CS771 MINI PROJECT-1

Riyanshu Kumar
Roll no: 220904

Atharva Singh
Roll no: 220251

Kawaljeet Singh
Roll no: 220514

Nikhil Jain
Roll no: 220709

Binary Classification ML Models

1 Introduction

In this project, we have to build machine learning models for binary classification tasks for multiple datasets. Each dataset is generated from the same raw data but differs in the feature representation of the input. The objective is to analyze and train models for each dataset individually and assess the performance based on accuracy and the amount of training data used. Also, we have to check whether the combination of all the datasets can enhance the model's performance by leveraging the varying feature representations.

2 Task 1: Approaches and the Best model for different datasets

2.1 Emoticons as Features Dataset

2.1.1 Feature Extraction

The features of each input are given in the form of 13 emoticons. Each input emoticon string is broken down into individual characters, which are then mapped to integers. This converts the textual data into a numerical format suitable for processing by the model. The input sequences are passed through an embedding layer, which transforms the integer-encoded characters into dense vectors of a fixed dimension (32 in this case). This layer learns the character representations during training.

2.1.2 Models Used

I) **Logistic Regression:** We got an accuracy of **97.17%**.
II) **Decision Tree:** We got an accuracy of **86.09%**.
III) **SVM:** We got an accuracy of **96.52%**.
IV) **KNN:** We got an accuracy of **94.27%**.
V) **Dense neural network:** We got an accuracy of **98.15%**.

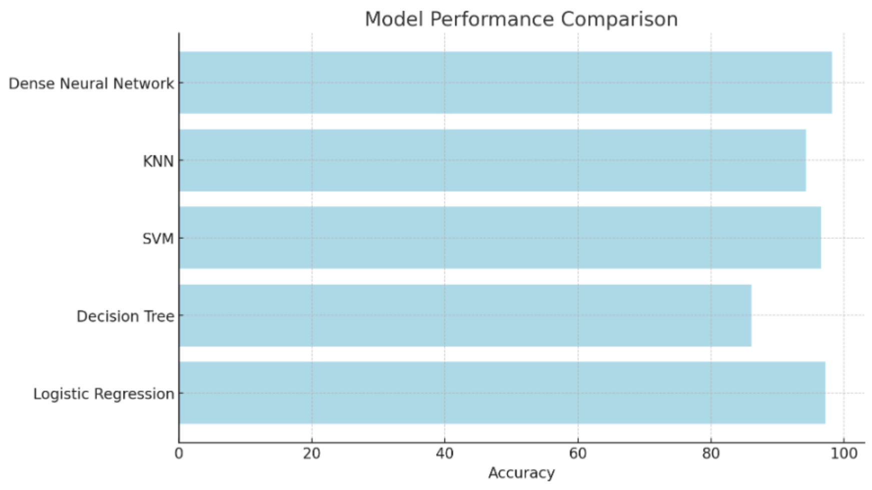

Figure 1: Accuracy comparison of different models used.

2.1.3 Best Model: Dense Neural Network

Dense Neural Network (DNN) performs better than models like logistic regression, decision trees, and SVMs because it Learns complex features through embeddings and layers, unlike traditional models. It captures non-linear patterns, which logistic regression and decision trees struggle with. Handles high-dimensional data better by learning hierarchical representations.

The total number of trainable parameters is 7297, and we got an accuracy of **98.15%** on this model.

