CS771: Mini-Project 1

Anushka Singh 220188

Arush Upadhyaya 220213

Aujasvit Datta 220254

Pahal Dhruvin Patel 220742

Pranav Agrawal 220791

1 Introduction

The project aims to predict the binary labels of an emoticons dataset.

2 Dataset

There are 3 datasets provided for the project:

1. **Emoticons dataset**: This comprises of emoticon strings and their corresponding binary labels. Each string consists of a sequence of 13 emoticons.
2. **Features dataset**: This dataset consists of features for the input emoticons dataset. Each row corresponds to a feature vector for the corresponding emoticon string.
3. **Text Sequences dataset**: This dataset consists of text sequences for the input emoticons dataset. Each row corresponds to a text sequence of size 50, consisting of digits 0 to 9 for the corresponding emoticon string.

3 Emoticons Dataset

3.1 Data Analysis

On analysing the count and distribution of the emojis (*cf.* `emoticon/eda.ipynb`), we found out that 10 out of 13 emojis in each string were redundant, as they were present in every string. These 10 emojis corresponded to 7 distinct emojis. On careful analysis of the entire dataset, this turned out to be true, leaving exactly 3 emojis for each input string.

3.2 Data Cleaning/Preprocessing

To make the dataset meaningful, these dummy emojis were dropped from the dataset. This was done by counting the frequency of each emoji and dropping the ones that occur in all entries.

Next, since our data is categorical, we split the strings into three columns, ith column denoting ith emoji, then we one-hot encode the data. This transforms each column into k columns where k is the number of unique emojis in the column, this new k dimensional vector is basically one hot vector for this column, where the i^{th} element is 1 if the emoji is the i^{th} unique emoji in the column, all others are 0.

This is done for each of the three columns, and then the three one-hot vectors are concatenated and flattened to form a single vector of size $k = k_1 + k_2 + k_3$, where k_i is the number of unique emojis in the i^{th} column.

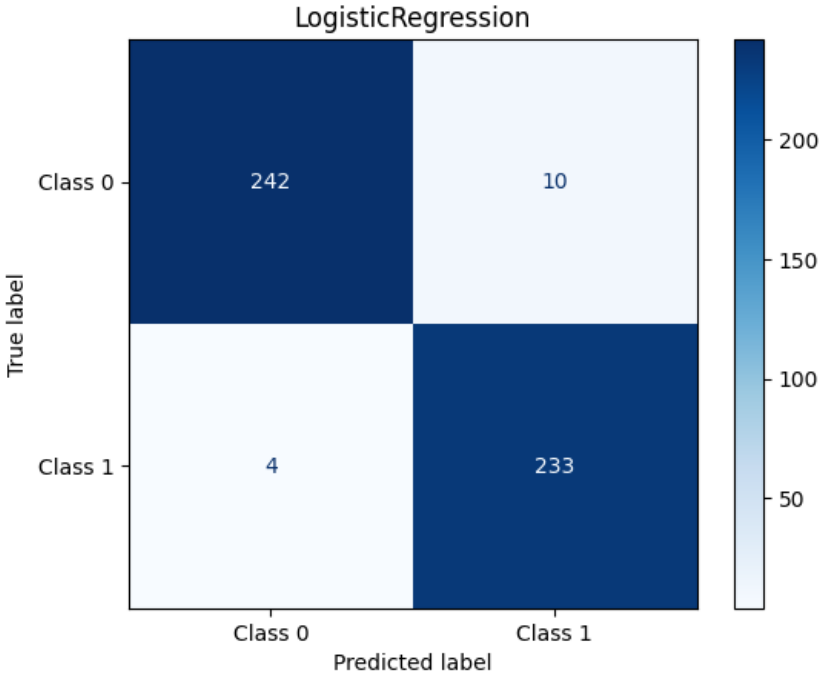
It is found that

$$k_1 = 207,\ k_2 = 207,\ k_3 = 87 \text{ giving } k = 501$$

3.3 Model Choice

1. **Logistic Regression**
   - We used `scikit-learn`'s `LogisticRegression` and applied `GridSearchCV` to get the best parameter choice. The total parameter count was 502.
   - The accuracy in the model shows a consistent improvement as the dataset size increases.
   - This trend suggests that the model benefits from larger datasets, with a gradual improvement in accuracy.

