**Introduction**

In this Document I Explained only for the LSTM method , Random Forest and Logistic regression is not included for this.

To build an emotion classification model using a Bidirectional Long Short-Term Memory (BiLSTM) neural network. The model takes text input and predicts the emotion associated with the text. The following document provides a detailed explanation of the methodology, model architecture, hyperparameter tuning, evaluation results, and instructions on how to run the project.

**Methodology**

The methodology for this project includes the following steps:

1. **Data Preprocessing**:
   * Tokenizing the text data.
   * Padding the sequences to ensure uniform input length.
2. **Model Building**:
   * Creating a BiLSTM model with specific layers and configurations.
3. **Training**:
   * Compiling the model with appropriate loss functions and optimizers.
   * Training the model on the training data.
4. **Evaluation**:
   * Evaluating the model on test and validation data.
   * Predicting emotions on new text data.

**Model Architecture**

The model architecture is a Bidirectional Long Short-Term Memory (BiLSTM) neural network. The following layers are included in the model:

1. **Embedding Layer**: Converts the input sequences into dense vectors of fixed size.
2. **Spatial Dropout Layer**: Applies dropout to the embedded input to prevent overfitting.
3. **First BiLSTM Layer**: Captures temporal dependencies in both forward and backward directions.
4. **Second BiLSTM Layer**: Further captures temporal dependencies.
5. **Dense Layer**: Applies a fully connected layer with ReLU activation.
6. **Dropout Layer**: Applies dropout to the dense layer to prevent overfitting.
7. **Output Layer**: Produces the final output with the appropriate activation function (sigmoid for binary classification, softmax for multi-class classification).

**Hyperparameter Tuning**

The following hyperparameters were tuned for optimal performance:

1. **Embedding Dimension**: Set to 128.
2. **LSTM Units**: Set to 128 for the first BiLSTM layer and 64 for the second BiLSTM layer.
3. **Dropout Rate**: Set to 0.3 for spatial dropout, LSTM dropout, and recurrent dropout.
4. **Output Dimension**: Set to 7 (number of emotions).
5. **Output Activation**: Set to 'sigmoid' for multi-label classification.

**Evaluation Results**

The model's performance is evaluated on both test and validation datasets. The predictions are as follows:

* **Test Predictions**: [[1.83380398e-04, 3.15165453e-05, 6.26723886e-06, ..., 1.45611164e-04, 2.09382371e-04, 9.99410689e-01], ...]
* **Validation Predictions**: [[1.42580573e-03, 1.11060348e-04, 2.34577135e-04, ..., 2.88585783e-03, 2.12021843e-02, 5.47596633e-01], ...]

**Instructions to Run the Project**

To run this project and reproduce the results, follow these steps:

1. **Install Dependencies**: Ensure you have the required libraries installed. You can use the following command:

**Create a virtual env using - python -m venv venv**

**Change directory to venv/scripts/activate (activate the virtual environment)**

**Then install the requirements.txt**

pip install -r requirements.txt

1. **Load Data**: Prepare your dataset and ensure it is in the required format. The data should be tokenized and padded as shown in the preprocessing step.
2. **Run the Notebook**: Execute the cells in the provided Jupyter notebook main.ipynb in order. Ensure you follow the sequence to preprocess data, build the model, train, and evaluate.
3. **Make Predictions**: Use the following code snippet to make predictions on new data:

Python - Copy code

loaded\_lstm\_model = create\_lstm\_model(output\_dim, output\_activation)

loaded\_lstm\_model.load\_weights('lstm\_emotion\_model.h5')

new\_data = ["Sample text to predict emotion"]

new\_data\_seq = tokenizer.texts\_to\_sequences(new\_data)

new\_data\_pad = pad\_sequences(new\_data\_seq, maxlen=200)

predictions = loaded\_lstm\_model.predict(new\_data\_pad)

print("Predictions:", predictions)

By following these steps, you should be able to reproduce the results and make predictions on new text data. ​