CS771 Mini Project 1 - Group 67 Transformers

Aayush Singh
220024

Aditya Gautam
220064

Akash Verma
220097

Ritik Shah
220894

Samarth Agarwal
220944

1 Introduction and Problem Statement

We were provided with 3 datasets each for training, validation and prediction. The datasets were different feature transformations of the same raw data and the problem statement was to learn binary classification models from each training dataset and predict the labels for the test data.

1.1 Overall Approach

Our team performed an analysis of the datasets to understand them. Post-analysis and preprocessing, we experimented with different methods of prediction including basic models like SVM (linear and rbf), XGBoost, Random Forest Classifiers and Logistic Regression. Additionally, Pre-trained models like BERT were tested. These performed well when the models were fined tuned for the dataset but that led to training a large number of parameters. Additionally for each model, we used GridSearchCV to find the best hyperparmaters for each percentage of training data **20, 40,60,80,100**, but the values of hyperparams mentioned in report and README file refers to those on a certain percentage (100% or 80%).

Task 1

2 Dataset 1

2.1 Initial naive attempt

The dataset's features are represented by 13 emojis, which we labelled as s1, s2, s3, ..., s13. Training various models on this raw dataset without applying any feature transformations resulted in only mediocre accuracy on the validation set, as illustrated in Figure 3.

2.2 Pre-processing

A frequency analysis revealed that s1 to s8 each had 213 possible emoji values, while s9 to s13 had 91 possible values. This allowed us to divide the input strings into two **distinct segments** for further analysis. Initially, through simple observation, and later confirmed through rigorous testing, we discovered that specific emojis (😛, 🛐, 😑, 🚼, 😣, 🖋) **consistently appeared** within the **s1 to s8** indices, while a different set of emojis (😑, 😣, 🖋) appeared within the **s9 to s13** indices. Importantly, the order or position in which these emojis appeared in their respective segments showed no discernible pattern.

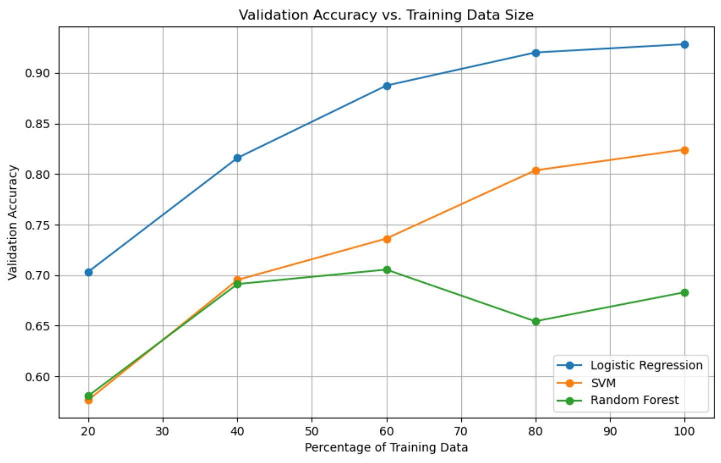

Figure 1: Validation Accuracy on Raw Data. The accuracy obtained was mediocre, which led us to experiment with feature transformations.

We hypothesized that the emojis occurring in every input string, regardless of the label, did not contribute meaningfully to label prediction. Based on this, we dropped five common emojis from the features s1 to s8 and three common emojis from s9 to s13, leaving two unique emojis in each segment. This reduced our feature set to a total of four emojis per input string.

2.3 Models Trained and their outcomes

For feature encoding, we used a straightforward approach: in each of the two segments, the presence of each unique emoji was represented as a 1 in a sparse binary vector. Training on this simple encoding of the features extracted in Section 2.2 significantly improved the model's accuracy on the validation dataset. Encouraged by these results, we proceeded with cross-validation to optimize the model and identify the best hyper-parameters for further improvement. The results are in Figure 2.