Dataset Size	20%	40%	60%	80%	100%
Accuracy	88.66%	95.38%	96.59%	96.42%	97.14%

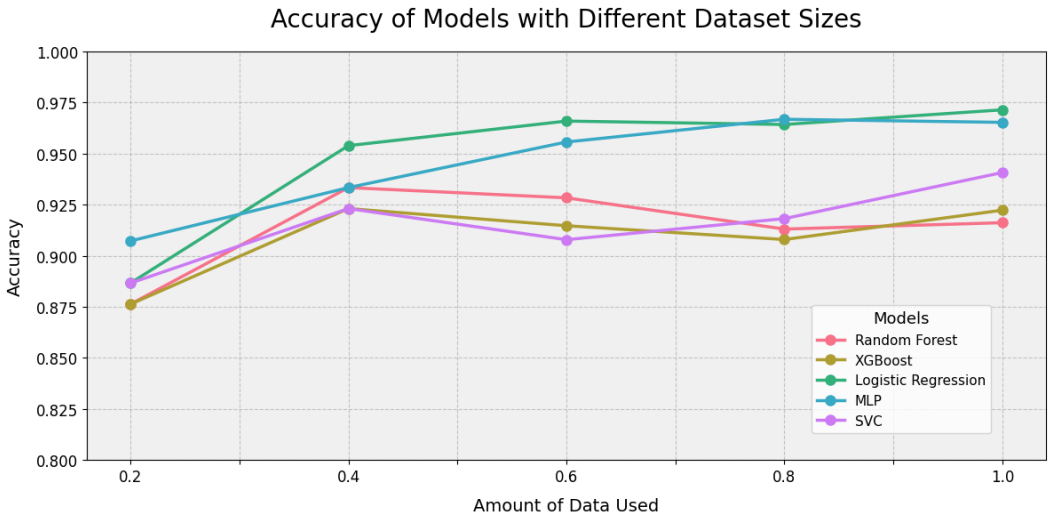


3.4 Results

Final validation accuracy using **Logistic Regression** was **97.14%**.

3.5 Experiments

Dataset Size	Random Forest	XGBoost	Logistic Regression	SVC
20%	0.876	0.876	0.918	0.887
40%	0.933	0.923	0.944	0.923
60%	0.928	0.915	0.952	0.908
80%	0.913	0.908	0.954	0.918
100%	0.916	0.922	0.949	0.941



4 Features Dataset

4.1 Data Analysis

The feature dataset contains 786-dimensional embeddings for each of the 13 emoticons in Dataset 1. However, for our models, we simply flattened these embeddings into a single feature representation, which we used directly as input.

It was also observed that these embeddings can be generated using a pre-trained BERT model, which transforms the text representation of each emoticon into a dense numerical vector. This method established a linear relationship between the given data and the embeddings we obtained from BERT.

4.2 Data Cleaning/Preprocessing

4.3 Model Choices

1. **Logistic Regression:** Of all the models used, the best accuracy was given by `scikit-learn`'s `LogisticRegression`. The total parameter count was 9985.

Dataset Size	20%	40%	60%	80%	100%
Accuracy	95.87%	97.95%	97.61%	98.72%	98.77%

4.4 Results

Final validation accuracy using **Logistic Regression** was **98.77%**.

4.5 Experiments

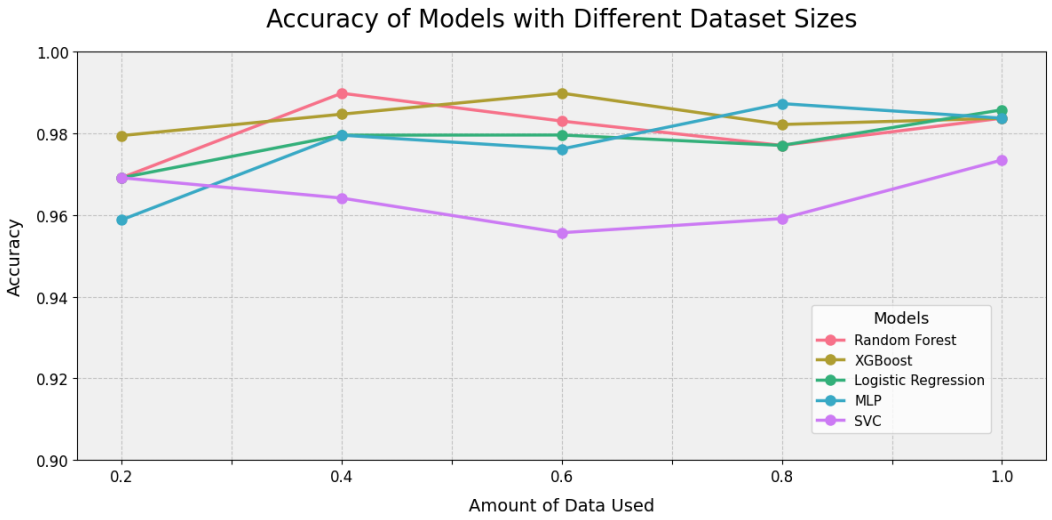
1. **Random Forest** : gave an accuracy of 98.16

