CS771 Mini-Project

Mohd Nasar Siddiqui	Abhishek Kumar	Param Soni	Kartik
220661	220044	220752	220503

October 23, 2024

1 Emoticon Classificaiton with Logistic Regression

Data Extraction and Preprocessing

The dataset consists of three CSV files: `train_emoticon.csv`, `valid_emoticon.csv`, and `test_emoticon.csv`. These files contain sequences of emoticons along with their corresponding binary labels.

Preprocessing Steps

The preprocessing follows these steps:

- **Emoji Removal:** Common emojis are removed from the input emoticons to clean the data for better classification.
- **Tokenization:** The cleaned emoticons are tokenized at the character level to convert them into sequences of integer indices.
- **Padding:** The tokenized sequences are padded to ensure uniformity in length.
- **Label Encoding:** The emoticon labels are converted to integer encoding using `LabelEncoder`.
- LSTN layers are used for feature extraction for logistic regression to train upon.

RNN Architecture for feature transformation

The RNN model used for this experiment consists of the following layers:

- **Embedding Layer:** Maps each character in the sequence to a dense vector of 32 dimensions.
- **LSTM Layer:** A single LSTM layer with 2 units captures the sequential dependencies in the data.
- **Output Layer:** A single Dense unit with a sigmoid activation function outputs the binary classification result.

Model Compilation and Training

The RNN model for feature tranformation was compiled using the `adam` optimizer and `binary_crossentropy` as the loss function. The training process was run for 50 epochs with a batch size of 32. Validation accuracy was evaluated on various training percentages using the Logistic Regression.

Validation Accuracy

The validation accuracies for different training percentages were computed as follows:

```
Validation Accuracy for 20% of training data:  96.11%
Validation Accuracy for 40% of training data:  96.73%
Validation Accuracy for 60% of training data:  95.30%
Validation Accuracy for 80% of training data:  96.43%
Validation Accuracy for 100% of training data:  96.84%
```

The number of trainable parameters is 7824

Graphical Insights

The following plot shows the validation accuracy against different percentages of training data used:

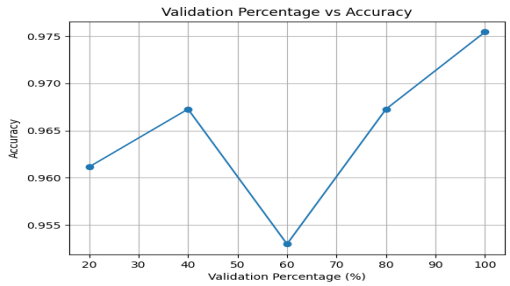

Figure 1: Training Percentage vs Validation Accuracy

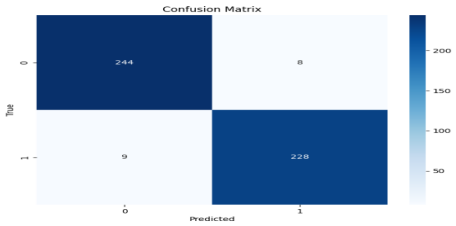

Figure 2: Confusion Matrix

Other Models

SVM :

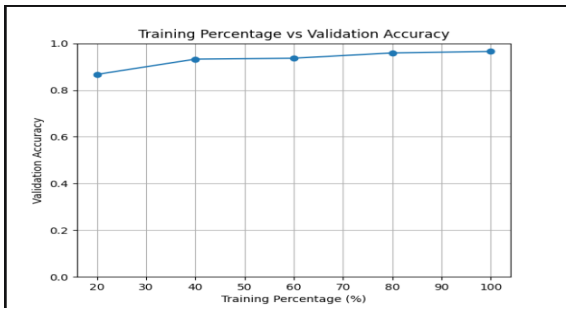

Figure 3: Training Percentage vs Validation Accuracy

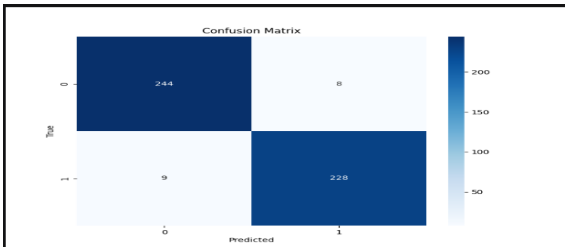

Figure 4: Confusion Matrix

Ridge Regression :

