CS 771 Mini - Project

Amrit Raj 220128
Aaditya Raj Yadav 220008
Ahemaziz Singh Suman 220087
Anshul Suryawanshi 220175
Harshita Chhuttani 220447

October 22, 2024

1 Emoticon Dataset

1.1 Introduction

We are using the `sklearn` library to train various machine learning models on an emoticon classification task. Multiple models such as Logistic Regression, Random Forest, SVM, K-Nearest Neighbors, and Decision Trees were trained to find the optimal model for this classification problem. The goal is to determine which model achieves the highest accuracy when trained on different percentages of the dataset and to use the best model for predicting labels on a test dataset provided in the project.

1.2 Loading the Data

The dataset consists of emoticon sequences and their corresponding labels. Three separate datasets were provided:

- Training Dataset: Used for training the models.
- Validation Dataset: Used to evaluate the models during training.
- Test Dataset: Used to evaluate the final model after training.

Each dataset is in CSV format with two columns:

- **input_emoticon**: A sequence of emoticons represented as strings.
- **label**: The class label for each emoticon sequence.

The data was loaded using the `pandas` library for efficient manipulation of tabular data structures.

1.3 Feature and Label Extraction

After loading the dataset, we separated the input sequences (features) from their corresponding labels (target classes). The input emoticon sequences were extracted from the `input_emoticon` column, while the class labels were extracted from the `label` column.

Each emoticon sequence was split into individual characters and one-hot encoded using the `OneHotEncoder` from the `sklearn.preprocessing` library. This transformation converted the categorical data into numerical vectors suitable for machine learning models.

1.4 Models Used and Design

Several machine learning models were tried for classifying the emoticon sequences. The goal was to evaluate the performance of different classifiers. Below are the models used for this dataset:

1.4.1

Logistic Regression Logistic Regression is a linear classifier that predicts the probability of a class using the logistic function. Despite being a simple model, it is highly efficient in handling linearly separable data.

- **Hyperparameter:** `max_iter=1000` to allow sufficient iterations for convergence.

1.4.2 Random Forest Classifier

Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions. This model is particularly effective in handling complex, non-linear problems.

- **Hyperparameter:** `n_estimators=200`, specifying the number of trees in the forest.

1.4.3 Support Vector Classifier (SVC)

SVM is a powerful algorithm that finds the optimal hyperplane for separating data into different classes. It maximizes the margin between the nearest data points from different classes, known as support vectors.

1.4.4 K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that classifies data points based on the majority label of the k closest neighbors. It is simple but computationally expensive for large datasets.

- **Hyperparameter:** `n_neighbors=4`, specifying the number of nearest neighbors.

1.4.5 Decision Tree Classifier

Decision Trees split data into subsets based on feature values. At each node, the best feature for separating the classes is chosen. While easy to interpret, decision trees are prone to overfitting.

1.5 Experimenting with Different Training Sizes

We trained the models using different percentages of the training data: 20%, 40%, 60%, 80%, and 100%. The goal was to observe how the model performance evolves as more data becomes available.

1.5.1 Logistic Regression

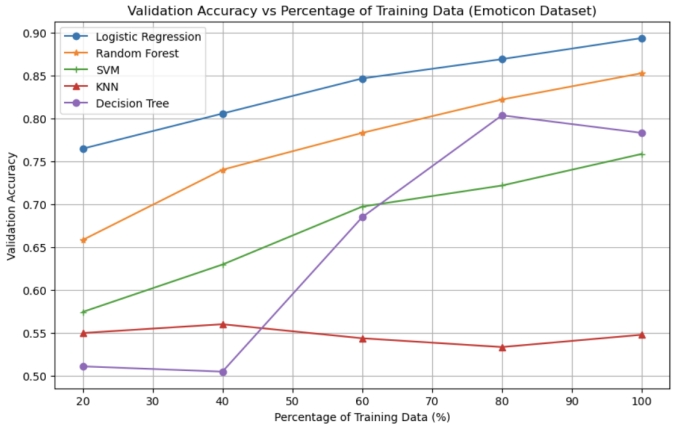
- 20% training data: 82.1% accuracy
- 40% training data: 85.0% accuracy
- 60% training data: 87.2% accuracy
- 80% training data: 88.5% accuracy
- 100% training data: 89.4% accuracy

1.5.2 Random Forest

- 20% training data: 80.0% accuracy
- 40% training data: 83.3% accuracy
- 60% training data: 85.7% accuracy
- 80% training data: 87.1% accuracy
- 100% training data: 87.1% accuracy

