CE-C24042 - Neural Networks and Deep Learning

COURSE PROJECT

Contents

INTRODUCTION

Problem Statment Selection | Motivation | Strategy

Problem Statement

Create a definitive and structured custom model architecture for multi - class sentiment analysis of tweets

Evaluation Criterion

Accuracy and Categorical Accuracy from the inbuilt Keras methods

Categorical Cross Entropy is the loss function used

$$CCE(p,t) = -\sum_{c=1}^{C} t_{o,c} \log\left(p_{o,c}\right)$$

Team Strategy

Initial thoughts | Approach to the problem

- Explore Data
- Feature Engineering
- Machine Learning Pipeline
- Postprocessing
- Evaluation

PART 1: DATA PREPARATION

Dataset + Features Overview

Dataset Overview

Some fast facts about the datasets used -

a. A total of 2 datasets given

b. Supervised Classification Problem

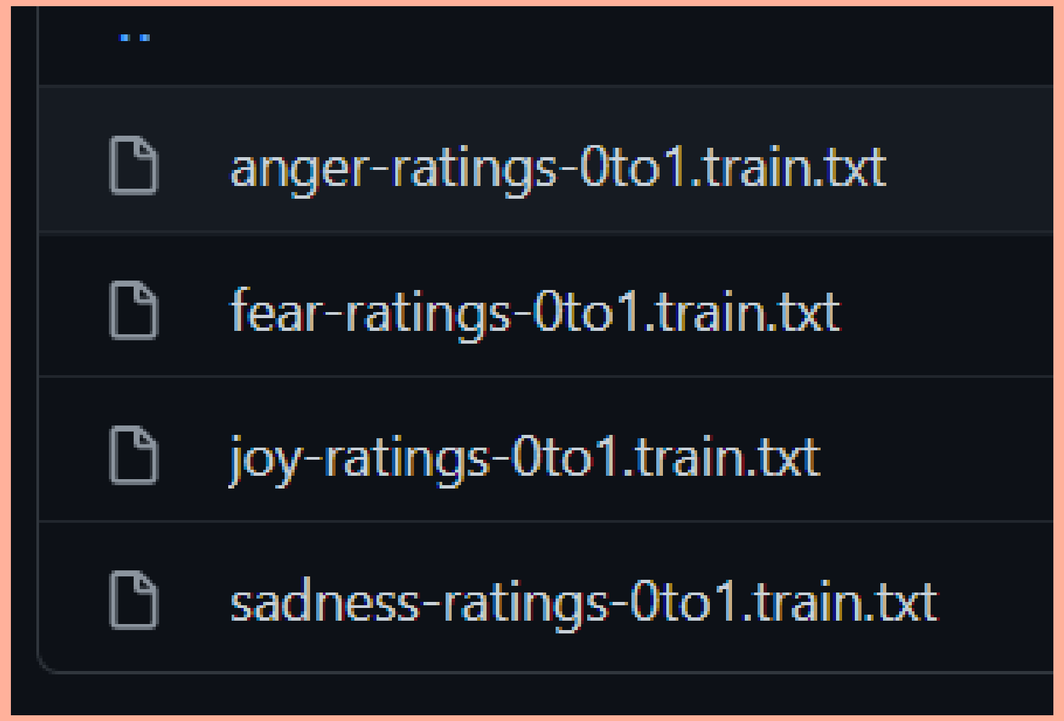
c. Presence of missing values

d. Multiple classes of emotions

Dataset 1: CROWDFLOWER

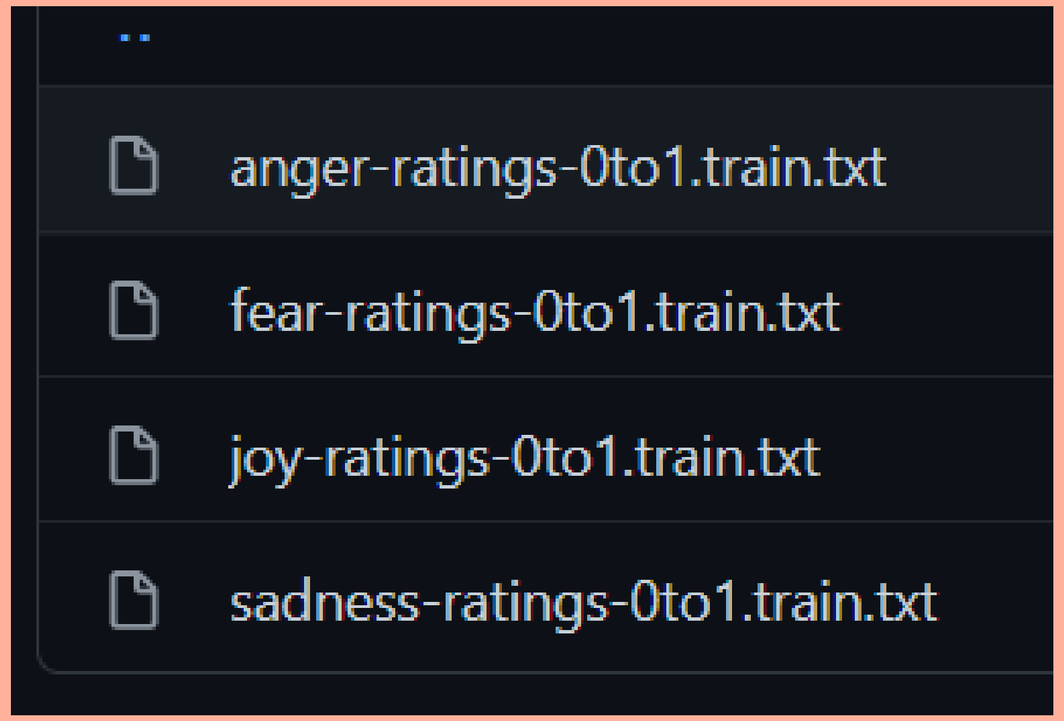
- 13 emotion labels
- 100s to 1000s of examples

Dataset 2: WASSA2017



- Dataset used in research paper: deepCyberNet at EmoInt-2017
- Focuses on 4 emotions: anger, fear, joy, sadness
- Ratings for each emotion (0 to 1 range)
- 4 txt files
- Have to be converted to csv