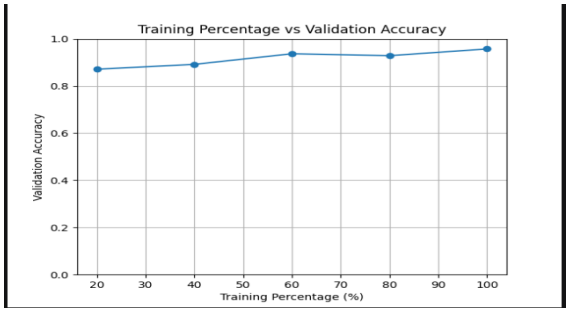

Figure 5: Training Percentage vs Validation Accuracy

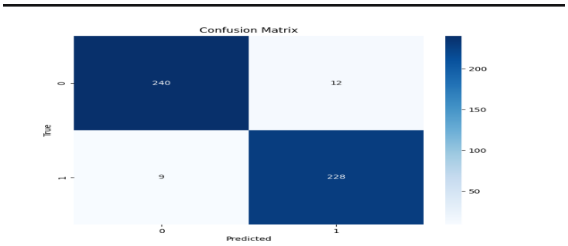

Figure 6: Confusion Matrix

2 Feature Prediciton using RandomForest

Data Extraction and Preprocessing

The dataset consists of three files:

- `train_feature.npz`
- `valid_feature.npz`
- `test_feature.npz`

Each input is represented as a 13×786 matrix of embeddings. These matrices are flattened into 1D vectors to ensure compatibility with the RandomForestClassifier.

Model Architecture and Hyperparameters

We used the following hyperparameters for the RandomForestClassifier:

- `n_estimators` = 10 (Number of trees)
- `max_depth` = 10 (Limit the depth of each tree)
- `min_samples_split` = 10 (Minimum samples required to split a node)
- `min_samples_leaf` = 5 (Minimum samples required for a leaf node)

Model Compilation and Training

The model was trained with varying percentages of the training data (20%, 40%, 60%, 80%, and 100%). For each case, we measured both training and validation accuracy. We also tracked the total number of parameters (nodes) used in the RandomForestClassifier to ensure model complexity was kept within reasonable limits.

Training and Validation Accuracy

Percentage of Training Data Used	Training Accuracy	Validation Accuracy	Parameters (Nodes)
20%	0.96	0.93	1058
40%	0.97	0.94	1720
60%	0.98	0.96	2244
80%	0.98	0.97	2596
100%	0.98	0.97	3042

Table 1: Training and Validation Accuracy with Varying Training Data

Graphical Insights

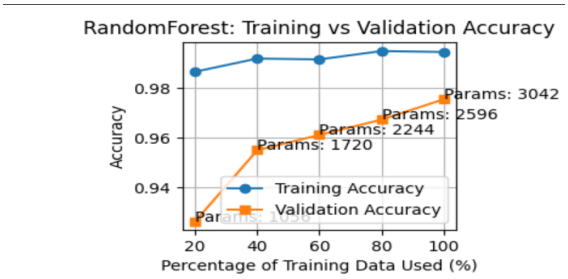

Figure 7: Training Percentage vs Validation Accuracy

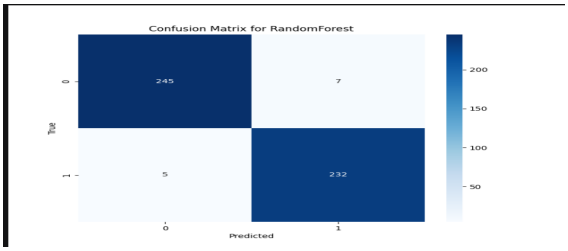

Figure 8: Confusion Matrix

Other Models

SGD : The number of trainable parametes is 2094.