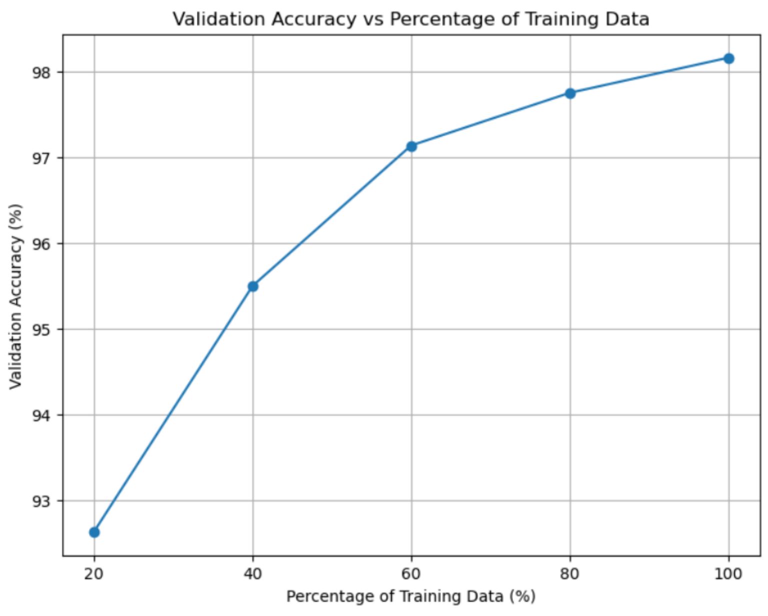

Figure 2: Accuracy of Dense Neural Network model for different percentage of training dataset used.

2.2 Deep Features Dataset

2.2.1 Feature Extraction

The features of each input are extracted using a simple deep neural network. This resulted in each input being represented using a 13 x 786 matrix which basically consists of 786-dimensional embeddings of each of the 13 emoticon features.

We flattened the 13 x 786 matrix into a single vector of size 10218 (13 * 786) to be used as input and used traditional machine learning models like SVM, Random Forest, etc. Also, Before feeding the input to the models, the flattened vectors are standardized using a StandardScaler.

This step ensures that each feature contributes equally, preventing features with larger ranges from dominating.

2.2.2 Models used

I) **Logistic Regression:** The loss function used is Binary cross-entropy, and we got an accuracy of **98.16%**.

II) **Decision Tree and Random Forest:** These models optimize based on information gain or Gini impurity at each node split to improve classification accuracy. No traditional loss function is used. We got an accuracy of **96.73%** on the Decision Tree and **96.36%** on the Random Forest.

III) **KNN:** We got an accuracy of **92.84%** on this.

IV) **SVM:** We got an accuracy of **98.77%**.
We tried all these models in Google Colab, so I am attaching the link to that. Click here to see models for this dataset.

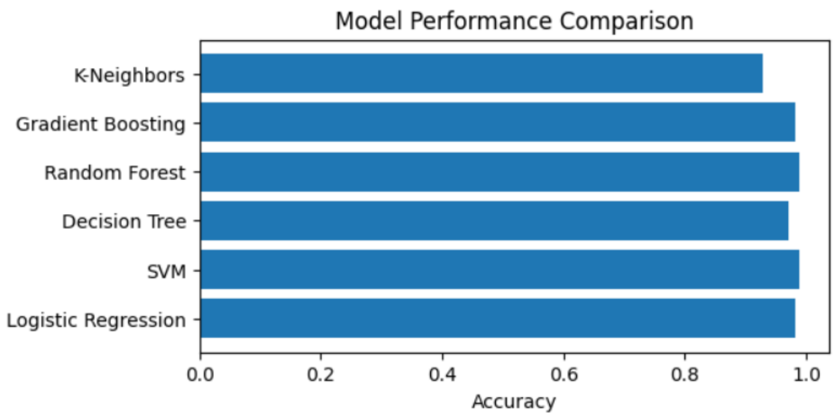

Figure 3: Accuracy comparison of different models used.

2.2.3 Details of the Best Performing Model: SVM

We have used the RBF kernel in SVM. The RBF kernel in SVM is excellent at handling non-linear patterns in the high-dimensional 13x786 matrix. It can map the data into a higher-dimensional space where a linear separation is possible, which might not be achievable by simpler models like Logistic Regression or Decision Trees.
We got an accuracy of **98.77%** on this model. The number of trainable parameters used for this model (SVM) is zero.