Dataset 2: WASSA2017



- Dataset used in research paper: deepCyberNet at EmoInt-2017
- Focuses on 4 emotions: anger, fear, joy, sadness
- Ratings for each emotion (0 to 1 range)
- 4 txt files
- Have to be converted to csv

PART 2: EXPLORATORY DATA ANALYSIS

Combined Dataset

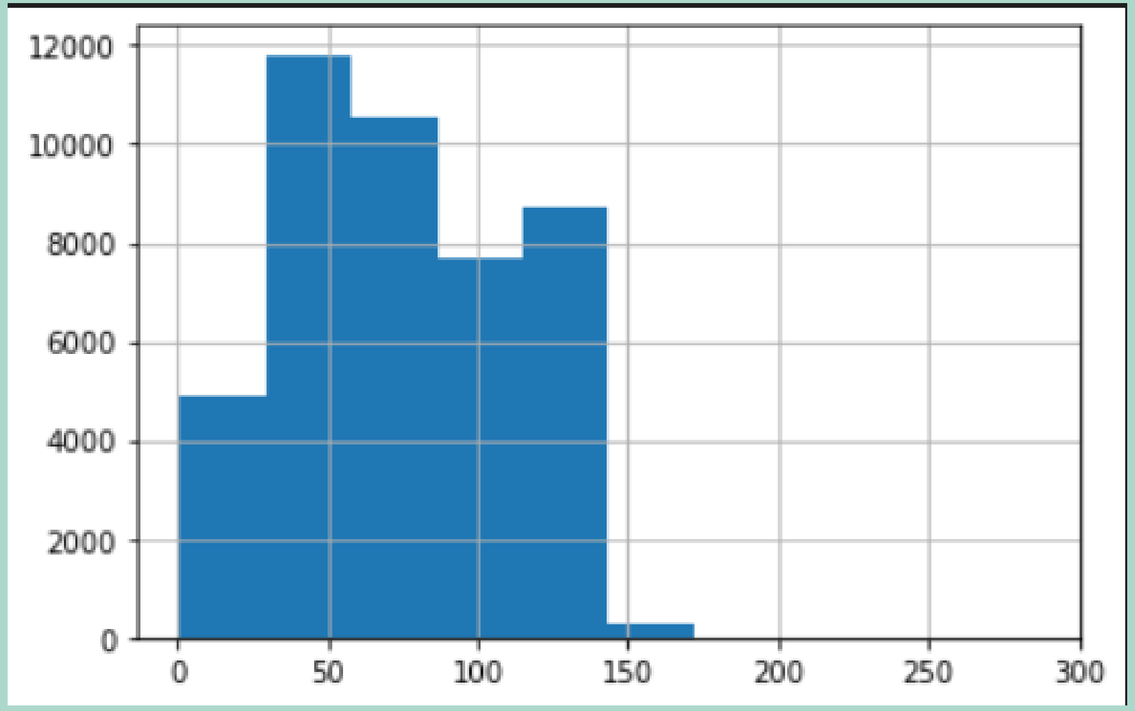
- **tweet_id, tweet, tweettype**
- Expected distribution of tweet length
- Number of words ranging from 1 to 35
- Avg. word length < 20

tweet_id	tweet	tweettype
0	@ZubairSabirPTI pls dont insult the word 'Molna'	anger
1	@ArcticFantasy I would have almost took offens...	anger
2	@IllinoisLoyalty that Rutgers game was an abom...	anger
3	@CozanGaming that's what lisa asked before she...	anger
4	Sometimes I get mad over something so minuscul...	anger
...	...	...
43955	@JohnLloydTaylor	neutral
43956	Happy Mothers Day All my love	love
43957	Happy Mother's Day to all the mommies out ther...	love
43958	@niariley WASSUP BEAUTIFUL!!! FOLLOW ME!! PEE...	happiness
43959	@mopedronin bullet train from tokyo the gf ...	love

ows × 3 columns

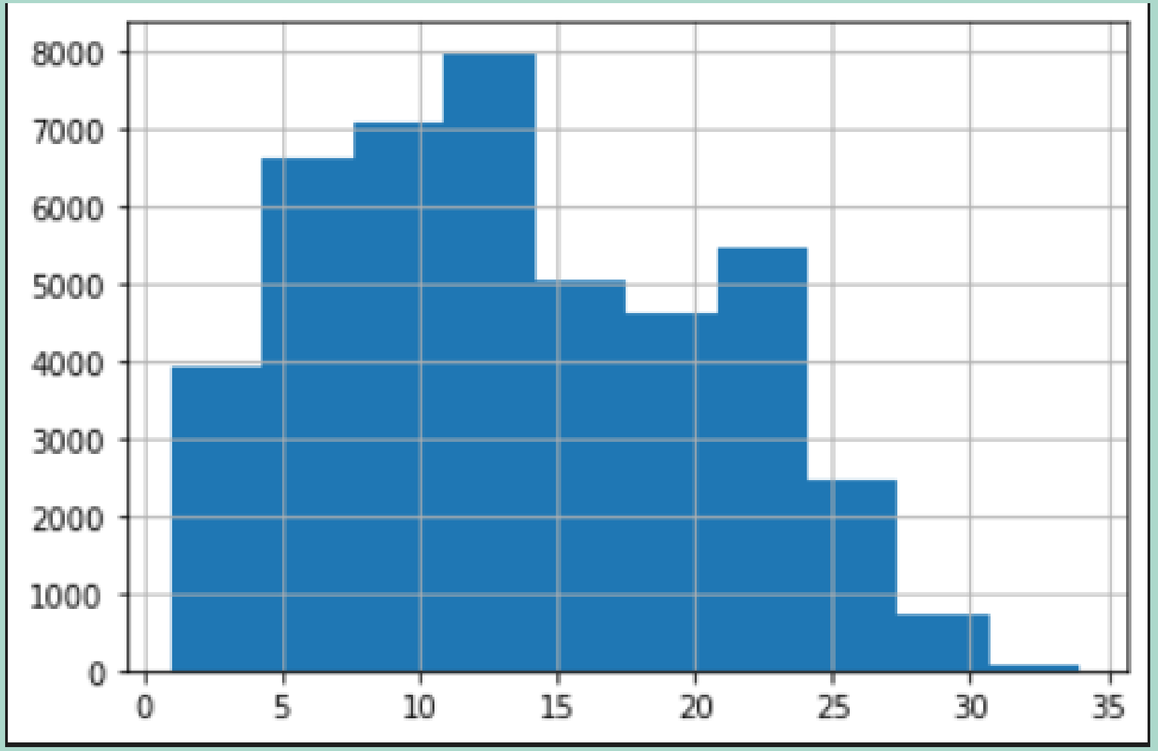
Combined Dataset

- tweet_id, tweet, tweettype
- **Expected distribution of tweet length**
- Number of words ranging from 1 to 35
- Avg. word length < 20