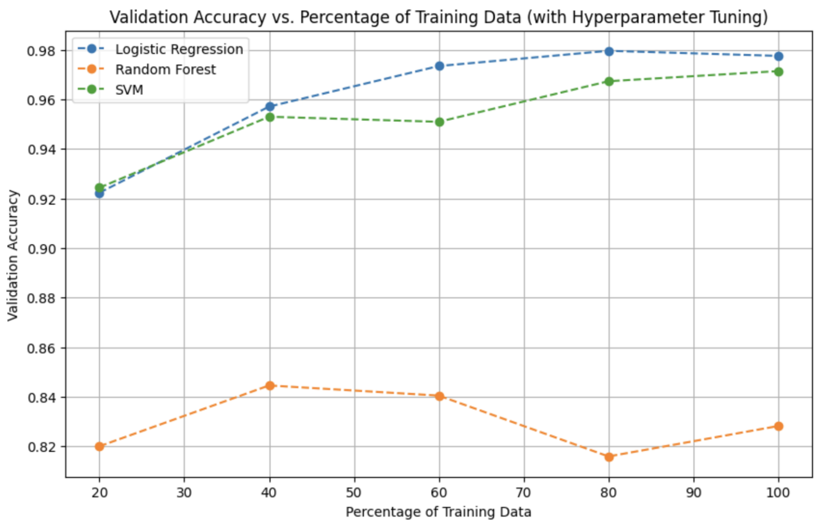

Figure 2: Validation accuracy after feature extraction. Among the models evaluated, Logistic Regression (with an L1 regularizer) yielded the best performance. The regularization hyper-parameter C switched between 1 and 10 across different training data percentages during CV. The validation accuracy was notably enhanced, reaching a peak of 98%. An accuracy of over 92% was achieved even when training on just 20% of the dataset.

2.4 Conclusion for Dataset 1

In conclusion, the best overall model for Dataset 1 was found to be **Logistic Regression** with **L1 regularizer**, having a regularization hyper-parameter $C = 1$. We made use of Scikit-learn's logistic regression implementation. The accuracy achieved was 97.75%.

3 Dataset 2

3.1 Cleaning and Preprocessing

The dataset was preprocessed by flattening the data into a 1-dimensional structure suitable for training models. On analysis of the data, no significant feature engineering or imputation of missing values was required. A key focus was on reducing the training data size, using {20%, 40%, 60%, 80%, 100%} of the training examples to analyze how the model performance varied with the data size. The training data was split into subsets accordingly.

3.2 Models Tried and Their Outcomes

We experimented with three different models, namely **Logistic Regression, Random Forest**, and **Linear SVM**. **GridSearchCV** was employed for hyperparameter tuning, and the models were trained on varying subsets of the training data. The results were analyzed based on the accuracy achieved on the validation set. Below is a graph the validation accuracies at different percentages of training data:

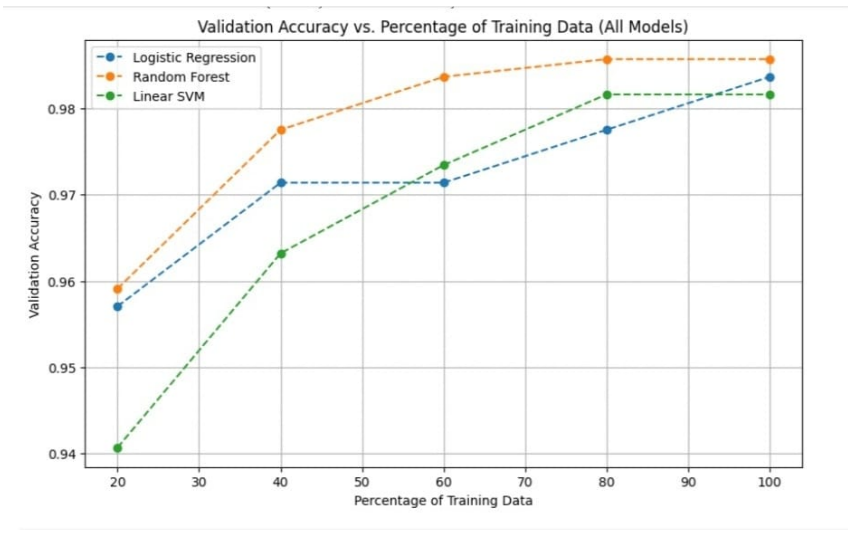

Figure 3: Validation Accuracy vs. Percentage of Training Data for Dataset 2

For each method, we performed **cross validation** for different hyperparameter values by using **GridSearchCV** method. Some details of the hyperparameters tested for are :

- **Random Forest** : Combinations of n_estimatiors and max_depth and min_samples_split.
- **Logistic Regression**: Different values of C and solver were used.
- **SVM**: Varying values of C and different types of kernels were tested.

3.3 Conclusion for Dataset 2

In conclusion, the best overall model for Dataset 1 was found to be the **Random Forest** model by ScikitLearn giving 98.57% accuracy for 100% of training data. The Best hyperparameters for Random Forest for the full dataset were found to be : max_depth: 20, min_samples_split: 2, n_estimators: 200. .

4 Dataset 3

Dataset 3 had 7080 inputs, each being 50-digit strings and label which was 0 or 1.