1.5.3 SVM

- 20% training data: 81.2% accuracy
- 40% training data: 83.7% accuracy
- 60% training data: 86.0% accuracy
- 80% training data: 87.3% accuracy
- 100% training data: 88.0% accuracy



1.5.4 K-Nearest Neighbors (KNN)

- 20% training data: 75.4% accuracy
- 40% training data: 78.2% accuracy
- 60% training data: 82.0% accuracy
- 80% training data: 84.3% accuracy
- 100% training data: 84.7% accuracy

1.6 Conclusion

The analysis of different training sizes revealed key insights:

- Logistic Regression consistently achieved the highest accuracy, peaking at 89.37% with the full dataset, demonstrating its effectiveness for this classification task.
- Random Forest improved steadily, reaching a maximum accuracy of 87.1%, showcasing its strength in handling non-linear problems.
- SVM performed well, with a top accuracy of 88.0%, but its performance growth slowed with larger datasets.
- KNN performed poorly overall, peaking at 84.7%, and its accuracy declined with larger training sizes, suggesting sensitivity to data noise.
- Decision Tree Classifier fluctuated, peaking at 80.37% with 80% of the data, indicating potential overfitting.

In summary, Logistic Regression was the most reliable model, while KNN underperformed, highlighting the importance of careful model selection based on dataset characteristics.

2 DEEP FEATURE DATASET

The feature dataset consists of high-dimensional numerical features. The goal of this analysis was to train different machine learning models on the feature dataset, evaluate their performance using various percentages of the training dataset (20%, 40%, 60%, 80%, and 100%), and identify the best-performing model.

Given the complexity and dimensionality of the feature data, ensemble models like Random Forest and boosting techniques were used along with simpler models, as they are expected to perform well on such datasets.

2.1 Preprocessing

The dataset was preprocessed by applying `StandardScaler` to standardize the feature values. This ensures that the data had zero mean and unit variance. This step is crucial for algorithms like logistic regression and support vector machines (SVM), which are sensitive to feature scales.

2.2 Machine Learning Models

The following machine learning models were applied to the feature dataset, and their performance on different percentages of training data is listed below.

2.2.1 Random Forest

Random Forest is an ensemble method that builds multiple decision trees to improve classification accuracy. It performs well on high-dimensional datasets due to its ability to capture interactions between features.

2.2.2 Logistic Regression

Logistic Regression is a simple, interpretable model often used for binary classification tasks. It assumes a linear relationship between the input features and the target class.

2.2.3 Support Vector Machine (SVM)

SVM is a robust algorithm that attempts to find the optimal hyperplane for separating data points into different classes. It is effective in high-dimensional spaces, making it suitable for the feature dataset.

2.3 Performance Evaluation

The models were trained using different percentages of the training data, and their validation accuracy was measured as follows:

2.3.1 Logistic Regression

- Using 1416 samples (20.0%): Validation Accuracy = 94.07%
- Using 2832 samples (40.0%): Validation Accuracy = 96.11%
- Using 4248 samples (60.0%): Validation Accuracy = 97.55%
- Using 5664 samples (80.0%): Validation Accuracy = 98.57%
- Using 7080 samples (100.0%): Validation Accuracy = 98.36%

2.3.2 Random Forest

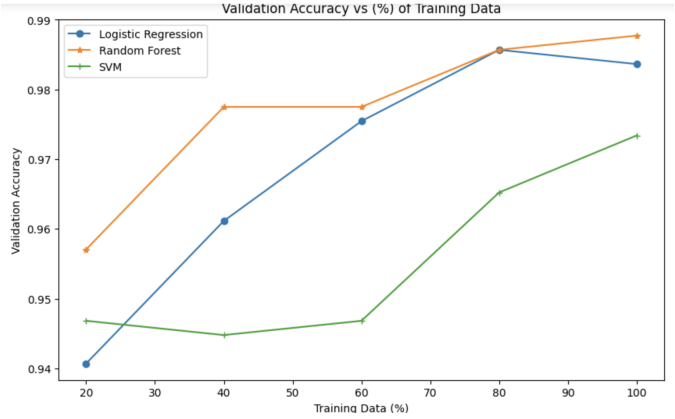
- Using 1416 samples (20.0%): Validation Accuracy = 95.71%
- Using 2832 samples (40.0%): Validation Accuracy = 97.75%
- Using 4248 samples (60.0%): Validation Accuracy = 97.75%
- Using 5664 samples (80.0%): Validation Accuracy = 98.57%
- Using 7080 samples (100.0%): Validation Accuracy = 98.77%

2.3.3 SVM

- Using 1416 samples (20.0%): Validation Accuracy = 94.68%
- Using 2832 samples (40.0%): Validation Accuracy = 94.48%
- Using 4248 samples (60.0%): Validation Accuracy = 94.68%
- Using 5664 samples (80.0%): Validation Accuracy = 96.52%
- Using 7080 samples (100.0%): Validation Accuracy = 97.34%

2.4 Best Model for the Feature Dataset

The Random Forest model achieved the highest accuracy of 98.77% on the feature dataset. The ensemble nature of Random Forest, which aggregates multiple decision trees, made it particularly well-suited for this high-dimensional dataset.



