting the input emoticon string into individual features.

3.1.4 XGBoost Classifier:

This is a boosting algorithm that we used in hope that while training after every iteration this will correct on the previous losses and might give some significant increase in the accuracy score. Eventually the same wrong thought process was there while using this model too.

3.2 Deep Features Dataset

For the Deep Features dataset, we experimented with several machine learning models, including Logistic Regression, Decision Trees, Random Forests, and SVM. Below is a summary of the results and the best-performing models for this dataset.

3.2.1 Logistic Regression:

With a deep features dataset we thought of using the traditional models as they themselves might understand the patterns in the dataset. Starting with the very basic classifier Logistic Regression. Logistic regression successfully understood the dataset and gave good results.

3.2.2 Decision Tree Classifier:

After LR we tried more models to check for some improvements and applied decision trees. This proved to be rather a simpler model for the given case and gave lower accuracy than logistic regressor.

3.2.3 Random Forest Classifier:

A bagging classifier that builds weak decision trees on a subset of the dataset. With this way we thought it might capture the information of each feature in a more descriptive manner. This gave the best results for this dataset.

3.2.4 Support Vector Machine (SVM):

Thought behind using this model was to find some good hyperplane that finds a good split in the dataset. We did found a good enough accuracy but not the best.

3.3 Text Sequence Dataset

For the Text Sequence dataset, we experimented with several machine learning models, including Logistic Regression, Decision Trees, Random Forests, and SVM. Below is a summary of the results and the best-performing models for this dataset.

3.3.1 Gradient Boosting Classifier:

The features found after the pre-processing appeared to be good and might work with traditional models itself. First model attempted was GBM. Accuracy of about 73% gave an idea that it is understanding the patterns to some extent. Issue might be with the way this model is trained. It takes a subset if the whole dataset in which it might have miss understood some patterns in the dataset.

3.3.2 LSTM Model:

In an attempt to understand the features itself from the dataset after preprocessing we applied the LSTM model. This did gave the best results in terms of accuracy, however, the trainable parameters were more than the allowed limit so we dropped this.

3.3.3 LightGBM Classifier:

A boosting classifier that learns on the whole dataset simultaneously. This gave significant results and gave the idea that boosting algorithms are able to capture the patterns among the features.

3.3.4 XGBoost Classifier:

Another boosting algorithm tested on the dataset which gave the accuracy score above the threshold.

4 Experiment Results

4.1 Performance on Individual Datasets

For each dataset, we experimented with varying the amount of training data (20%, 40%, 60%, 80%, 100%) and plotted validation accuracy against the size of the training data for best model.

- **Emoticons Dataset:** The best model is **Long Short-Term Memory** with accuracy of 98.47%. The results from the various machine learning models trained on the text sequence feature dataset are summarized below:

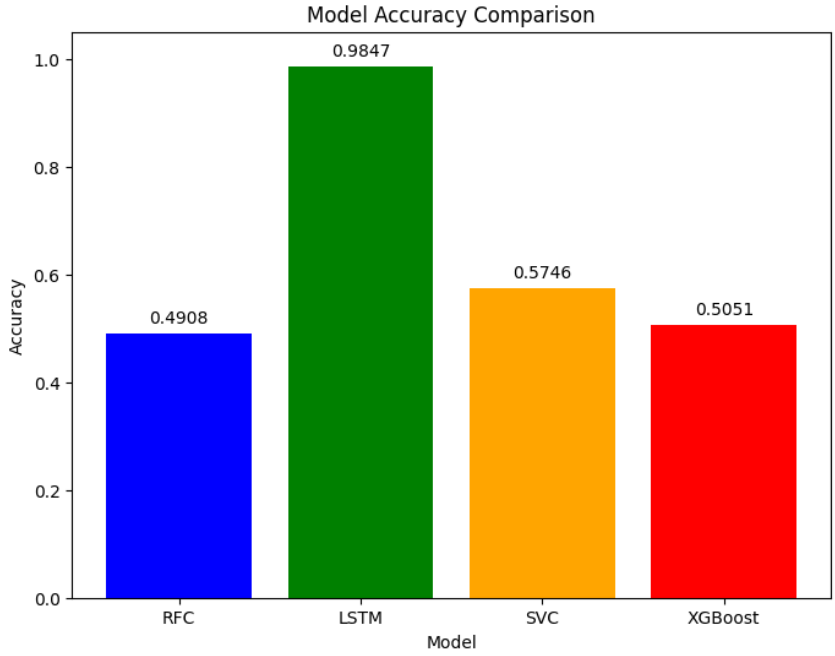

Figure 1: Comparison of various models for dataset 1

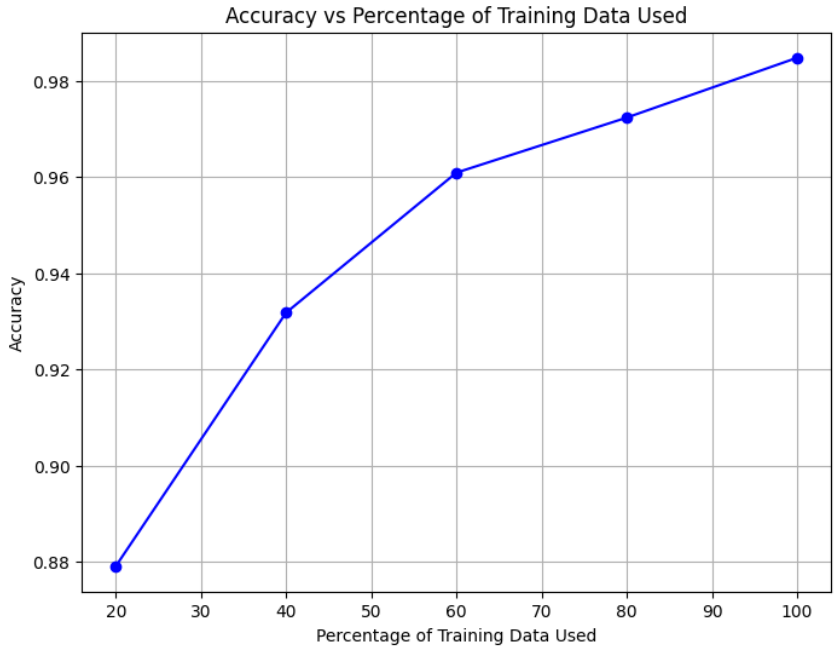

Figure 2: Proportion of Training Data vs. Accuracy for best model

- **Deep Features Dataset:** The best model is **Random Forest Classifier** with accuracy of 98.36%. The results from the various machine learning models trained on the text sequence feature dataset are summarized below:

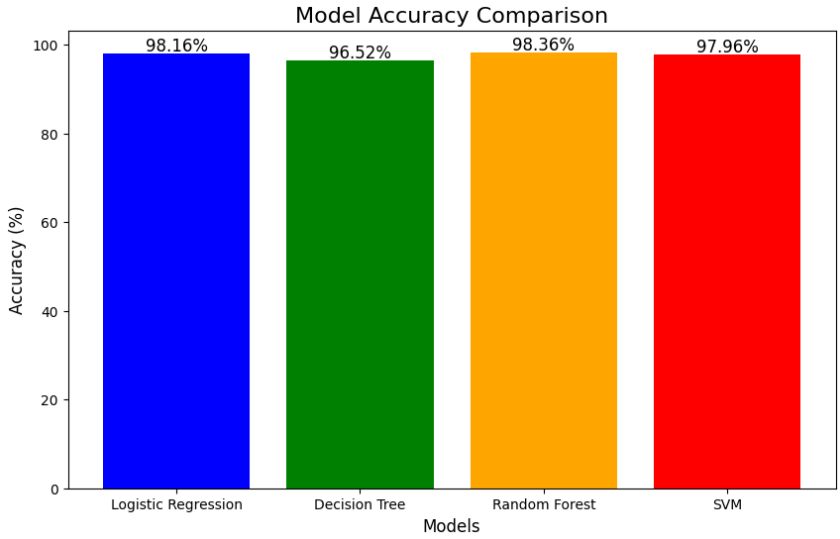

Figure 3: Comparison of various models for dataset 2

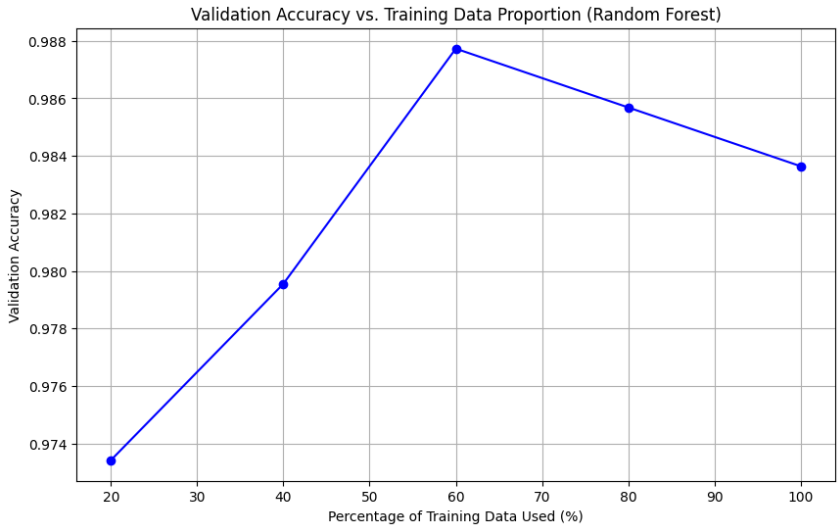

Figure 4: Proportion of Training Data vs. Accuracy for best model

- **Text Sequence Dataset:** The best model is **XGBoost Classifier** with accuracy of 87.20%. The results from the various machine learning models trained on the text sequence feature dataset are summarized below:
  Note: We don't consider LSTM as the best model because trainable parameters is more than 10000.