4.1 Pre-processing

4.1.1 Feature Engineering and dropping methods

To test the data for linear separation, we applied linear SVM with a **large C value** to allow small margins or slack. This resulted in nearly 50% accuracy on validation and train dataset hence proving that the data is **not linearly separable** After the failure of linear separation, we looked into analysing the class-wise distribution for every feature or every position of the 50 digits along with the joint distribution of features. From both, we discovered that the positional distribution matched closely for both classes. Additionally, we also tried analysis by applying **N grams vectorisation** to achieve dimensionality increases by having N-gram vectors of lengths 2,3,4 and (2,4) as well, while limiting the N to 10,000 the rationality being that a further model like SVM would learn 10,000 weights in such a case, yet the maximum accuracy was 58%.
The remarkable discovery was when we checked the **frequency of occurrences** of 2,3,4,5... length **sub-strings** which further revealed that certain sub-strings occurred with a frequency of 1 or 2 in **each entry** of the dataset. We dropped these strings from each input which largely helped us in**dimensionality reduction to 13 length** strings.

4.1.2 Overcoming Extra reduction in string length

Certain inputs had multiple occurrences of the frequent strings, as result post-dropping their length was less than 13 leading to issues while training and testing. First, we obtained the overlapping strings, for example, ”159614” which is a combination of the frequent strings ”1596” and ”614”. Using further functionality, we removed these frequent strings till we had 13 length strings. In overlapping cases, not all frequent strings were dropped and some were kept to allow all strings to be 13 in length,

4.2 Models trained and their Outcomes

Following is the result of the various models applied to the preprocessed dataset with testing on the validation dataset. The results are in Figure 4.

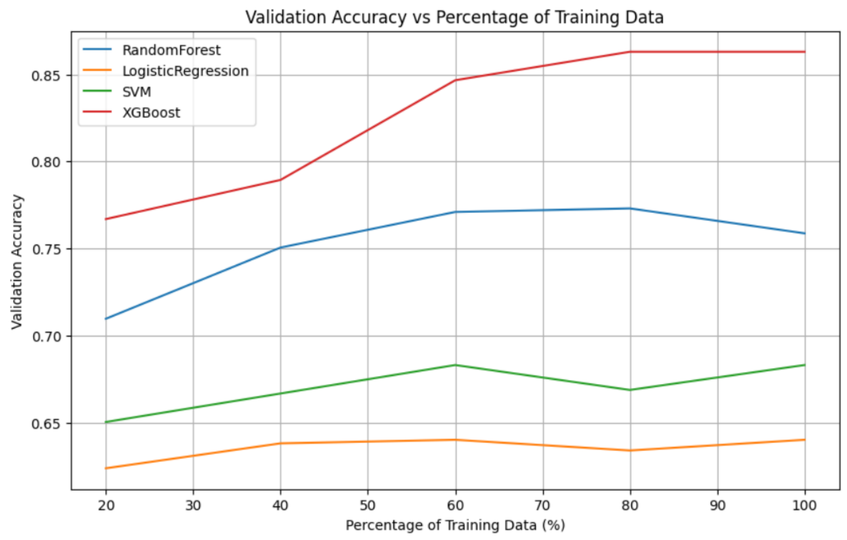

Figure 4: Validation accuracy after feature extraction. Among the models evaluated, Xgb yielded the best performance on all amounts of training data leading to 86% accuracy for 100% of the data

For each method, we performed **cross validation** for different hyperparameter values by using **GridSearchCV** method. Some details of the hyperparameters tested for are :

- **Random Forest** : Combinations of n_estimatiors and max_depth.
- **LogisticRegression**: C parameter was varied
- **SVM**: Various values of C along with different kernels were tested.

- **XGBoost** : Combinations of max_depth, n_estimators and learning_rate were carried out.

4.3 Conclusion for Dataset 3

Across various percentages of training data, **XGBoost** seems to be the model for appication on the preprocessed dataset. The val accuracy for 100% train data is 86.71%. The best hyperparameters : lr=0.1, max_depth = 6, n_estimators = 1400

4.4 Other Approaches and Why we Rejected them

Repeating substring patterns

On simple observation, we discovered that the third dataset had a lot of continuous subsequences which repeated frequently, like *'614'* and *'15436'*. We then proceeded with another approach which involved training a model on the basis of the presence of 3-digit, 4-digit and 5-digit substrings in each input string. This yielded an accuracy of **o** on the full dataset. But unfortunately, this approach used 110,000 trainable weights (more than the limit allowed) and took a noticeably longer time to train than the previous approach, without much improvement in validation set accuracy.

This observation paved the way for us the decipher what the numbers in Dataset 3 meant in Task 2, where we found out that the number string was nothing but a substitution cipher of the emoji string from Dataset 1, with zeroes added in the beginning to equalize the encoded string's length to 50. However since we were supposed to look at the Datasets individually for Task 1, we didn't go down that route now. Complete information can be found in Section 6.

The BERT Model

We also used Google's BERT Model to make a prediction. It gave an accuracy of 89.1% on the validation dataset. However, BERT is a very large natural language processing (NLP) model, and its trainable weights exceed our limit. In addition to that, it took overnight to train compared to the near-instataneous XGBoost model, but only a small improvement in accuracy was produced. So, BERT was rejected.