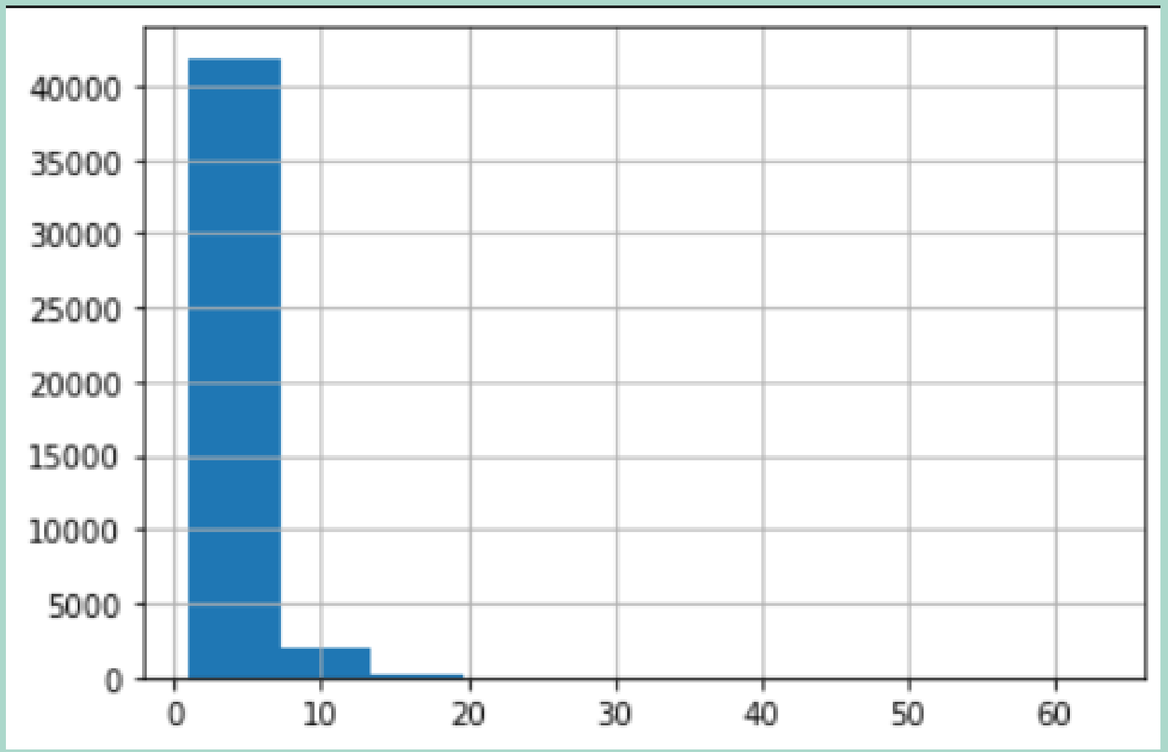
Combined Dataset

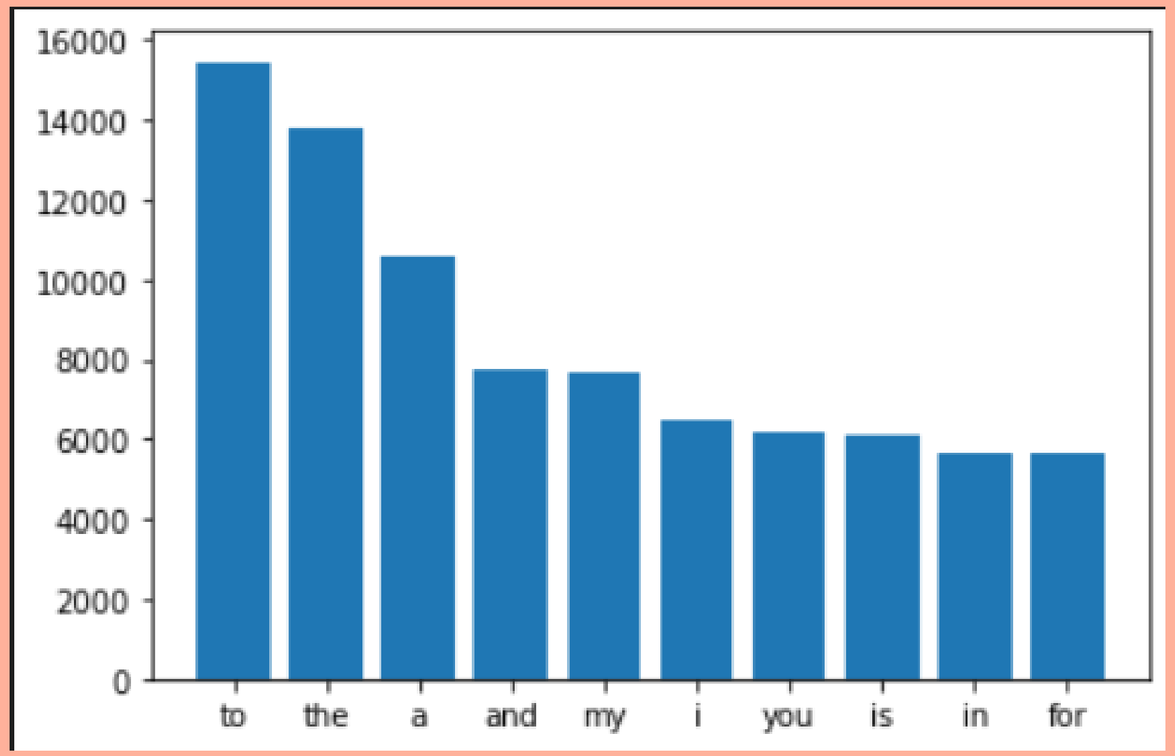
- tweet_id, tweet, tweettype
- Expected distribution of tweet length
- **Number of words ranging from 1 to 35**
- Avg. word length < 20