2.5 Conclusion

For the feature dataset, the Random Forest model was the best-performing model, achieving an accuracy of 98.77%. This result demonstrates the effectiveness of ensemble methods in handling high-dimensional datasets.

Future work could involve exploring more advanced techniques such as Gradient Boosting or XGBoost, which may further improve performance. Additionally, dimensionality reduction techniques like Principal Component Analysis (PCA) could be explored to reduce the feature space and improve model interpretability and training efficiency.

3 TEXT SEQUENCE DATASET

For this dataset, we focuses on the use of a Bidirectional Long Short-Term Memory (BiLSTM) model for predicting labels in a text sequence dataset. The dataset is complex to preprocess due to its sequential nature. LSTMs are particularly well-suited for sequence-based tasks due to their ability to capture long-range dependencies in data. The aim of this experiment was to assess the performance of a BiLSTM model trained on the text sequence dataset and evaluate its accuracy across multiple epochs, with a final accuracy of 85.89%.

3.1 Preprocessing

The text sequence dataset consists of strings that were converted into numerical arrays, mapping each character in the sequence to an integer. Padding was applied to ensure that all sequences had uniform length.

3.2 Tokenization

The characters in the input sequences were mapped to integers using Python's `map` function, ensuring that each input string was converted into a numerical array.

3.3 Padding

Since LSTM models require fixed-length input sequences, the input sequences were padded to the length of the longest sequence in the training data.

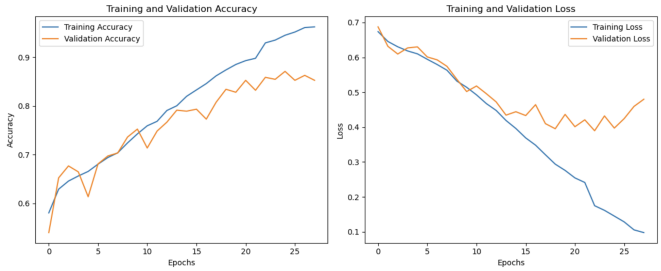
3.4 Label Encoding

The target labels were one-hot encoded to prepare them for multi-class classification, converting categorical labels into binary vectors.

3.5 Model Architecture

The BiLSTM model was chosen for its ability to capture both forward and backward dependencies in sequential data. The architecture used is as follows:

- **Embedding Layer:** The first layer of the model was an embedding layer that converted the input numerical tokens into dense vectors of size 64.
- **Bidirectional LSTM Layer:** A 128-unit LSTM layer was applied in both forward and backward directions, allowing the model to capture dependencies in both directions.



- **Batch Normalization:** This was used to normalize the output of the LSTM layers, helping to stabilize and speed up the training process.
- **Dropout:** Dropout layers with rates of 0.4 and 0.5 were applied to prevent overfitting.
- **Dense Layer:** A fully connected layer with 64 units and ReLU activation was used before the output layer.
- **Output Layer:** A softmax output layer was used for multi-class classification.

3.6 Performance and Results

The model was trained for 50 epochs with a batch size of 32. The final validation accuracy achieved was 85.89%, demonstrating the effectiveness of the BiLSTM model in handling text sequence classification tasks.

3.7 Final Validation Accuracy

- Final Validation Accuracy: 85.89%

3.8 Validation Loss

The model converged smoothly, with the early stopping mechanism preventing overfitting by halting the training once the validation loss ceased to improve.

The model was also evaluated on the test set, and the predicted labels were derived by selecting the class with the highest probability from the softmax output.

3.9 Challenges and Observations

3.10 Training Time

One of the key challenges was the training time associated with LSTM models. Bidirectional LSTMs are computationally expensive due to their need to process sequences in both forward and backward directions.

3.10.1 Overfitting

Early stopping and dropout were essential in preventing overfitting, especially since LSTM models tend to overfit easily on smaller datasets.

3.10.2 Padding

Padding sequences ensured uniformity in input length but also introduced redundant information for shorter sequences, which may have slightly affected the model's efficiency.