Task 2

5 The Combined Dataset

5.1 Initial Approach - Majority Prediction

Our initial approach involved selecting the best-performing model for each dataset. For each input, we would then use these models to make predictions and determine the majority value as the final label. Specifically, if all models predicted a label of 1, the final label would be 1. If, for instance, two model-dataset pairs predicted a label of 1 and one model-dataset pair predicted a label of 0, the final label would be 1, and so on. The following approach performed better in terms of accuracy by a large margin (nearly 8%) and hence was selected.

5.2 Final Approach - Concatenation

The training data for the three datasets are identical, with each being a feature transformation of the others (as described in the problem statement). For each dataset, we extracted features using separate models, then concatenated the features derived from datasets 1, 2, and 3. A combined dataframe was created from these concatenated features, which was subsequently used to train various models. We believe that concatenation of features allows the models to learn weights for the most reliable features across all 3 datasets.The performance of these models, as shown in the provided plot, was evaluated at different input percentages. Notably, the Support Vector Classifier (SVC) yielded the best result at 80%

input. The model parameters were optimized using cross-validation. The best parameters correspond to using an Radial Basis Function (RBF) kernel with C=10. Hence, we infer that it is quite helpful to use the combined dataset as it yields a validation set accuracy of 99.39%

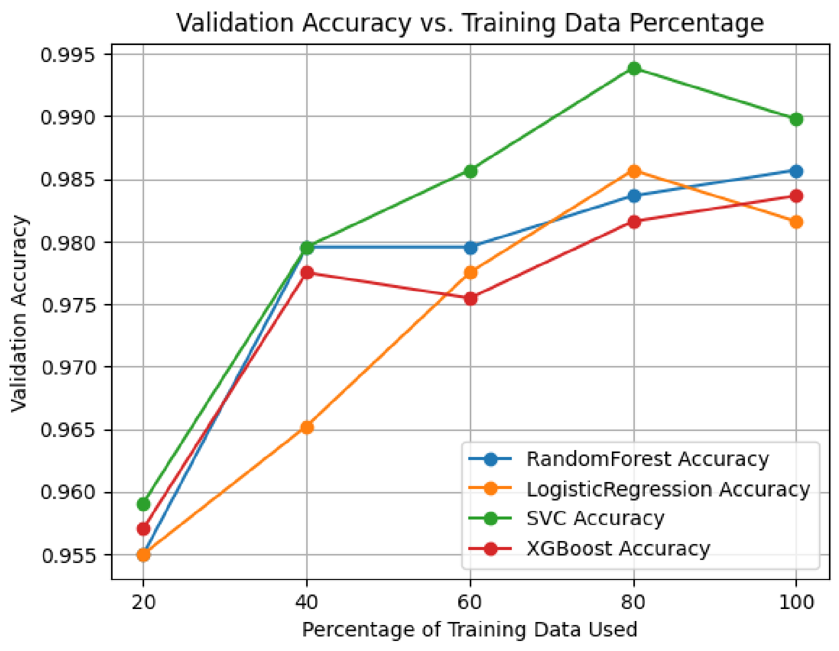

Figure 5: The peak accuracy was more than 99%. The accuracy was very good even on small parts of the training data.

5.3 Conclusion for Task 2 (The Combined Dataset)

In conclusion, the best overall model for Task2 was found to be **SVC** with best hyperparamaters being : C = 10, kernel = rbf with the highest accuracy of 99.39%, on 80% of the training dataset. The accuracy remained high even for lower percentages of training input as can be analyzed from the graph shown above.

6 An interesting observation: Dataset 3 is just a substitution cipher of Dataset 1, and further attempts

As hinted before in Section 4.5, we were able to conclude that the number string was nothing but a substitution cipher of the emoji string from Dataset 1, with zeroes added in the beginning to equalize the encoded string's length to 50. The emojis were substituted with 3/4/5 digit numeric codes. We, with much codebreaking and ingenious thinking as laid out in the attached `Decipher.ipynb` file, were able to decipher the code for all 214 emojis in Dataset 1. We confirmed this by constructing Dataset 3 from Dataset 1. The codes we found are also attached in `emoji_codes.csv`.

We tried looking for possible hidden information in the emoji codes to improve our current model. Despite much searching, we were only able to find only ephemeral evidence. For example, we were able to see that the scaled distribution of the logistic regression weights learned in Dataset 1, was somewhat similar in shape to the distribution of the emoji_codes. But we could not establish anything concrete regarding the codes.

Acknowledgements

We thank Prof. Piyush Rai for his guidance and explaining the concepts in a well defined manner and the TAs of the course for their support in execution of the course.

References

- The Scikit Learn Library and their models and other tools.
- Google's BERT model for helping explore the application of neural networks on Dataset 3.
- The XGBoost library.