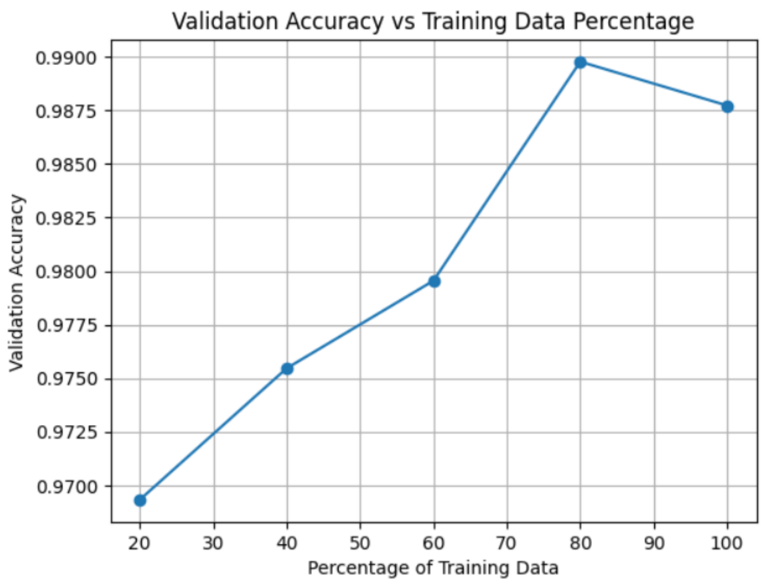

Figure 4: Accuracy of SVM model for different percent of training dataset used.

2.3 Text Sequence Dataset

2.3.1 Feature Extraction

The features of each input are given in form of a length 50 string consisting of 50 digits. We extracted features from input strings of digits, where each digit acts as a categorical feature representing properties of the emoticon. This process converts the raw string data into a structured format (a matrix of integers). The embedding layer then transforms these integer representations into dense embeddings, capturing semantic relationships between digits. This transformation allows the model to learn complex patterns in the data more effectively than using raw digits.

2.3.2 Models Used

I) **Random Forest:** We used TF-IDF (Term Frequency-Inverse Document Frequency) which converts the text data into a numerical matrix where each row represents a document, and each column represents a word or combination of words (n-gram). then, we applied a random forest model to this and got an accuracy of **50.72%**.

II) **LSTM with embedding:** The feature extraction involves tokenizing input strings and converting them to integer sequences and then embedding layer transforms these integer representation to dense vectors, then an LSTM layer with 32 units is applied. Finally, we applied a dense layer of sigmoid function for binary classification. The loss function was used is binary cross-entropy and we got an accuracy of **69.53%**.

III) **CNN + LSTM:** First, Embedding converts input sequences to dense vectors of size(dim = 64). then 1D convolutional layer with 16 filters and a kernel of size 3 is applied followed by a Maxpooling1D layer is included with a pool size of 2 which downsamples the output from the convolutional layer, finally LSTM layer with 32 units is applied followed by a dense layer with a softmax activation function which outputs class probabilities. We got an accuracy of **82.21%** but the total number of trainaible parameters was 10,066 which is very large.

iV) **GRU:** An embedding layer is used to convert the input sequences into dense vector representations.Two GRU layers are added. GRUs are a type of recurrent neural network (RNN) that help capture temporal patterns in sequences. The first GRU layer has 16 units and returns sequences, while the second GRU layer has 32 units but outputs the final hidden state. A Dropout layer is added between the GRU layers with a dropout rate of 0.4, which helps prevent overfitting by randomly deactivating 40 of the neurons during training. A Dense layer with a sigmoid activation function is used to output a single value, making it suitable for binary classification. We got an accuracy of **86.09%**.

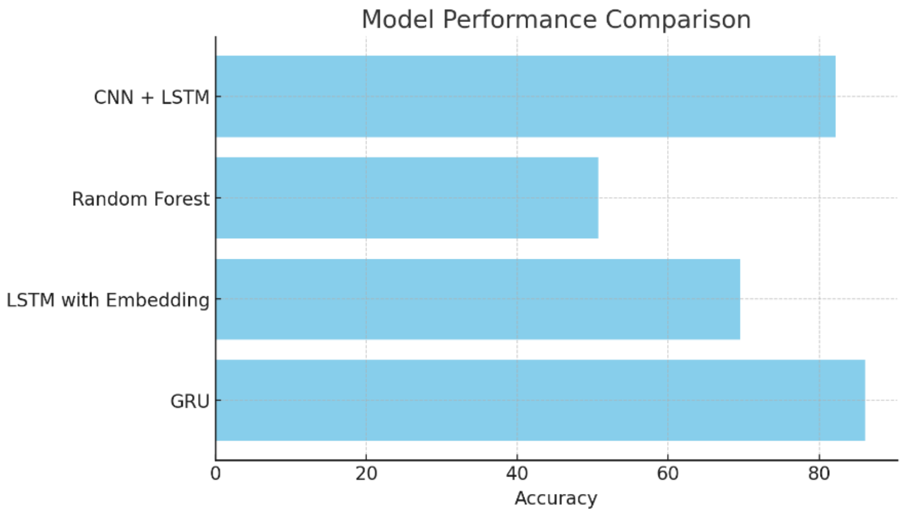

Figure 5: Accuracy comparison of different models used.

