CS771 Project Report for Group 90

Group Members: Anant Gauniyal(220132), Mudita Jain(220675), Pranshu Kumar(220799), Shrimi Agrawal(221036), Vikas(231080104)

October 22, 2024

1 Introduction

This project tackles a binary classification problem using three distinct datasets derived from the same underlying source, which remains undisclosed until submission. Each dataset represents the same task but differs in feature representations. The goal is twofold: first, to train individual models on each dataset, identify the best performer in terms of accuracy and minimal training data, and second, to explore whether combining the datasets enhances model performance. We will train models using varying amounts of data (20%-100%), evaluate their validation accuracy, and assess generalization on unseen test data. The challenge lies in selecting the best models, understanding the impact of feature representations, and minimizing data usage while maintaining accuracy.

2 Methods

In this section, we detail the machine learning models and algorithms employed to train on each dataset, highlighting the rationale behind model selection, hyperparameters used, and tuning procedures followed. We have imported the necessary libraries, including scikit-learn for machine learning, numpy for numerical operations, and matplotlib for plotting. Loading of datset is done through pandas. Using scikit-learn allowed us to easily implement machine learning algorithms and provides built-in functions for model evaluation

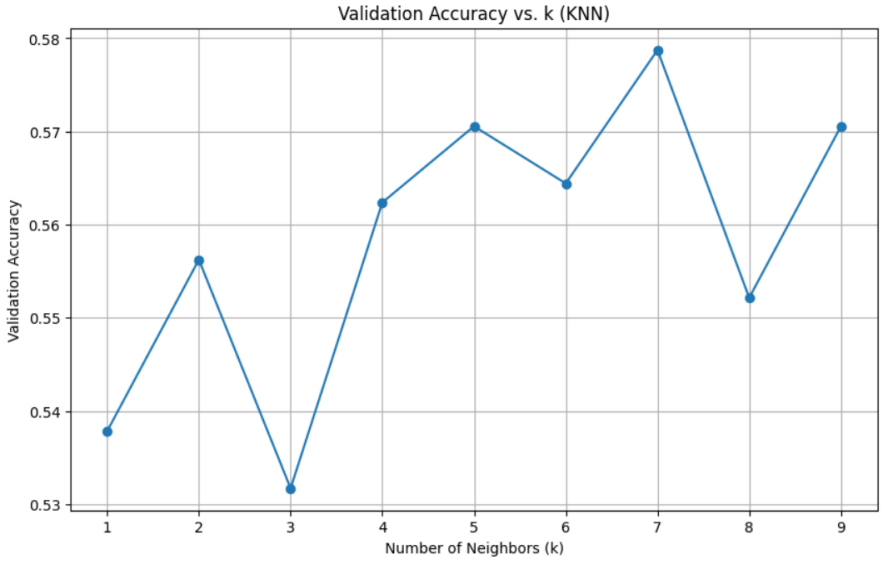
2.1 Task-1 : Training on Individual Datasets

In this task, we focus on training and evaluating machine learning models on each of the three distinct datasets.The goal is to identify the best-performing model for each dataset based on validation accuracy and other relevant metrics.

- **Emoticons as Features Dataset**

  1: KNN
  We trained the KNN classifier using a subset of the training data, specifically 40% of the total training set, to observe how well the model performs with different values of k. We varied k from 1 to 9 to find the optimal number of neighbors that yields the highest validation accuracy. For each value of k, we trained the KNN model and evaluated its performance on the entire validation set.