Combined Dataset

- tweet_id, tweet, tweettype
- Expected distribution of tweet length
- Number of words ranging from 1 to 35
- **Avg. word length < 20**

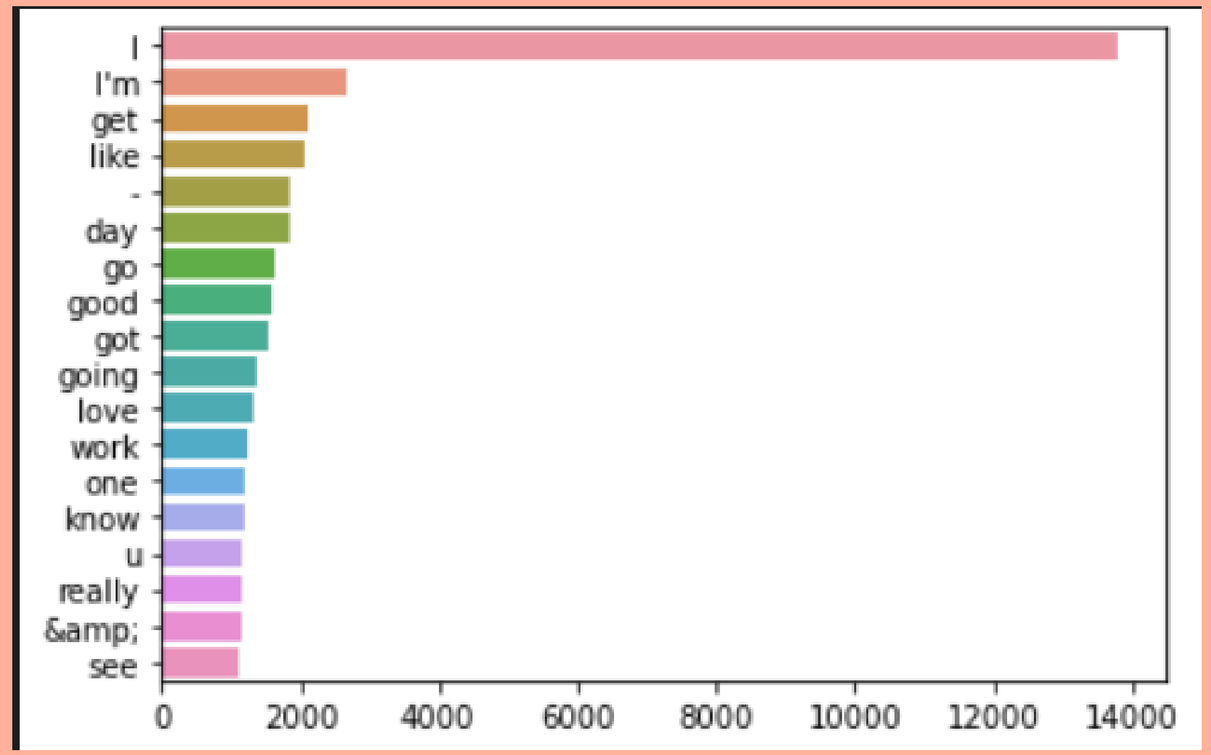




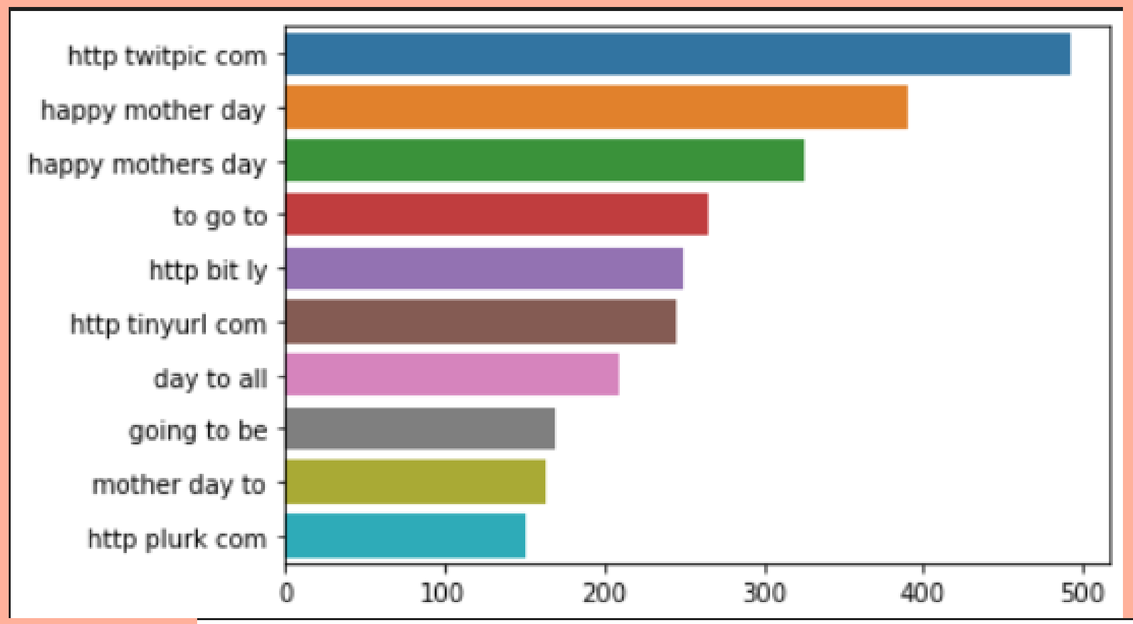
NLP based EDA

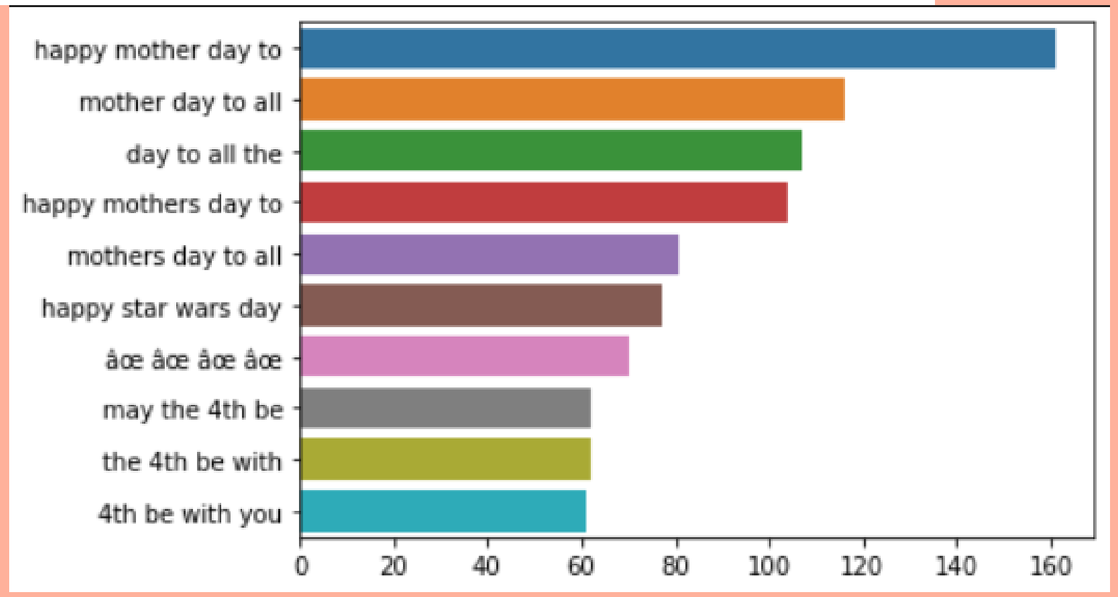
- **Most common stopwords**
- Most common non - stopwords
- Most common phrases
- Wordcloud
- Tweet types distribution