2.3.3 Details on Best model: GRU

The input is a sequence of 50 digits, and using a GRU (a type of RNN) allows the model to capture dependencies between digits in the sequence. This is particularly useful for learning temporal or order-dependent patterns. GRU layers are computationally efficient compared to traditional RNNs while still being able to capture long-term dependencies. They have fewer parameters compared to LSTMs but can still model sequential data effectively. Because of this we are able to use 2 GRU's without exceeding the parameter limit helping us to improve the accuracy. Total number of trainable parameters used is 6045. Accuracy of the model is **86.09%**.

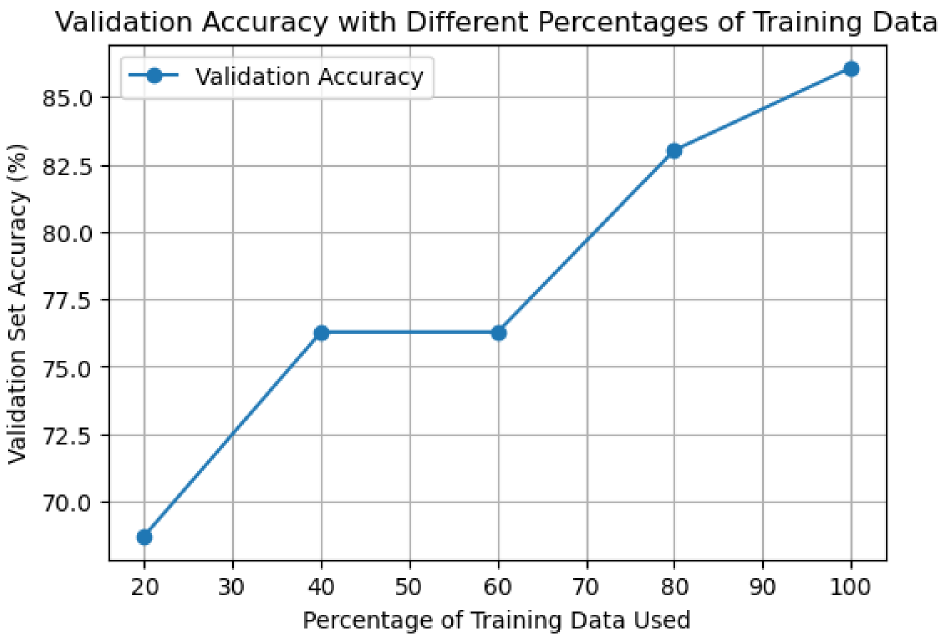

Figure 6: Accuracy of GRU model for different percentage of training dataset used.

3 Task 2: Combined Dataset

3.1 Feature Extraction

Emoticon Dataset:
OneHotEncoding: Categorical emoticons are transformed into a binary format, where each category is represented as a separate column. For example, if there are three emoticons, they will be represented as three binary columns indicating the presence or absence of each emoticon.

Deep Features Dataset:
Flattening and Scaling: The deep features, typically multi-dimensional, are reshaped into a 2D array. This is followed by standard scaling, which standardizes the features to have a mean of 0 and a standard deviation of 1, ensuring that they contribute equally to model performance.

Text Sequence Dataset:
TF-IDF Vectorization: This converts the text data into a numerical format using Term Frequency-Inverse Document Frequency (TF-IDF), focusing on character-level n-grams (1-gram and 2-gram). This highlights the importance of each character or character pair relative to the entire text corpus.

Concatenation: All processed features from the emoticon, deep features, and text sequence datasets are concatenated into a single feature matrix.

3.2 Models Used

I) **KNN:** We got an accuracy of **93.25%**.

