CS771 : Intro. to ML (project)

Sankul
Roll No. 220965

Bhavishya Gupta
Roll No. 220295

Harshit Gupta
Roll No. 220439

Gautam Kumar
Roll No. 220407

Ayush Patel
Roll No. 220269

1 Task 1

This report summarizes the findings from Task 1 of the CS771 Mini-Project. In this task, we worked on a binary classification problem using Datasets from Task 1, which consists of emoticon features, deep features and text sequences features. Our goal was to identify the best-performing model that could generalize well using the least amount of training data.

1.1 Emoticons as Features Dataset

Dataset Description: Dataset 1 consists of training examples, with each input being represented by 13 emoticons. The task is to classify each input as one of two classes (0 or 1). The dataset was provided in .csv format.

Model Tried: We experimented with the following machine learning models on Dataset 1:

- **Model 1**: LSTM
- **Model 2**: SVM
- **Model 3**: Random Forest

We chose these models based on their suitability for binary classification and the constraints of the task e.g. number of parameters, efficiency.

Feature engineering/preprocessing applied: We used a tokenizer and converted the sequence of emoticons into a sequence of integers.

Model Selected: LSTM

Reasoning for LSTM Selection: We chose to use a Long Short-Term Memory (LSTM) network for this dataset due to its ability to effectively capture sequential dependencies, even in non-time-series data like emoticons. The 13 emoticons, though individually representing features, may contain implicit patterns or dependencies that contribute to the classification task.

LSTMs are well-suited to model such potential dependencies by preserving relevant information across a sequence of inputs, ensuring that each emoticon's contribution to the binary label (0/1) is considered in the context of the entire input sequence.

Training and Accuracy Analysis: To evaluate the performance of our models, we trained each model using varying amounts of the training data (20%, 40%, 60%, 80%, 100%) and recorded their accuracy on the validation set.

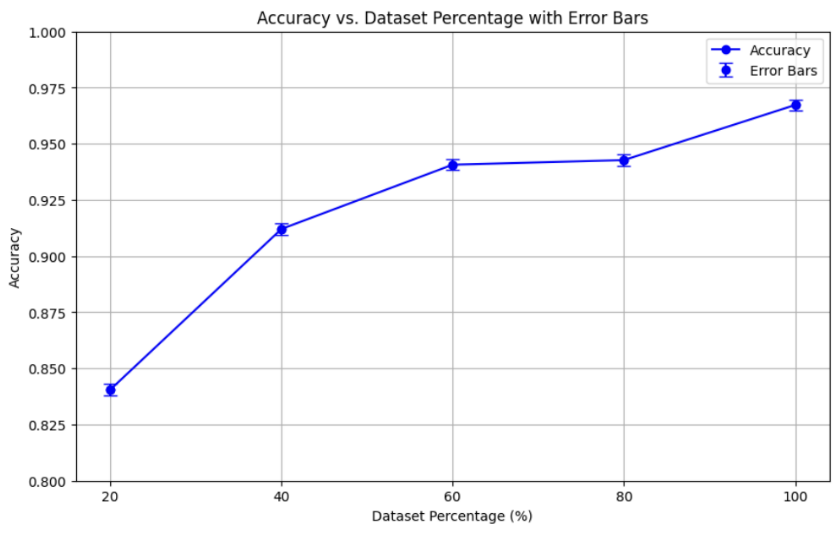

Figure 1: Validation Accuracy vs. Training Data Size for Dataset 1

Conclusion: In Task 1, Dataset 1, we experimented with several binary classification models and identified **LSTM** as the best model for this dataset. The model achieved a validation accuracy of 97% and was able to generalize well.

1.2 Deep Features Dataset

Dataset Description: This dataset consists of inputs represented as 13 $\times$ 786 matrices. Each input corresponds to 13 emoticons, where each emoticon is transformed into a 786-dimensional embedding using a deep neural network. These embeddings represent the features of each input.

Goal: The task is to classify each input as either class 0 or 1.

Format: The dataset is provided as an `.npz` file, with the following structure:

- **features**: A 13 $\times$ 786 matrix representing the 786-dimensional embeddings of the 13 emoticons.
- **label**: A binary value (0 or 1) indicating the class of the input.