- After combining some types

NLP based EDA



- Most common stopwords
- **Most common non - stopwords**
- Most common phrases
- Wordcloud
- Tweet types distribution
- After combining some types



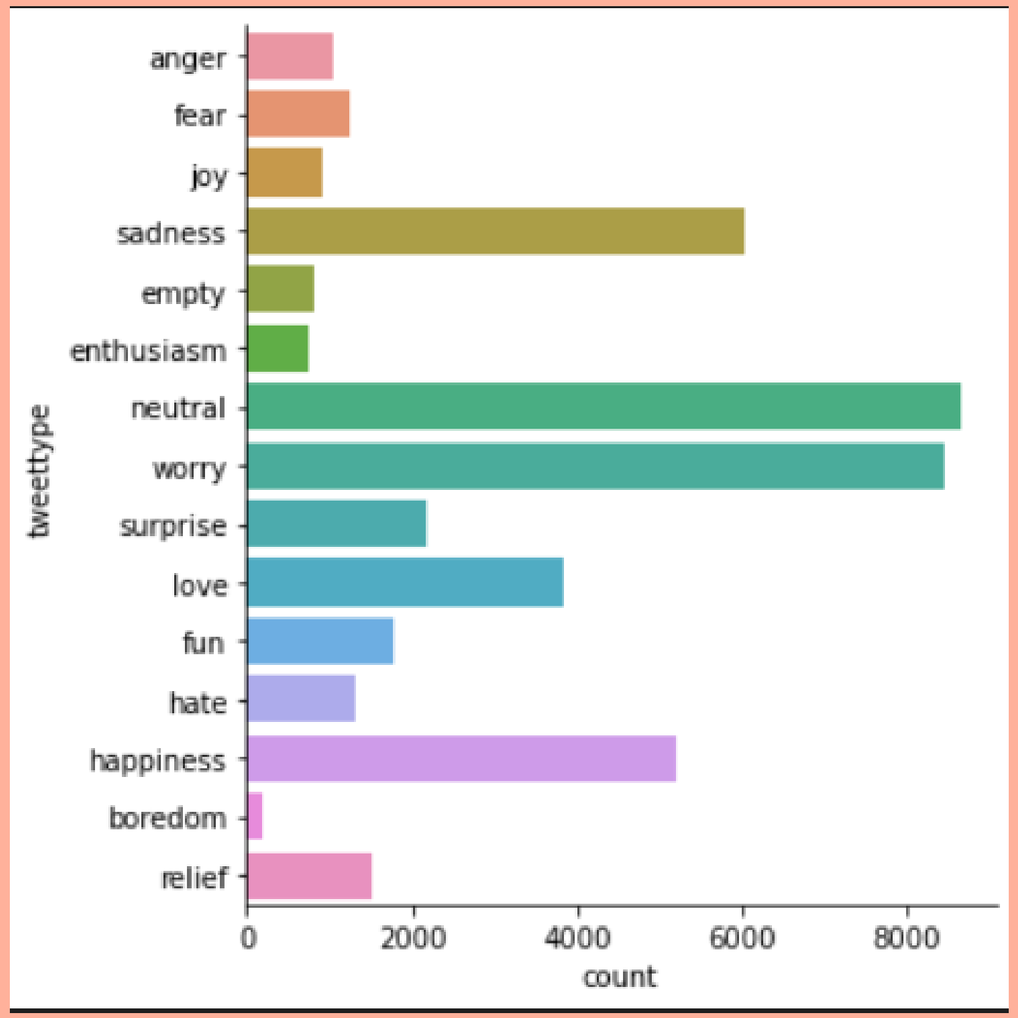


NLP based EDA

- Most common stopwords
- Most common non - stopwords
- **Most common phrases**
- Wordcloud
- Tweet types distribution
- After combining some types