  Result and Performance
  The validation accuracy varied as k increased, but overall, KNN struggled to perform well on this dataset, with a maximum validation accuracy of 57%. The graph showing the variation of validation accuracy is given below:

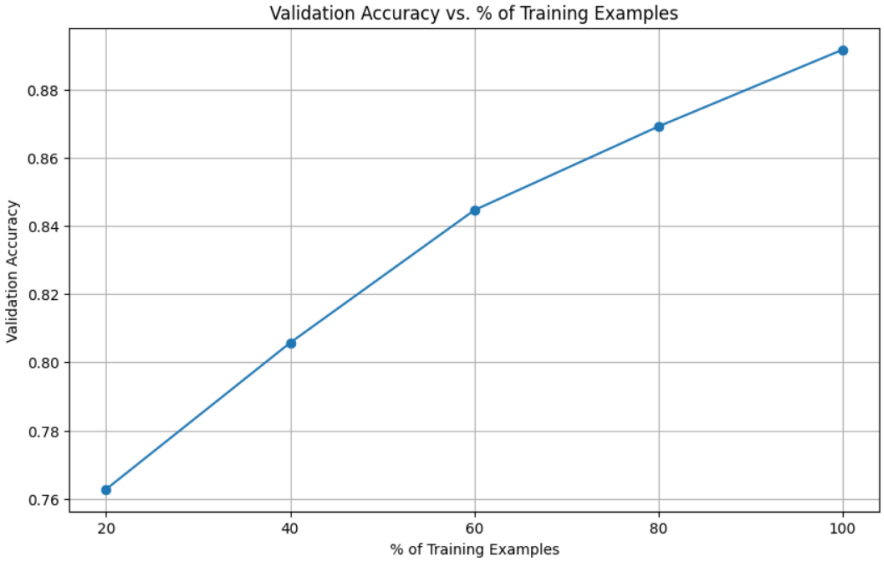


We reject this model as it does not give us enough effective accuracy over validaition set.

2: Logistic Regression:
In this specific data we have used the logistic regression from which we obtained 89% accuracy. It likely performed better because the feature space for emoticons might have contained distinct linear boundaries between the two classes, allowing it to effectively differentiate between them.

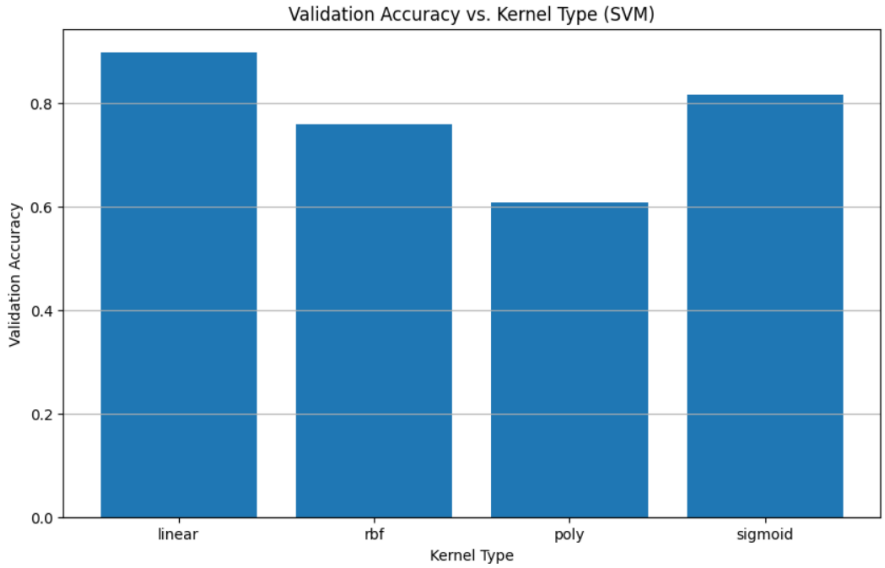
How did it optimise our result?
Logistic regression has built-in regularization techniques (like L2 regularization) which help prevent overfitting. This could have played a role in ensuring that the model generalized well to the validation set. In contrast, KNN might have struggled because it relies on proximity in feature space.