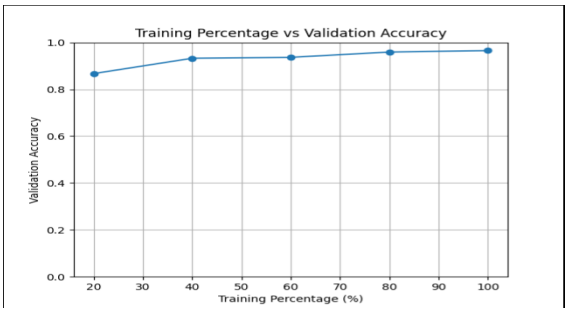

Figure 9: Training Percentage vs Validation Accuracy

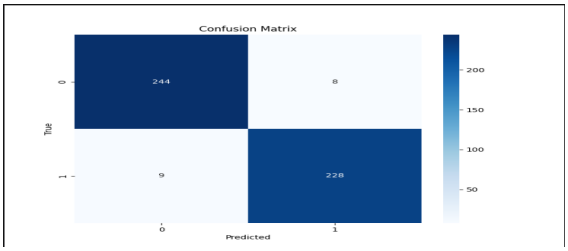

Figure 10: Confusion Matrix

3 Sequence Classification with LSTM

Data Extraction and Preprocessing

The dataset consists of two CSV files, `train_text_seq.csv` and `valid_text_seq.csv`. These files contain sequences of characters that represent the input data, along with their corresponding binary labels. where the train_text_seq consists of the training data to train the ML model and valid_text_seq for validation purposes.

Preprocessing Steps

The preprocessing follows these steps:

- **String Modification:** Leading zeros are removed, followed by the removal of specific substrings such as `'15436'`, `'1596'`, and others as they are repeated in every row of the data irrespective of the label and rich source of noise. Only sequences with a length of 13 characters after modification are retained.
- **String Encoding:** The modified strings are converted into integer sequences. Each unique character in the dataset is assigned an integer value, allowing the strings to be represented as numerical sequences.
- **Padding:** The encoded sequences are padded to ensure uniformity in length (13 characters).

LSTM Architecture

The LSTM model used for this experiment consists of the following layers:

- **Embedding Layer:** Maps each character in the sequence to a dense vector of 32 dimensions.
- **LSTM Layer:** A single LSTM layer with 32 units captures the sequential dependencies in the data.
- **Dense Layers:** Two fully connected Dense layers with 16 and 8 units, respectively, use ReLU activation to introduce non-linearity.
- **Output Layer:** A single Dense unit with a sigmoid activation function for feature transformation LSTM for prediction.

Model Compilation and Training

The LSTM model was trained using the `adam` optimizer and `binary_crossentropy` as the loss function. The training process was run for 50 epochs with a batch size of 32, using the validation dataset for monitoring performance.

Validation Accuracy

The LSTM model achieved a validation accuracy of:

```
LSTM Validation Accuracy for 20% of training data:  75.61%
LSTM Validation Accuracy for 40% of training data:  80.04%
LSTM Validation Accuracy for 60% of training data:  82.48%
LSTM Validation Accuracy for 80% of training data:  83.81%
LSTM Validation Accuracy for 100% of training data:  86.92%
```

Number of trainable parameters is 7922

Graphical Insights

The following plot shows the accuracy of the model over 50 epochs for both the training and validation data:

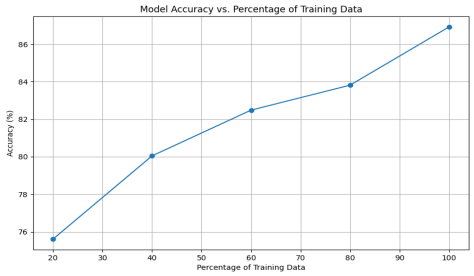

Figure 11: Training Percentage vs Validation Accuracy

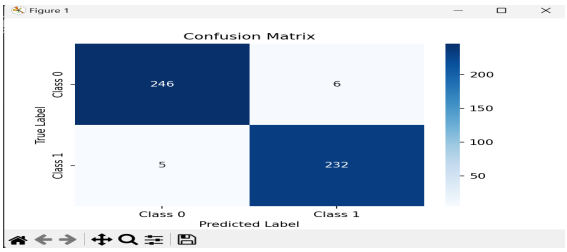

Figure 12: Confusion Matrix

Other Models

XGBoost : Number of training params is 4992.