II) **Logistic Regression:** We got an accuracy of **97.96%**.

III)**SVM:** We got an accuracy of **98.77%**.

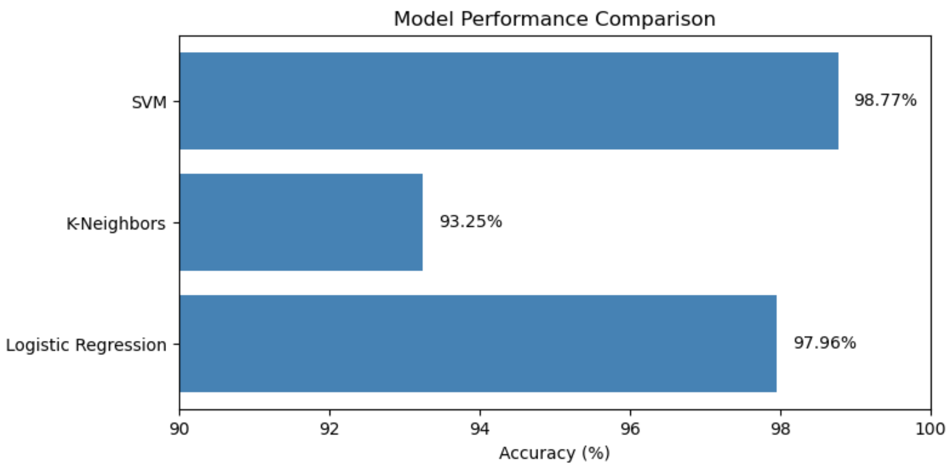

Figure 7: Accuracy comparison of different models used.

3.3 Best model: SVM

We got an accuracy of **98.77%** on SVM. The number of support vectors in the SVM model is 1841.

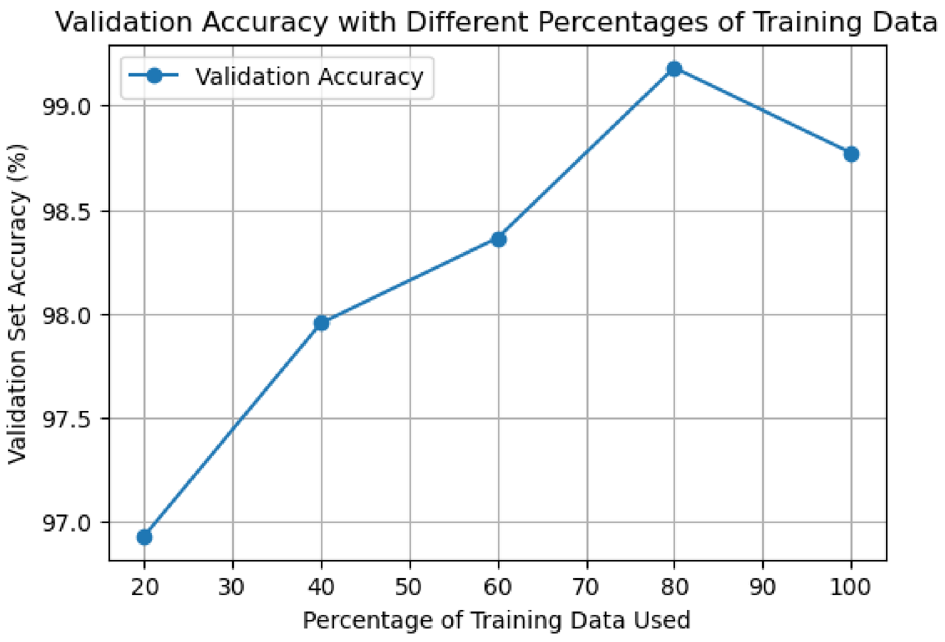

Figure 8: Accuracy of SVM for different percentage of training dataset used.

3.4 Final analysis of all models

The combined model performs better than the individual dataset model. It has better accuracy than the emoticon and text sequence models and is almost similar in accuracy to the feature dataset model.

3.5 Acknowledgement

All the ML models that we have used, like logistic regression, SVM, Decision Tree, Random Forest, and KNN, are inbuilt models in the sklearn library.