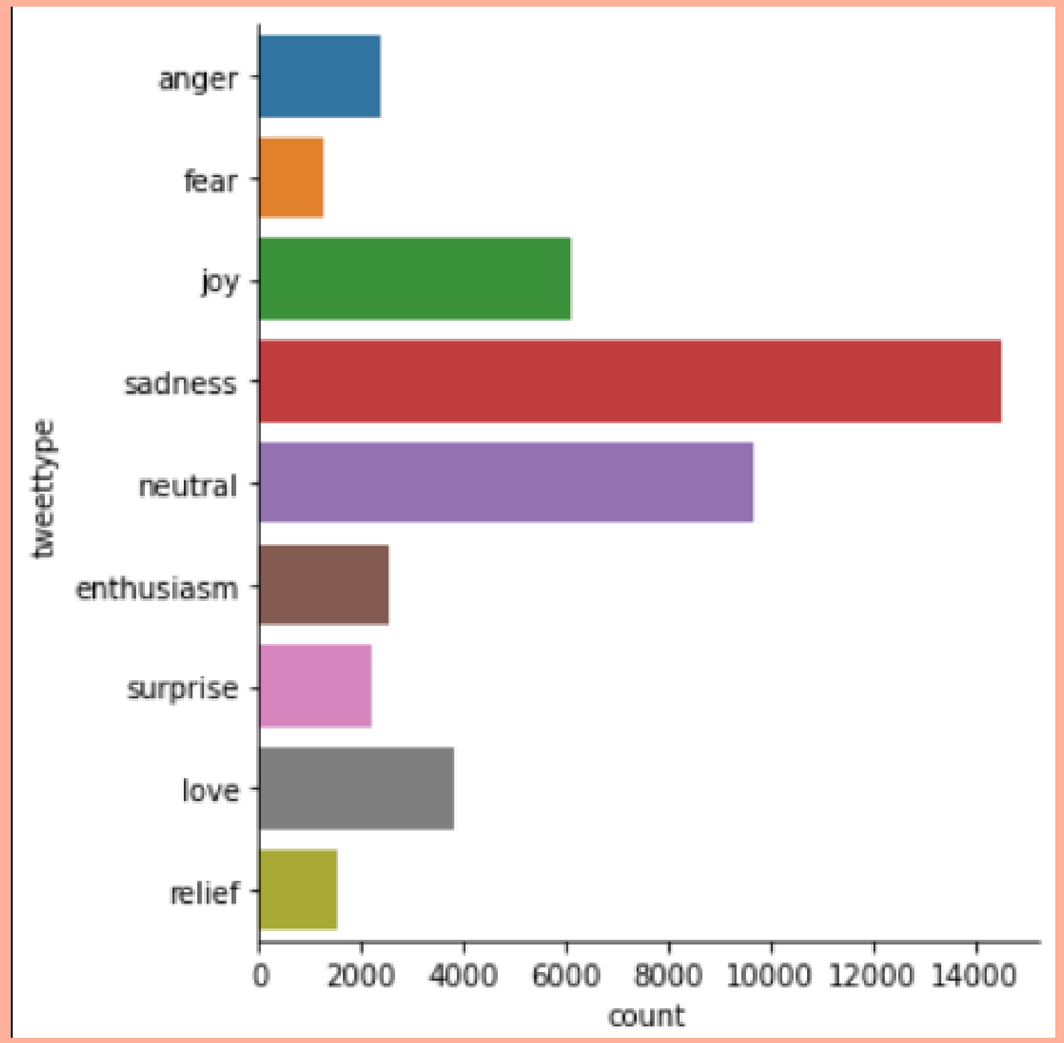
NLP based EDA

- Most common stopwords
- Most common non - stopwords
- Most common phrases
- **Wordcloud**
- Tweet types distribution
- After combining some types



NLP based EDA

- Most common stopwords
- Most common non - stopwords
- Most common phrases
- Wordcloud
- **Tweet types distribution**
- After combining some types



NLP based EDA

- Most common stopwords
- Most common non - stopwords
- Most common phrases
- Wordcloud
- Tweet types distribution
- **After combining some types**

Summary

a. Combine datasets, Merging strategy

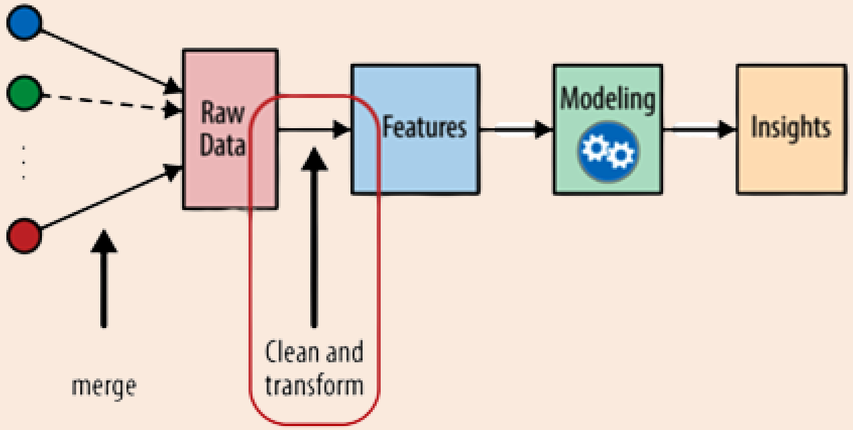
b. Missing values

c. Unnecessary columns

d. Importance of pre-processing

PART 3: FEATURE ENGINEERING

Feature Engineering Techniques | Lemmatization



Feature Engineering

From the exploratory data analysis of the datasets, it is imperative that the given data is manipulated in order to build the machine learning model for solving the given data science problem.

```
# Applying the transformation to create columns
tweetData['charCount'] = tweetData["tweet"].apply(lambda x:count_characters(x))
tweetData['wordCount'] = tweetData["tweet"].apply(lambda x:count_words(x))
tweetData['sentenceCount'] = tweetData["tweet"].apply(lambda x:count_sentences(x))
tweetData['capCharCount'] = tweetData["tweet"].apply(lambda x:count_capital_characters(x))
tweetData['capWordCount'] = tweetData["tweet"].apply(lambda x:count_capital_words(x))
tweetData['quotedWordCount'] = tweetData["tweet"].apply(lambda x:count_words_in_quotes(x))
tweetData['stopwordCount'] = tweetData["tweet"].apply(lambda x:count_stopwords(x))
tweetData['uniqueWordCount'] = tweetData["tweet"].apply(lambda x:count_unique_words(x))
tweetData['hashCount'] = tweetData["tweet"].apply(lambda x:count_hashtags(x))
tweetData['mentionCount'] = tweetData["tweet"].apply(lambda x:count_mentions(x))
tweetData['punctCount'] = tweetData["tweet"].apply(lambda x:count_punctuations(x))
tweetData['avgWordLen'] = tweetData['charCount']/tweetData['wordCount']
```

Feature Engineering

Different features based on the sentence structure and tweet were extracted for example: word count, @ mentions, etc.