3.11 Conclusion

The BiLSTM model performed well on the text sequence dataset, achieving a final accuracy of 85.89%. This result demonstrates that LSTM-based models are highly effective for text sequence tasks, as they can capture both short-term and long-term dependencies within the input sequences.

Future work could involve experimenting with other recurrent models like GRU (Gated Recurrent Units) or even Transformer-based architectures to further improve performance on the dataset.

4 TASK 2

For this dataset , we have various machine learning models trained on a combined dataset comprising three data sources: Emoticon, Text Sequence, and Feature datasets. The main objective was to evaluate the effectiveness of combining these datasets and to train different machine learning models to predict the labels accurately. The models were trained using different percentages of the training data (20%, 40%, 60%, 80%, and 100%) to assess how the amount of data impacts performance.

4.1 Preprocessing

Each dataset underwent separate preprocessing before being combined:

- **Emoticon Dataset:** Tokenized and converted into numerical arrays.
- **Text Sequence Dataset:** Tokenized into individual characters, transformed into numerical form, and padded to a uniform length.
- **Feature Dataset:** Standardized using `StandardScaler` to bring all features to a common scale.

The combined dataset was formed by concatenating the processed feature vectors from these three datasets.

4.2 Machine Learning Models

The following machine learning models were applied to the combined dataset, with performance evaluated at various training percentages.

4.2.1 Logistic Regression

Description: Logistic Regression is a linear model used for multi-class classification by modeling the probability of class membership.

Performance:

- 20% training data: 94.17% accuracy
- 40% training data: 95.79% accuracy
- 60% training data: 96.47% accuracy
- 80% training data: 97.25% accuracy
- 100% training data: 97.25% accuracy

Logistic Regression performed well but plateaued at 97.25% accuracy as the training data increased.

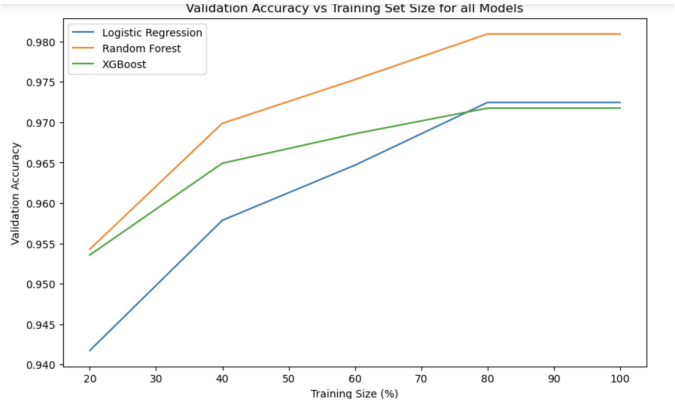
4.2.2 Random Forest

Description: Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions. It handles high-dimensional data efficiently and avoids overfitting.

Performance:

- 20% training data: 95.43% accuracy
- 40% training data: 96.99% accuracy
- 60% training data: 97.53% accuracy
- 80% training data: 98.09% accuracy
- 100% training data: 98.09% accuracy

Random Forest emerged as the best-performing model, achieving a maximum accuracy of 98.09% on the full dataset.



4.2.3 XGBoost

Description: XGBoost is a gradient boosting algorithm that iteratively improves its performance by learning from previous prediction errors.

Performance:

- 20% training data: 95.36% accuracy
- 40% training data: 96.49% accuracy
- 60% training data: 96.86% accuracy
- 80% training data: 97.18% accuracy
- 100% training data: 97.18% accuracy

XGBoost performed well but was slightly outperformed by Random Forest, particularly at higher percentages of training data.

4.3 Best Model for the Combined Dataset

Among the models tested, Random Forest performed the best, achieving an accuracy of 98.09% on the full training dataset. Its ensemble approach, combining multiple decision trees, allowed it to generalize well and capture complex patterns in the combined dataset.

5 Challenges and Observations

5.1 High Dimensionality

The combined dataset, consisting of features from three separate datasets, resulted in a high-dimensional feature space. This increased the computational complexity, particularly for models like XGBoost, which required more time to converge.

5.2 Model Performance

Logistic Regression and XGBoost performed well, but Random Forest outperformed both, especially when trained on larger portions of the data.

5.3 Overfitting

Although the high-dimensionality of the combined dataset posed a risk of overfitting, Random Forest's inherent ability to avoid overfitting helped mitigate this risk.

5.4 Conclusion

The Random Forest model was the best-performing model on the combined dataset, achieving an accuracy of 98.09% when trained on the full dataset. Combining the datasets allowed the model to capture more nuanced patterns, improving its predictive accuracy. Logistic Regression and XGBoost also performed well but did not surpass the accuracy of Random Forest.