Performance
The plot of variation of validation accuracy is shown below:



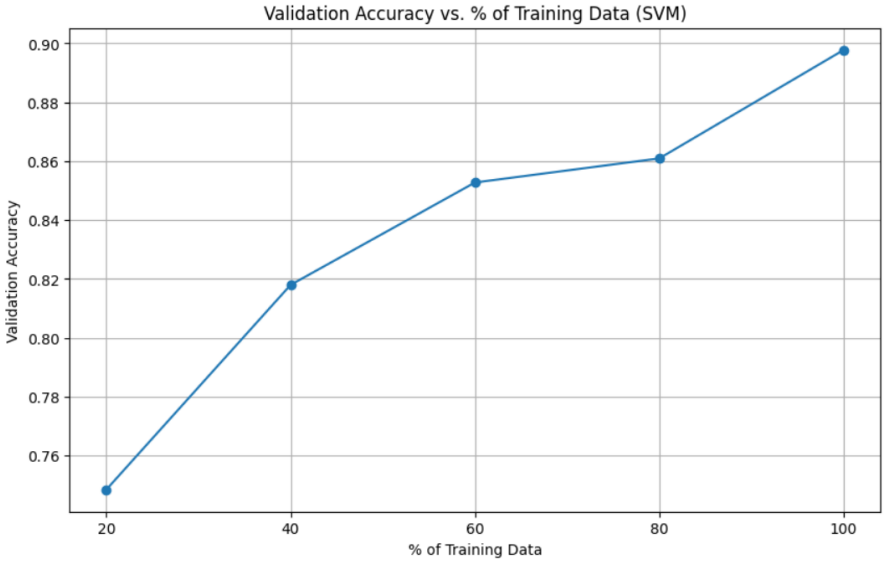
3: Support Vector Machine (Best performance)
SVM (Support Vector Machine) achieved 90% accuracy on the emoticon dataset, outperforming other models like KNN and logistic regression. Since SVM can be associated with kernels, they could further be used as a parameter to judge or determine the feature space of this dataspace. To perform this, we trained this model using different kernel types, namely, linear, rbf, polynomial and kernel.

How it optimised our result?
SVM aims to find the hyperplane that maximizes the margin between classes. We utilised this by plotting different kernel SVMs and we observed a clear variation.



On analysing the histogram, we observe that linear Kernel performs the best, achieving the highest validation accuracy, indicating that the dataset is likely linearly separable and benefits from a straightforward decision boundary.

Performance
We can now confidently say that liner kernel will give us the best result on vaildation set. A plot is shown below:

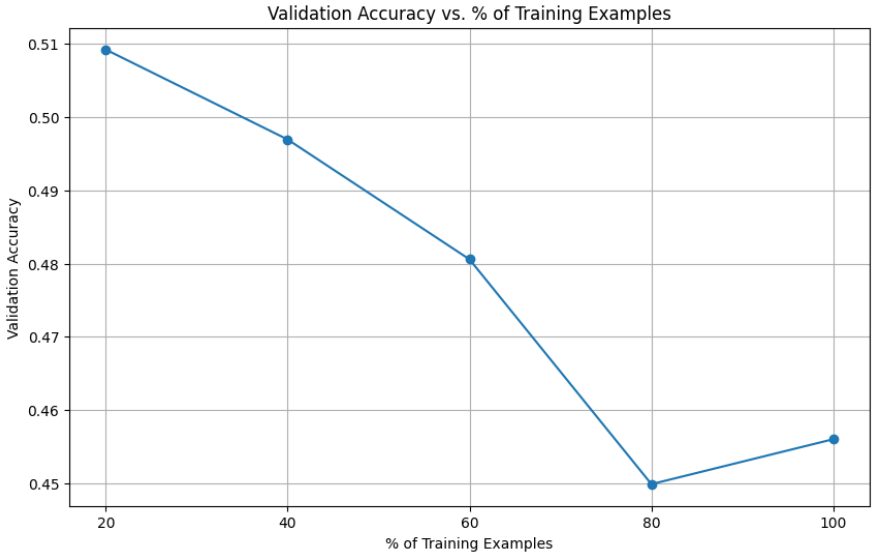


The linear kernel's simplicity leads to faster training and avoids overfitting, making it the most suitable choice for this task.

- **Deep Feature Dataset 1: Logistic Regression + Averaging**: In this approach, we averaged the 13 embeddings of size 786 to create a single 786-dimensional feature vector for each input. Logistic Regression was then applied to this averaged representation to perform binary classification.