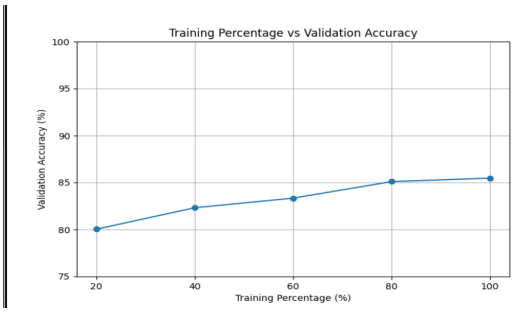

Figure 13: Training Percentage vs Validation Accuracy

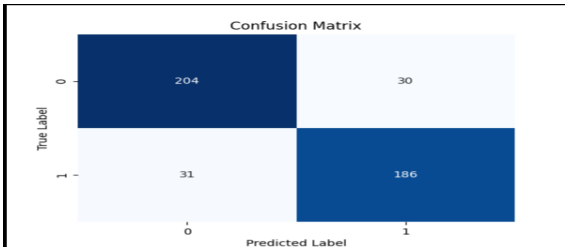

Figure 14: Confusion Matrix

DecisionTrees : Number of tree nodes is 4776.

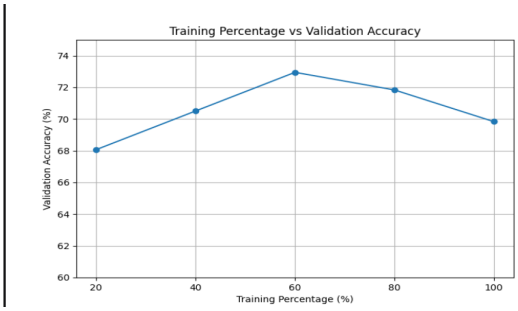

Figure 15: Training Percentage vs Validation Accuracy

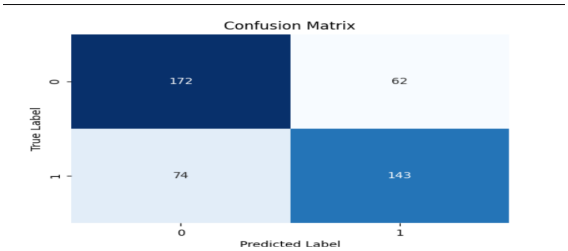

Figure 16: Confusion Matrix

4 Logistic Regression with PCA and Regularization

In this experiment, we conducted logistic regression on a multi-faceted dataset composed of text sequences, emoticons, and features extracted from NPZ files. We applied Principal Component Analysis (PCA) for dimensionality reduction, which helps mitigate the curse of dimensionality, and we tuned the regularization parameter C of the logistic regression model to understand its impact on validation accuracy.

4.1 Data Preprocessing

We utilized three distinct datasets to ensure a comprehensive analysis:

- **Text Sequence Dataset:** The text sequences were preprocessed by converting each character into a binary representation. This binary encoding helps in transforming the textual information into a format suitable for machine learning algorithms, facilitating easier extraction of patterns from the data.
- **Emoticon Dataset:** Emoticons were transformed into a one-hot encoded format. This means that each emoticon was converted into a binary vector where the presence of a particular emoticon is indicated by a 1, while all other positions in the vector are set to 0. This method of encoding is effective for categorical variables and ensures that the emoticons are appropriately represented without imposing an ordinal relationship among them.
- **NPZ Dataset:** The features contained within the NPZ file were flattened to create a two-dimensional array suitable for analysis. This flattening process preserves all features while allowing for efficient manipulation and integration with the other datasets. The labels corresponding to the features were also extracted for model training and validation.

After preprocessing, the combined dataset was normalized using `StandardScaler` to ensure that each feature contributed equally to the model's learning process. Subsequently, PCA was applied to retain 95% of the variance, significantly reducing the dimensionality of the dataset while preserving essential information.