Model Tried: We experimented with the following machine learning models on Dataset 2:

- **Model 1**: XGBoost **Why**? Efficient for structured data, provides feature importance, and has built-in regularization to prevent overfitting.
- **Model 2**: CatBoost **Why**? Handles categorical features automatically, reducing preprocessing effort, and is robust against overfitting.
- **Model 3**: Random Forest **Why**? Uses ensemble learning for improved accuracy and robustness to overfitting, while providing feature importance insights.
- **Model 4**: SVM

We chose these models based on their suitability for binary classification and the constraints of the task (e.g., number of parameters, efficiency).

Feature engineering/preprocessing applied: `Flattening`: The input data was reshaped into a 2D matrix to ensure compatibility with the model.
`Standardization`: The features were standardized using `StandardScaler`, ensuring a mean of 0 and a standard deviation of 1 for both the training and validation sets.

Model Selected: SVM

Reasoning for SVM Selection:

1. **High Dimensionality Handling**: SVM excels in high-dimensional spaces, effectively managing the many features from one-hot encoding.
2. **Non-Linearity**: The RBF kernel enables SVM to create non-linear decision boundaries, capturing complex patterns in the data.

3. **Sensitivity to Margin**: SVM focuses on support vectors, enhancing generalization on unseen data.
4. **Versatile with Sparse Data**: SVM handles sparse representations well, learning patterns from one-hot encoded features.

Training and Accuracy Analysis: To evaluate the performance of our models, we trained each model using varying amounts of the training data (20%, 40%, 60%, 80%, 100%) and recorded their accuracy on the validation set.

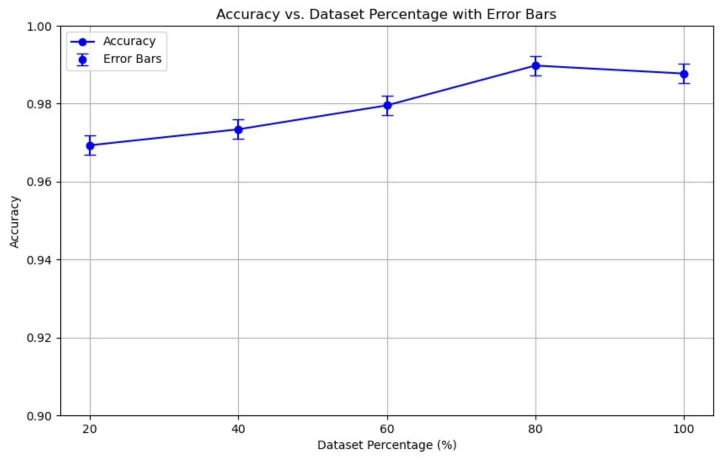

Figure 2: Validation Accuracy vs. Training Data Size for Dataset 1

Conclusion: In Task 1, Dataset 2, we experimented with several binary classification models and identified **XGBoost** as the best model for this dataset. The model achieved a validation accuracy of 98.8% and was able to generalize well.

1.3 Text Sequence Dataset

Dataset Description: The features of each input are provided as a string of length 50, consisting of digits (0-9). Each string corresponds to an input, and the task is to classify it into one of two classes (0 or 1). While this dataset is challenging, please note that the string-based representation of each input is related to the emoticon and deep feature representations of the same input in the other datasets.

Goal: The task is to classify each input based on the 50-digit sequence into either class 0 or 1.

Format: The dataset is provided in CSV format, containing the following columns:

- **input_str**: A length 50 string consisting of digits (e.g., "0241812141226549266461...").
- **label_str**: A binary label (0 or 1) indicating the class of the input.