Feature Extraction

*

*

*

*

*

```python
# Function to modulate a tweet
def featureEngineering(tweet):
    # Lower case tweet
    tweetMod = tweet.lower()
    # Replace URLs with a space in the message
    tweetMod = re.sub('https?:\/\/[a-zA-Z0-9@:%._\/+~#=?&;-]*', ' ', tweetMod)
    # Replace ticker symbols with a space. The ticker symbols are any stock symbol that starts with $.
    tweetMod = re.sub('\$[a-zA-Z0-9]*', ' ', tweetMod)
    # Replace StockTwits usernames with a space. The usernames are any word that starts with @.
    tweetMod = re.sub('\@[a-zA-Z0-9]*', ' ', tweetMod)
    # Replace everything not a letter or apostrophe with a space
    tweetMod = re.sub('[^a-zA-Z\']', ' ', tweetMod)
    # Remove single letter words
    tweetMod = ' '.join( [w for w in tweetMod.split() if len(w)>1] )

    return tweetMod
```

Feature Engineering

- Convert to lowercase
- Replace URLs with a space
- Convert @ mentions to username
- Replace everything not a letter or apostrophe with space
- Remove single letter words

* Feature Extraction
* Tweet Cleanup
*
*
*

```
ls = [word for word in word_tokenize(tweet) if (word.isalpha()
move stop words
 = set(stopwords.words('english'))
ls = [word for word in words if (word not in stop)]
```

```
# Padding of sequences based on number of unique words
tokenizer = Tokenizer(num_words=27608, split=' ')
tokenizer.fit_on_texts(tweetData['lemmatizedText'].values)
X = tokenizer.texts_to_sequences(tweetData['lemmatizedText'].values)
X = pad_sequences(X)
```

Feature Engineering

Tokenization is the process of breaking down a phrase, sentence, paragraph, or an entire text document into smaller parts, such as individual words or phrases. Tokens are the name given to each of these smaller units.

- Feature Extraction
- Tweet Cleanup
- Tokenization
-
-

```
labels = np.array(tweetData['tweettype'])
y = []
for i in range(len(labels)):
    if labels[i] == 'sadness':
        y.append(0)
    elif labels[i] == 'neutral':
        y.append(1)
    elif labels[i] == 'joy':
        y.append(2)
    elif labels[i] == 'love':
        y.append(3)
    elif labels[i] == 'enthusiasm':
        y.append(4)
    elif labels[i] == 'anger':
        y.append(5)
    elif labels[i] == 'surprise':
        y.append(6)
    elif labels[i] == 'relief':
        y.append(7)
    elif labels[i] == 'fear':
        y.append(8)
y = np.array(y)
labels = tf.keras.utils.to_categorical(y, 9, dtype="float32")
del y
```

Feature Engineering

Label Encoding is used to convert our output labels to numberical representations for easy input into the neural network for training

* Feature Extraction
* Tweet Cleanup
* Tokenization
* Label Encoding
*

```python
def lemmatizeTweet(tweet):
    words = [word for word in word_tokenize(tweet) if (word.isalpha()==1)]
    # Remove stop words
    stop = set(stopwords.words('english'))
    words = [word for word in words if (word not in stop)]
    # Lemmatize words (first noun, then verb)
    wnl = nltk.stem.WordNetLemmatizer()
    lemmatized = [wnl.lemmatize(wnl.lemmatize(word, 'n'), 'v') for word in words]
    return " ".join(lemmatized)
```

Feature Engineering

Lemmatization typically refers to doing things correctly by using a vocabulary and morphological analysis of words, with the goal of removing only inflectional ends and returning the base or dictionary form of a word, which is known as the lemma.

- Feature Extraction
- Tweet Cleanup
- Tokenization
- Label Encoding
- Lemmatization

Summary

a. Final feature count is 46

b. 27608 unique words

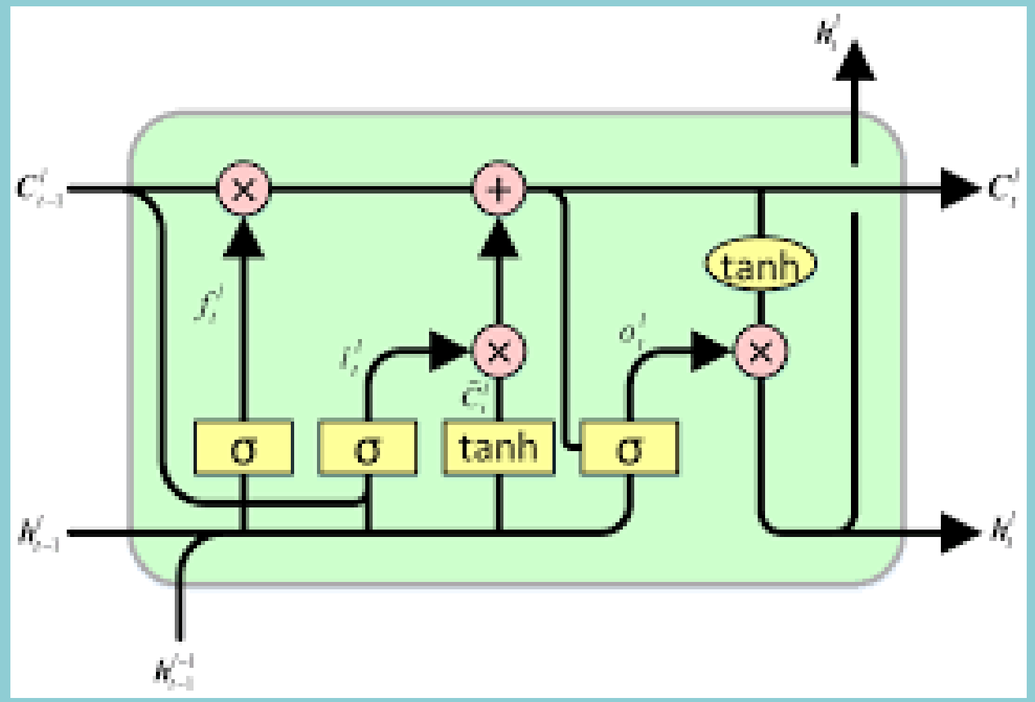
c. Data pre - processed and ready for moded construction and training

PART 2: MODEL ARCHITECTURES

Experiments | Observations

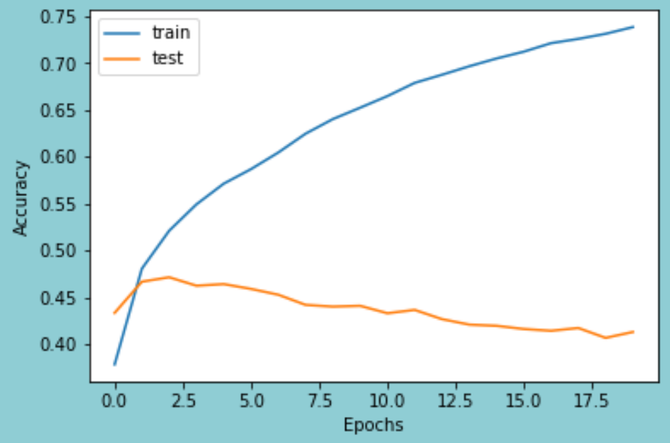
LSTM: What and Why?