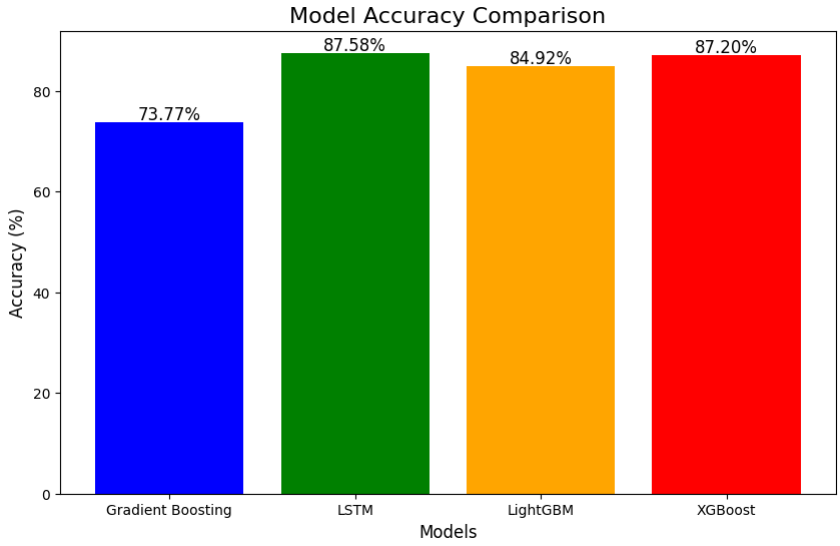

Figure 5: Comparison of various models for dataset 3

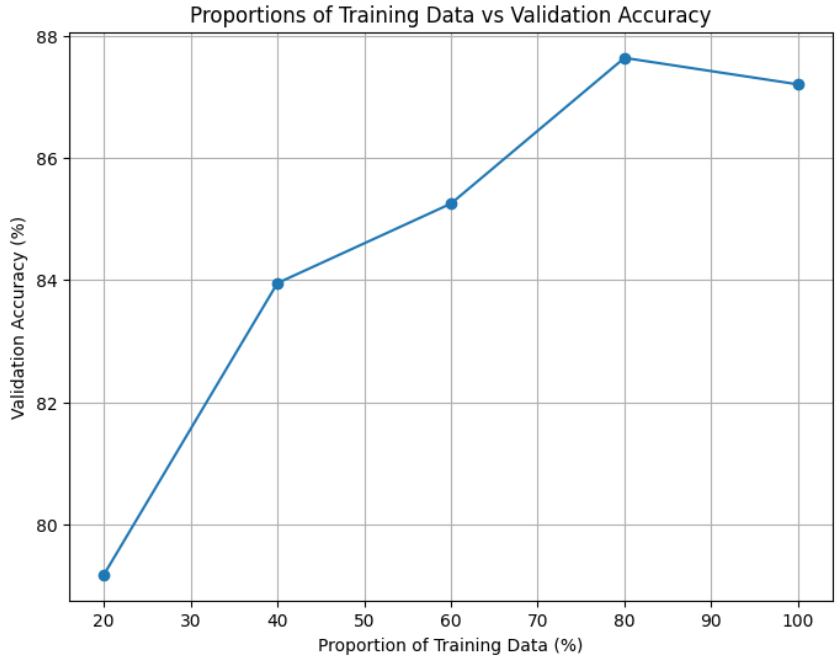

Figure 6: Proportion of Training Data vs. Accuracy for best model

4.2 Predictions on Test Sets

For each of the three datasets, we used the best-performing models to make predictions on the test sets. The predictions were stored in text files as per the guidelines.

5 Combining Datasets (Task 2)

5.1 Dataset Preparation

We combined multiple feature sets: emoticon features, text sequences, and deep embeddings. The following datasets were used:

1. **Emoticon Dataset:**
   This dataset consists of a sequence of emoticons that are relevant to the classification task. We extracted the `input_emoticon` feature and its corresponding `label` for both training and validation datasets.
2. **Text Sequence Dataset:**
   This dataset contains sequences of text input that complement the emoticon features. The **input_str** feature from this dataset was used along with the `label`.
3. **Deep Feature Dataset (Embeddings):**
   The embeddings (deep features) were pre-extracted from a neural network model. These features provide a dense representation of input data.

We performed **one-hot encoding** for both the **emoticon dataset** and the **text sequence dataset**. The deep feature dataset was scaled using a **StandardScaler** to normalize the input data. After encoding and scaling, the data was concatenated to form a combined feature set.
The combined feature set consisted of:

- One-hot encoded emoticon features
- One-hot encoded text sequences
- Scaled deep feature embeddings

5.2 Model Training: Random Forest Classifier

The Random Forest model was trained on the combined feature set. Here's a summary of the steps:

- **One-Hot Encoding:**
  Both the emoticon and text sequence datasets were one-hot encoded. This transformed categorical features into a sparse binary matrix.
- **Standardization:**
  The deep feature embeddings were standardized to have zero mean and unit variance.
- **Model:**
  We trained a **Random Forest Classifier** with 100 trees and a random seed for reproducibility. Random Forest is a powerful ensemble learning method that aggregates multiple decision trees to improve accuracy and reduce overfitting.
- **Results:**
  After training, predictions were made on the validation set. The model achieved an **accuracy of 97.96**% on the validation set, showing that combining these feature sets results in robust performance.

5.3 Model Training: Support Vector Machine (SVM)

In addition to Random Forest, we also trained an SVM model with a linear kernel using the combined dataset.

- **One-Hot Encoding:** Similar to the Random Forest model, both the emoticon and text sequence datasets were one-hot encoded.
- **Standardization:** The deep embeddings were standardized to ensure the SVM model worked well with features of different scales.

- **Model:** A **Support Vector Machine (SVM)** classifier with a linear kernel was used. SVMs are known for their effectiveness in high-dimensional spaces.
- **Results:** The SVM classifier achieved an **accuracy of 97.96**% on the validation set. The high accuracy achieved is majorly due to the deep features dataset. Due to the presence of other two datasets during classification diluted it and returned slightly lower accuracy than that by the model trained only on the deep features dataset.

6 Conclusion

In this mini-project, we trained multiple binary classifiers on three distinct datasets, each with unique feature representations. We found that:

- **Best individual models:** Long Short-Term Memory for the Emoticons dataset, Random Forest for the Deep Features dataset, and XGBoost Classifier for the Text Sequence dataset.
- **Combining datasets:** Feature concatenation yielded a modest improvement in accuracy compared to models trained on individual datasets, while ensemble learning was not as effective.