Model Tried: We experimented with the following machine learning models on Dataset 3:

- **Model 1**: XGBoost **Why**? Efficient for structured data, provides feature importance, and has built-in regularization to prevent overfitting.
- **Model 2**: CatBoost **Why**? Handles categorical features automatically, reducing preprocessing effort, and is robust against overfitting.
- **Model 3**: Random Forest **Why**? Uses ensemble learning for improved accuracy and robustness to overfitting, while providing feature importance insights.
- **Model 4**: SVM

Feature engineering/preprocessing applied: `One-Hot Encoding` The data (*df_train_one_hot*, *df_val_one_hot*, *df_test_one_hot*) is preprocessed using one-hot encoding to convert categorical variables into a binary format for better model interpretation.
In the preprocessing step, one-hot encoding was applied to convert each 15-character string in the

`result_str10` column into a 150-dimensional binary vector. Each digit (0-9) at positions 0-14 was represented as a distinct feature. This transformation prepared the sequential data for machine learning models by encoding digit occurrences in a structured binary format, with only the first 491 rows of the dataset processed.

Our code systematically identifies and removes common patterns within the text data. Initially, the longest common substring across all input strings is extracted and removed, leading to a reduced representation of the text. This process is repeated several times, allowing for the extraction of further distinct features by progressively refining the strings. This approach helps in reducing redundancy and focuses on the unique aspects of the text, which can enhance the performance of subsequent machine learning models.

Model Selected: SVM

Reasoning for SVM Selection:

1. **High Dimensionality Handling**: SVM excels in high-dimensional spaces, effectively managing the many features from one-hot encoding.
2. **Non-Linearity**: The RBF kernel enables SVM to create non-linear decision boundaries, capturing complex patterns in the data.
3. **Sensitivity to Margin**: SVM focuses on support vectors, enhancing generalization on unseen data.
4. **Versatile with Sparse Data**: SVM handles sparse representations well, learning patterns from one-hot encoded features.

Training and Accuracy Analysis: To evaluate the performance of our models, we trained each model using varying amounts of the training data (20%, 40%, 60%, 80%, 100%) and recorded their accuracy on the validation set.

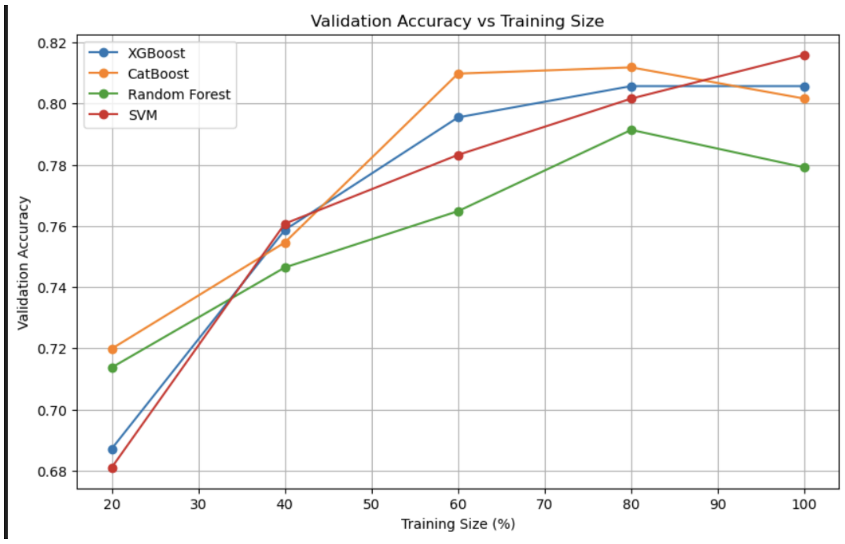

Figure 3: Validation Accuracy vs. Training Data Size for Dataset 3

Conclusion: In Task 1, Dataset 3, we experimented with several classification models and identified **SVM** as the best model for this dataset. The model achieved a validation accuracy of 81.6% and was able to generalize well.

2 Task 2

2.1 Combined DataSets

Dataset Description: After combining all three data sets 1)Emoticons as Features Dataset 2)Deep Features Dataset 3)Text Sequence Dataset . The dataset was provided in .csv format.