2. **XGBoost** : gave an accuracy of 98.36

3. **Multi-Layer Perceptron (MLP)** : gave an accuracy of 98.36

4. **Support Vector Classifier (SVC)** : gave an accuracy of 97.95

Dataset Size	Random Forest	XGBoost	Logistic Regression	MLP	SVC
20%	0.969	0.979	0.969	0.959	0.969
40%	0.989	0.984	0.979	0.979	0.964
60%	0.983	0.989	0.979	0.976	0.955
80%	0.977	0.982	0.977	0.987	0.959
100%	0.984	0.984	0.986	0.984	0.973



5 Text Sequences Dataset

5.1 Data Analysis/Cleaning

Careful examination of the data reveals that text sequences have some padding of 0s at the beginning. This suggests that the text sequences could be some sort of encoding of the emoticons.

Like the first dataset, the text sequences dataset also has some dummy values, which are substrings present in all entries.

We find each dummy substring by running frequency analysis on the substrings of the dataset. These substrings have been picked out by inspection, with two defining characteristics:

1. Size 3-5 digits.

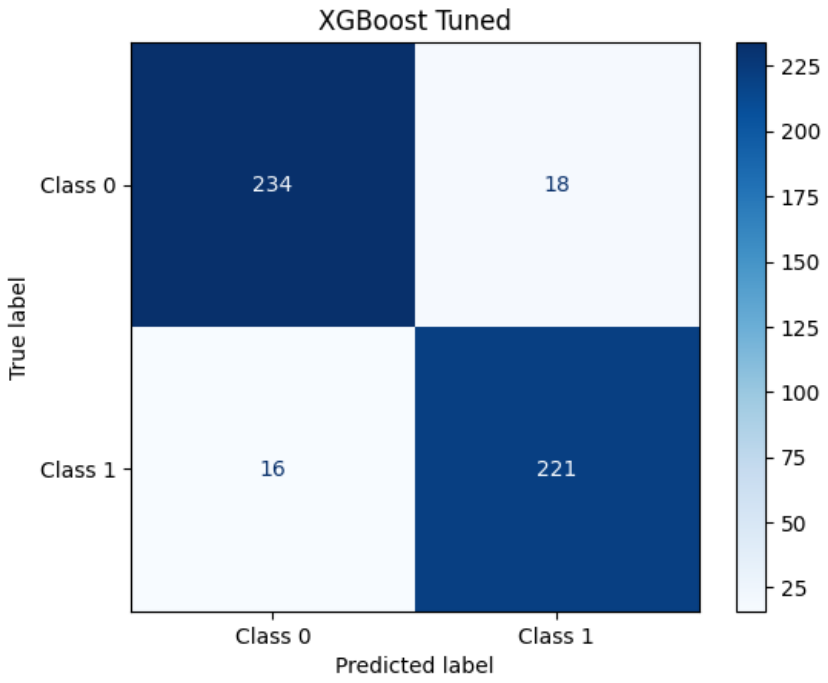
2. Occuring 7080/14160 times.

Then, all the dummy substrings are removed from each input string. This leaves varying strings of size ranging from 12 to 15. We then run a vectorizer to extract N-Grams varying in length from 3-5 digits, and process the datasets into sparse matrices.

5.2 Model Choices

1. **XGBoost**: Used `GridSearch` tuning to create an optimal `XGBoost` classifier that finally performed on the processed data.

Dataset Size	20%	40%	60%	80%	100%
Accuracy	81.44%	90.26%	88.74%	91.82%	93.05%



5.3 Results

Final validation accuracy was **93.05%**.

5.4 Experiments

We tried to learn the mapping of each emoji from the first to the third dataset. We attempted to create a sequence-to-sequence classifier, finally getting 75% accuracy; This approach was abandoned.

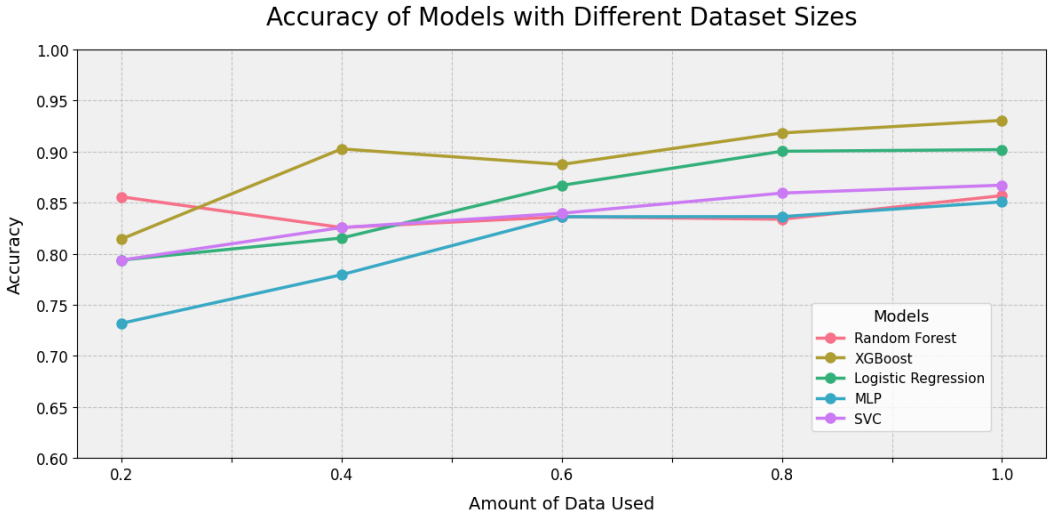
We also attempted a variety of models:

1. **Logistic Regression** performed as well as XGBoost on the processed dataset pre-tuning.

2. **Autoencoder** for dimensionality reduction of the dataset. However, it could not effectively 'learn' that the dataset had redundant substrings.

3. **1-dimensional Convolutional Neural Network (CNN)**: We attempted to use a simple Neural Network architecture to learn the internal structure of the database, but only reached 75% accuracy.

Threshold	Random Forest	XGBoost	Logistic Regression	MLP	SVC
0.2	0.855670	0.814433	0.793814	0.731959	0.793814
0.4	0.825641	0.902564	0.815385	0.779487	0.825641
0.6	0.836177	0.887372	0.866894	0.836177	0.839590
0.8	0.833760	0.918159	0.900256	0.836317	0.859335
1.0	0.856851	0.930470	0.901840	0.850716	0.867076



6 Task 2

We explored ensemble methods to combine the results of our top three individual models, achieving an accuracy of **98.36% with hard voting** and **97.95% with soft voting** on the validation dataset. However, both strategies underperformed compared to using only the second dataset and its best-performing model, which achieved an accuracy of **98.77%**. This outcome was anticipated, as the three datasets represent the underlying data in very different ways and contain varying amounts of information.

1. The first dataset consists solely of emoticon sequences, which by themselves do not convey much about the underlying data.

2. While experimenting with the third dataset, we discovered that each emoticon was mapped to a **fixed sequence of 3-5 characters**, and the inputs for the third dataset were generated by concatenating the corresponding character sequences of the emoticons from the first dataset, padded with zeros to reach a total length of 50. We were unable to establish a meaningful connection between the emoticons and their character sequences. Instead, the character sequences appeared random, making it challenging for a machine learning model to learn from this dataset. This difficulty was exacerbated by the padding, which introduced non-informative data, and the inconsistent lengths of the emoticon sequences. As a result, it was much harder for the model to effectively learn from this dataset.

3. The second dataset provided the richest representation of the underlying data because it was extracted using a deep learning framework, which creates dense embeddings that capture complex patterns and relationships within the data. The deep learning framework was able to capture subtle dependencies and intricate patterns, which is why the model trained on this dataset performed the best, achieving an accuracy of **98.77%**.

4. Our hypotheses were validated by the fact that we achieved best performance on the second dataset, followed by the first and second datasets respectively.(**97.14%** on the first dataset, **98.77%** on the second dataset, and **93.05%** on the third dataset).

Additionally, we experimented with fitting machine learning models such as **XGBoost** and **linear regression** on the outputs of the three best models, but this approach only yielded a validation accuracy of **96.73%**. Furthermore, the training accuracy was 100%, indicating **severe overfitting** and making this method unreliable. Again, this was expected, because the resulting predictions from the three models. This result was not surprising, as the predictions from the three best models alone do not contain sufficient information to reliably assess the quality of those predictions. Without additional context or data, the model's outputs lack the necessary depth to make accurate evaluations.

6.1 Results

Final validation accuracy using **Logistic Regression** on **Dataset 2** was **98.77%**.

6.2 Experiments

We tried the following methods to combine the predictions from the best performing model for each dataset.

1. **Hard voting** : gave an accuracy of 98.36%
2. **Soft voting** : gave an accuracy of 97.95%
3. **Linear Regression** : gave an accuracy of 96.73%
4. **XGBoost** : gave an accuracy of 96.73%

Acknowledgments

We would like to thank Prof. Piyush Rai for giving us this opportunity to work on such an interesting problem where the same data has three different representation and his invaluable guidance throughout this course. It was a lot of learning, seen from the perspective of how does one deal with various types of data and the most valuable lesson that we have learnt is to have a careful look at the data to rule out insignificant features! That really reflects the lesson of how deep learning is not always the solution, its much more about learning meaningful representations of data, that is where real machine learning lies.