  Why averaging: Averaging helped reduce dimensionality, but the linear nature of Logistic Regression may not capture the complex relationships inherent in the deep features.

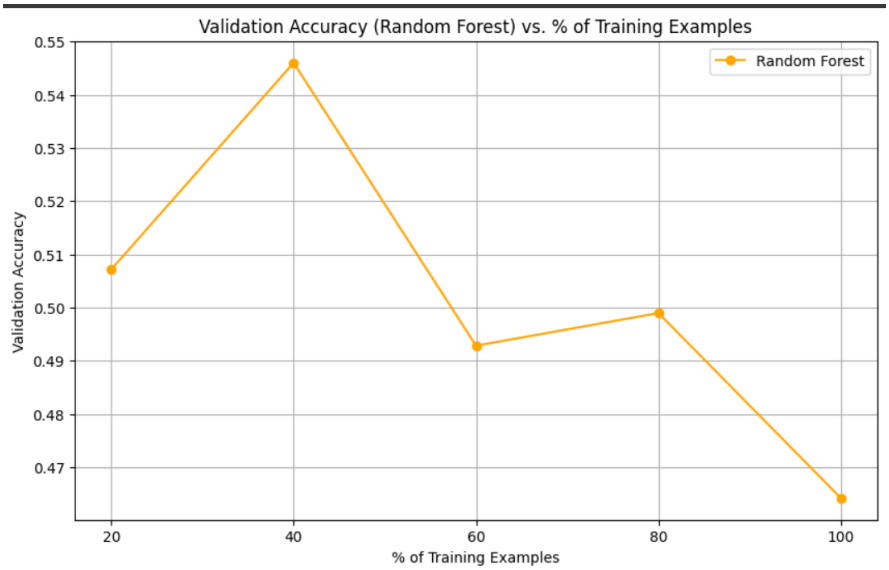
  Performance



2: Random Forest using averaging
Similar to the Logistic Regression approach, the 13 embeddings were averaged to form a single feature vector. A Random Forest classifier, which is an ensemble of decision trees, was then employed.

How it optimised our result
As an ensemble method composed of multiple decision trees, Random Forest can model complex, non-linear relationships between the input features and the target variable. Each tree in the forest makes splits based on feature values, allowing it to learn intricate patterns in the data.

Performance
The performance of dataset on applying Random Forest was plotted and is shown as below:



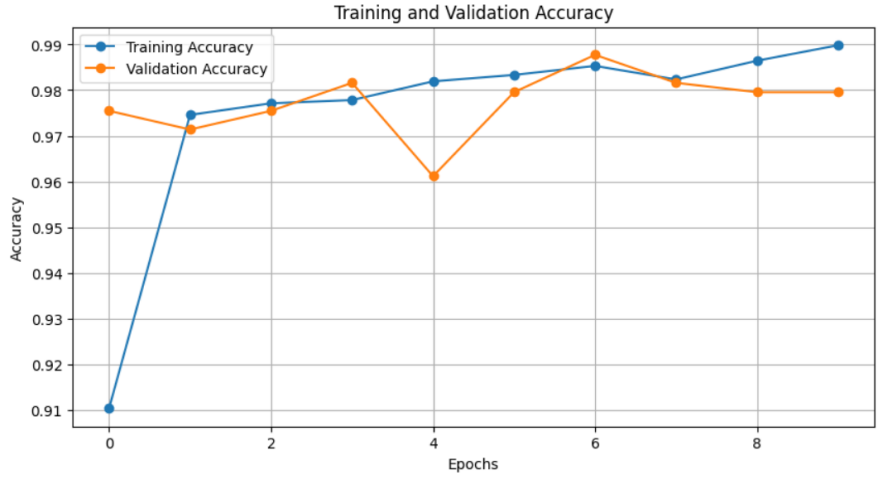
3: Long-Short Term Memory LSTM networks are a type of recurrent neural network designed to learn from sequences. The 13×786 matrices were treated as sequences of embeddings, enabling the model to capture temporal relationships and patterns between the features.