Model Tried: We experimented with the following machine learning models on Dataset 1:

- **Model 1**: XGBoost **Why**? Efficient for structured data, provides feature importance, and has built-in regularization to prevent overfitting.
- **Model 2**: Logictics Regression **Why**? effective for **binary classification** tasks, modeling class probabilities using a **logistic (sigmoid) function**. It is ideal for categorical outputs (e.g., 0 or 1) and handles **linearly separable data** efficiently. Additionally, it offers **L1/L2 regularization** to prevent overfitting, especially useful in high-dimensional datasets. Its simplicity and interpretability make it a popular choice for classification problems.
- **Model 3**: Random Forest **Why**? Uses ensemble learning for improved accuracy and robustness to overfitting, while providing feature importance insights.
- **Model 4**: Ridge Classifier **Why**? ensemble learning for improved accuracy and robustness to overfitting, while providing feature importance insights.
- **Model 5**: SVM

Feature engineering/preprocessing applied: Here we preprocesses and extracts features from emoticon and text datasets for a machine learning task. Initially, it converts emoticon sequences from the training, testing, and validation datasets into lists and then encodes these sequences using **LabelEncoder**, handling unknown emoticons with a designated token. The emoticon data is transformed into one-hot encoded vectors, while the text sequences are formatted to fixed lengths and converted into lists. Finally, the emoticon, text, and additional features are concatenated into combined feature matrices, which are then reduced in **dimensionality** using **PCA** to extract the most relevant features for model training and evaluation.

Model Selected: SVM

Training and Accuracy Analysis: To evaluate the performance of our models, we trained each model using varying amounts of the training data (20%, 40%, 60%, 80%, 100%) and recorded their accuracy on the validation set.

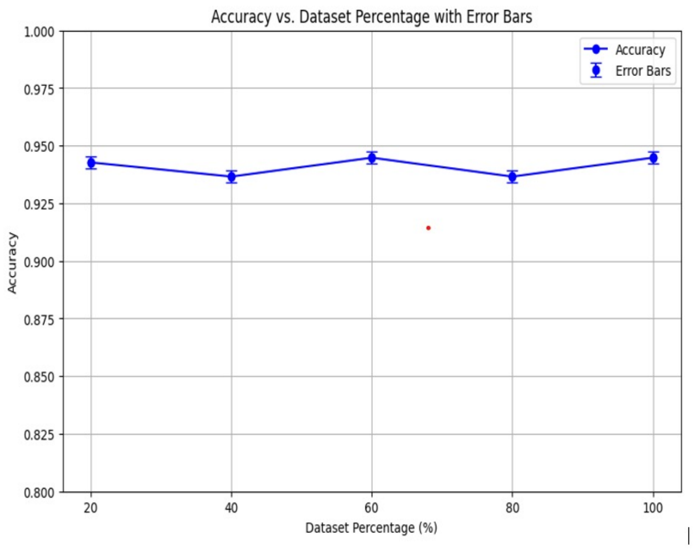

Figure 4: Validation Accuracy vs. Training Data Size for Combined Datasets

Final Conclusion: In Task 2 on Combined Datasets, we experimented with several binary classification models and identified **SVM** as the best model for this dataset. The model achieved a validation accuracy of 94.47% and was able to generalize well.

Final Remarks

The dataset reveals itself to be largely parametrized by the "non-noise" emojis and their positional occurrences, presenting an intriguing underlying structure. Our ability to achieve high accuracies by flattening the deep feature matrices points to two essential inferences. First, the dataset's complexity is more profound than it may initially appear when viewed solely through the lens of the emoticon representation. Second, the combination of position-emoji pairs serves as a deceptively simple, yet effective, abstraction of the underlying data.

Furthermore, when we flatten the deep feature matrix, the entire set of 768 deep features can be "fine-tuned," thereby allowing the model to capture more intricate patterns, which results in significantly higher accuracies than those achieved using only the emoji dataset. This enhancement demonstrates the potential of the deep features to encapsulate finer-grained information and adds depth to the model's predictions.

Interestingly, we observed that the test set introduced emojis that were absent from the training and validation sets. Since these unseen emojis did not contribute to our known feature space, they were rightfully ignored, ensuring no adverse impact on the model's performance.

Ultimately, the primary takeaway from our analysis is that the provided dataset exhibits strong signs of linear separability. This is evidenced by the success of linear classifiers, particularly the Support Vector Machine (SVM), which achieved exceptional accuracy on the validation sets. The performance of SVM highlights that the dataset's structure lends itself well to separation via hyperplanes, further confirming its linear separability.