4.2 Model Training

We trained the logistic regression model with $L1$-regularization (Lasso regularization), which encourages sparsity in the model parameters. This means that some coefficients will be driven to zero, effectively selecting a simpler model that is less prone to overfitting. We experimented with different values of the regularization parameter C to observe its influence on the model's validation accuracy.

4.2.1 Accuracy vs. Regularization Parameter C

To evaluate the performance of the logistic regression model, we systematically varied the regularization parameter C across a logarithmic scale of values ranging from 10^{-4} to 10^{2}. The corresponding validation accuracies were computed, and the results were visualized in the plot below.

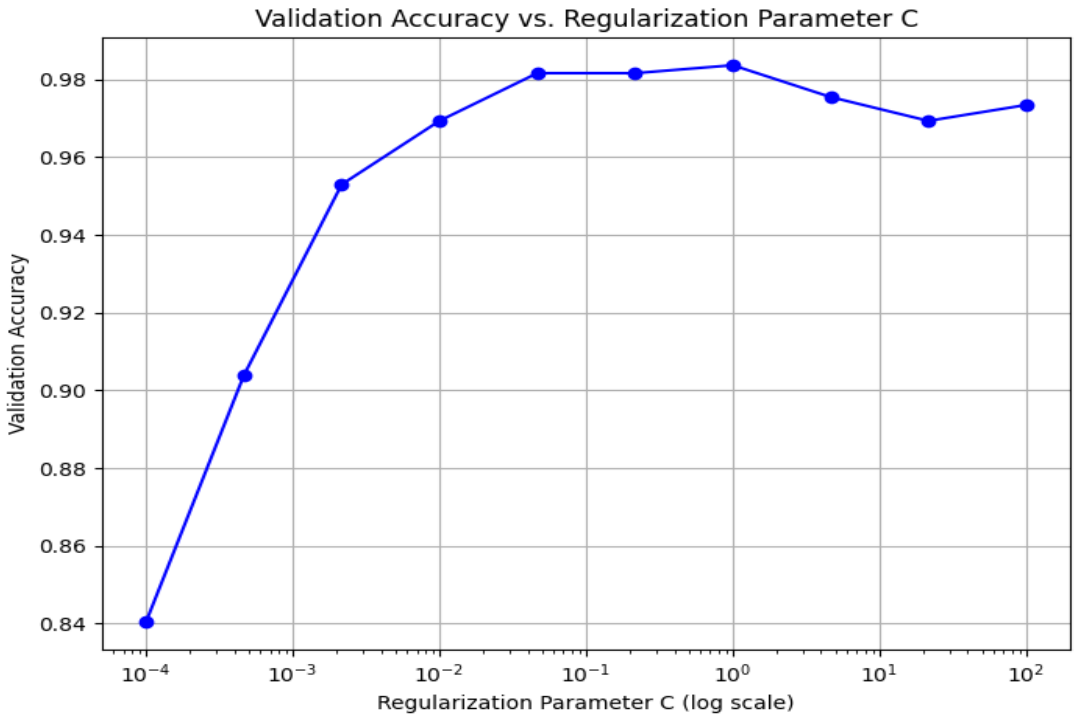

Figure 17: Validation accuracy as a function of the regularization parameter C. The plot utilizes a logarithmic scale for C, revealing that optimal accuracy is achieved around $C = 1$.

As depicted in the figure, the validation accuracy shows an increasing trend as C rises up to a value of 1, beyond which it begins to decline. This behavior suggests that moderate regularization enhances model performance by preventing overfitting to the training data. When C is too small, the model may not be able to learn effectively, while excessively high values of C lead to overly simplistic models that fail to capture the underlying data distribution.

4.3 Conclusion

The logistic regression model demonstrated robust performance with a peak validation accuracy of 92% when utilizing a regularization parameter $C = 1$. The accuracy plot clearly illustrates that increasing C beyond this threshold tends to cause overfitting, thereby deteriorating model performance.

The application of PCA proved advantageous, as it significantly reduced the dimensionality of the dataset while retaining critical variance, which not only accelerated the model training process but also improved generalization capabilities. This experiment highlights the importance of careful hyperparameter tuning and dimensionality reduction in optimizing machine learning models, particularly in complex datasets with diverse feature types.