How it optimised our result?

LSTMs leverage the sequential nature of the data, allowing them to consider context across the embeddings, which can be vital for understanding how features interact over time.

Performance

The graph of the model's result is shown below:



4. Logistic Regression + Flattening (Best performance)

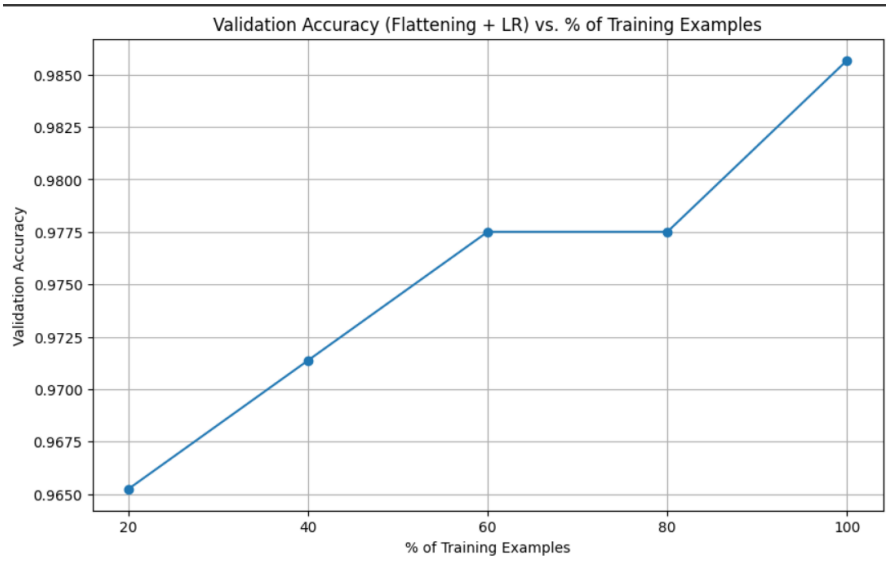
By flattening the 13 × 786 matrix into a single 10,188-dimensional vector, you retain all of the high-dimensional information from the deep feature embeddings. In this case, the flattened deep features might exhibit a structure that allows for effective linear separation between the classes, making Logistic Regression particularly well-suited for the task.

How it optimised our final result

Using this model, we were able to recieve the highest validation accuracy across all models, at approximately 98.7%. In this case, the flattened embeddings may have captured enough meaningful, linear patterns in the data for Logistic Regression to provide high accuracy. Thus, giving us higher accuracy than nonlinear models like Kernel SVMs and LSTM.

Performance

The graph of the performance for of this model is shown below:



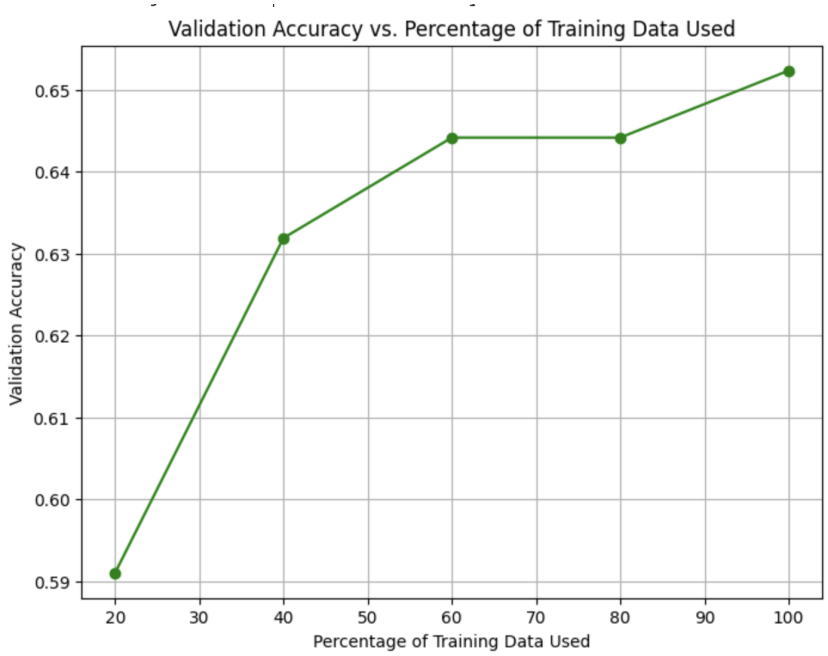
- **Text Sequence Dataset**

  1. Logistic Regression: We transformed sequence data into high-dimensional one-hot encoded vectors to retain structural information about character-level composition. This enabled the logistic regression model to handle categorical data efficiently by converting it into binary feature vectors.

  How it optimised our final result

  Using this approach, we achieved a notable validation accuracy of approximately 65.24%. Despite using one-hot encoded sequences, which preserve valuable categorical information, the model could not learn the temporal relationships inherent in the data. This limited its performance compared to more sophisticated models.

  Performance

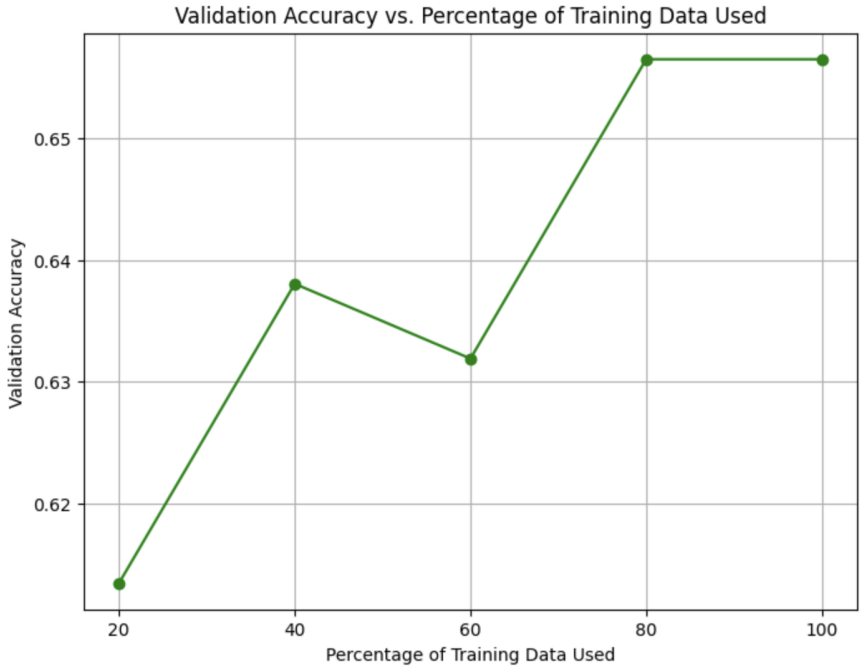


2. Support Vector Machine + Averaging: SVM was trained using various subsets of the dataset, and results were averaged to smooth out potential noise or overfitting. The linear kernel was used, which is appropriate for linearly separable data.

How it optimised our final result

This method achieved 65.69% accuracy, slightly better than Logistic Regression. Although SVM is more robust than Logistic Regression, the linear kernel still struggled with capturing the complexity and sequence dependencies in the dataset. The improvement in performance likely came from SVM's ability to find a better separating hyperplane in high-dimensional space, but it was still limited in handling time-related patterns.

Performance The graph of the performance for of this model is shown below:

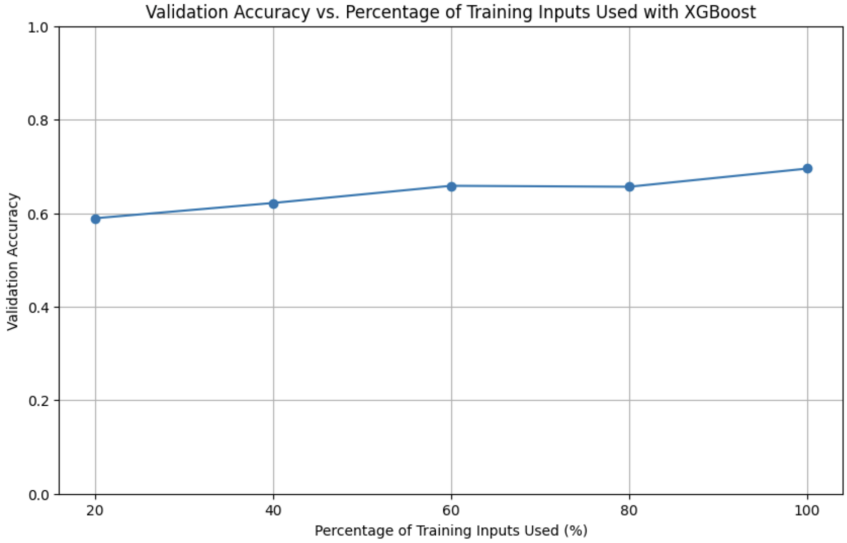


3. XG Boost: XGBoost is a powerful gradient boosting algorithm that builds a series of decision trees, with each tree correcting errors from the previous one. We used One-Hot Encoding for feature extraction.

How it optimised our final result

We optimized the model using early stopping, learning rate tuning, and a controlled number of trees to prevent over fitting. XGBoost outperformed the linear models, achieving 69.53% accuracy. This is because it captures non-linear relationships in the data and corrects errors through iterative boosting.

Performance The graph of the performance for of this model is shown below:

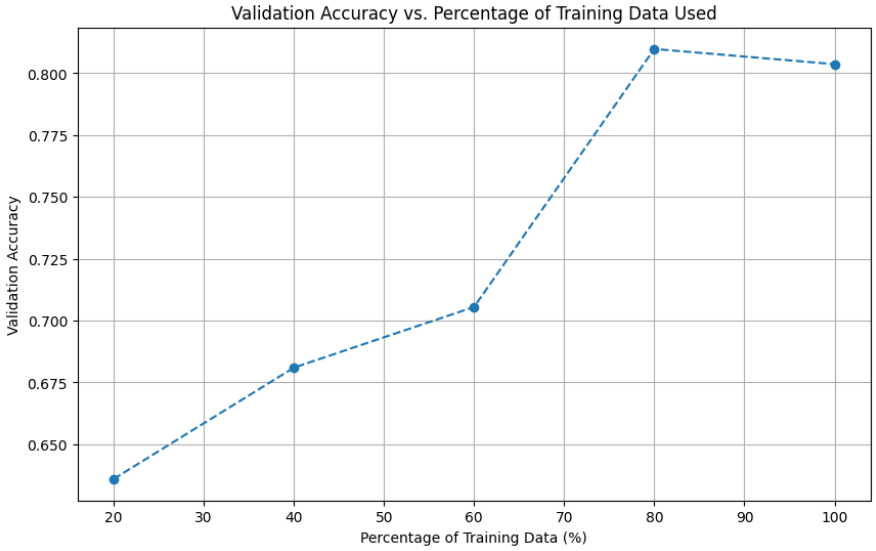


4. Long-Short Term Memory(Best performance): The LSTM model was trained using an embedding layer to convert the input sequences into dense vectors. It also included a Dropout layer to prevent overfitting and was optimized using the Adam optimizer with binary crossentropy as the loss function. We also used early stopping to halt training when the model's performance on the validation set stopped improving.

How it optimised our final result

LSTM achieved the highest accuracy, 80.98%, due to its ability to model temporal dependencies in sequential data. Unlike the other models, which treated sequences as unordered features, LSTM learned the time-related relationships between elements in the sequence. This gave it a significant edge in performance, making it the best-suited model for this task.

Performance The graph of the performance for of this model is shown below:



2.2 Task-2 : Training on Combined Dataset

In Task 2, we extend our analysis by exploring whether combining the three distinct datasets (Emoticon, Text-Sequence, and Deep-Feature datasets) can yield better performance than using each dataset individ-

ually. Each dataset offers a different representation of the same underlying raw data, providing diverse feature sets for the binary classification problem. Our objective is to investigate how effectively these varying feature representations can be merged to improve model accuracy.

Approach

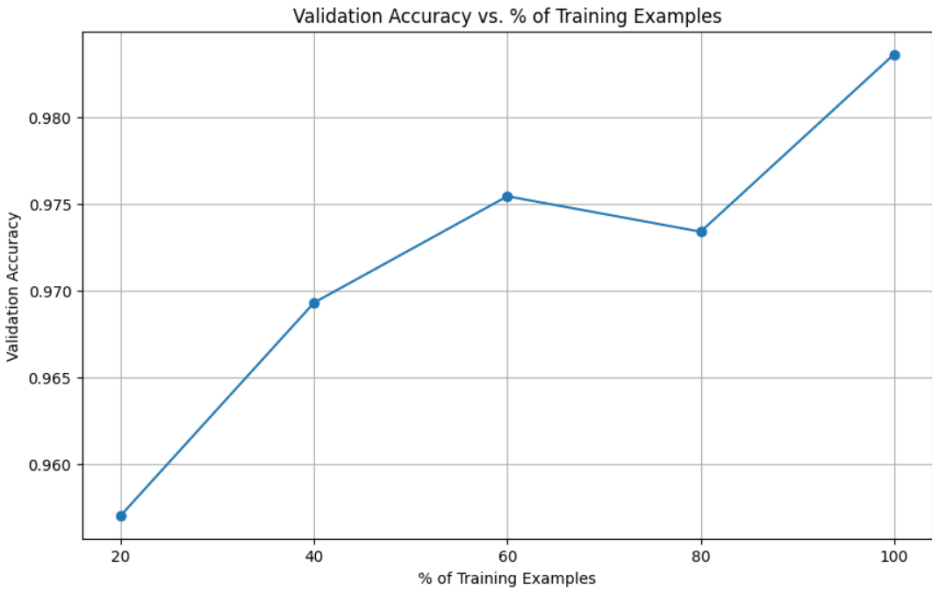
Firstly, the datasets are loaded and preprocessed. Since, deep features were represented as a 786x13 matrix, it is flattened into a 1D vector, to simplify the higher dimensionality. The emoticon-based features were represented as categorical character sequences. We utilized CountVectorizer from the sklearn library to convert these emoticon strings into a sparse matrix of features suitable for machine learning models. Similarly, the text-sequence data was a string of 50 digits. CountVectorizer was also applied to the text sequences to transform them into numerical feature vectors. The deep feature dataset provided 13 high-dimensional embeddings for each input. To integrate these embeddings, we flattened the 13x786 matrices into 1x10218 vectors using a simple flattening approach, which was later combined with the other feature representations. After vectorizing the emoticon and text data, and flattening the deep features, we combined all three sets into a unified feature matrix using horizontal stacking (hstack), creating a large sparse matrix with all feature representations. Given the high-dimensional nature of the combined dataset (especially after flattening the deep features), we applied Principal Component Analysis (PCA) to reduce the feature space. This step was essential to:

- Avoid overfitting, as high-dimensional feature spaces can lead to models that perform well on training data but fail to generalize on unseen data.
- Enhance computational efficiency by limiting the number of parameters that the model needs to learn

By reducing the feature dimensions to 500 components, we balanced the trade-off between model complexity and performance.

Model Training

For the combined dataset, Logistic Regression was selected as the baseline model due to its simplicity



and interpretability. We applied logistic regression to the PCA-reduced dataset, training the model using the combined features from all three datasets. Training data percentages of 20%, 40%, 60%, 80%, and 100% were used, and for each percentage, the logistic regression model was trained and evaluated. The performance of the model on the reduced dataset is given in the plot shown.

Did combining the datasets improve our accuracy?

The accuracy achieved after combining the data set is more or less equal to the accuracy achieved on the second dataset, where we first flattened the dataset and then trained using logistic regression same as we did after combining the datasets(used LR).

.