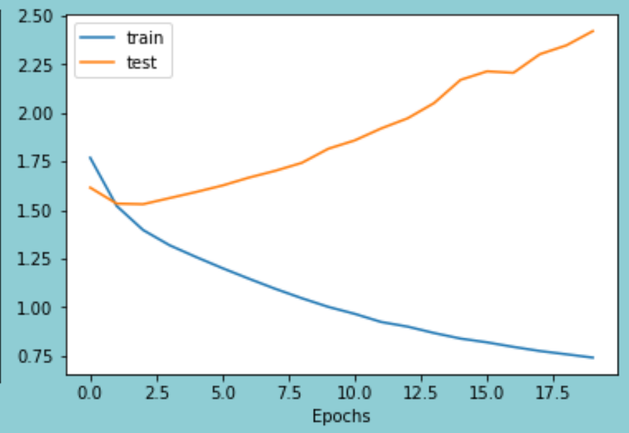
- Long Short Term Memory
- Recurrent Neural Network
- Local and global information
- Flexibility in controlling the outputs
- Variations of LSTM



Model Architectures

The team spent time trying out different custom architectures to find what suits best for our dataset with unique problems of 9 labels as compared to 3 in the usual sentiment analysis, as well as skewed and less number of data points.

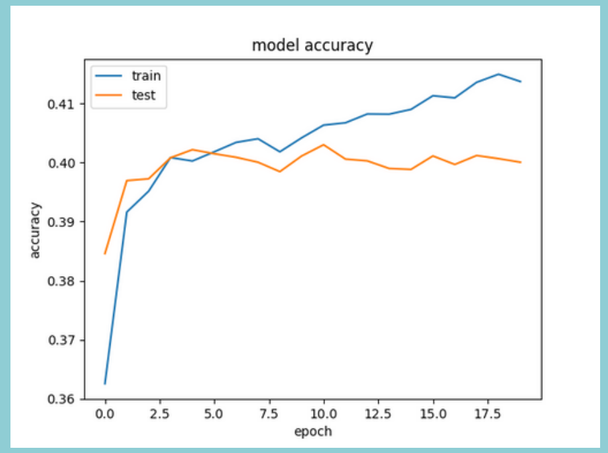


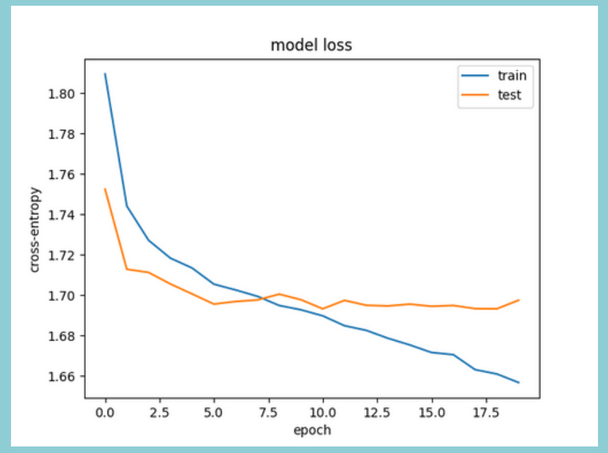


Model Architectures

A Vanilla LSTM is an LSTM model that has a single hidden layer of LSTM units, and an output layer to make a prediction. We use a Basic LSTM with Spatial Dropout in this experiment.

Vanilla LSTM

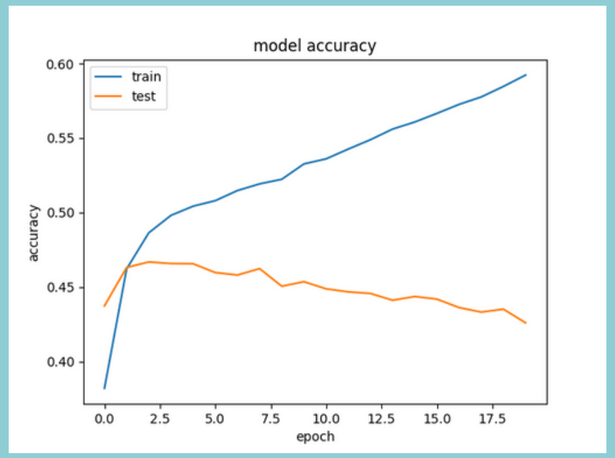


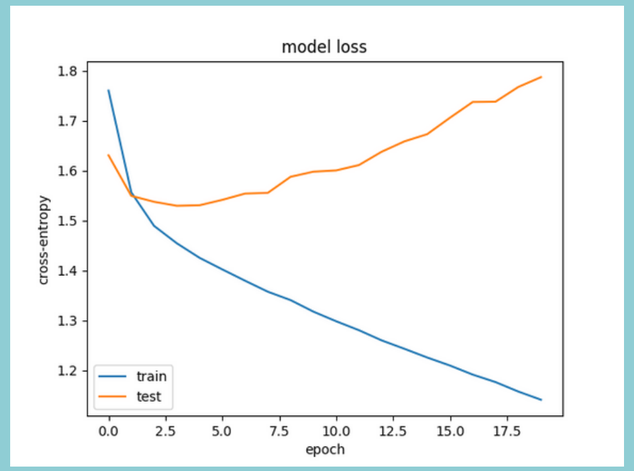


Model Architectures

The CNN LSTM architecture involves using Convolutional Neural Networks (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction. CNN LSTMs class of models are both spatially and temporally deep and have the flexibility to be applied to a variety of vision tasks involving sequential inputs and outputs.

- Vanilla LSTM
- Convolutional LSTM
-
-
-

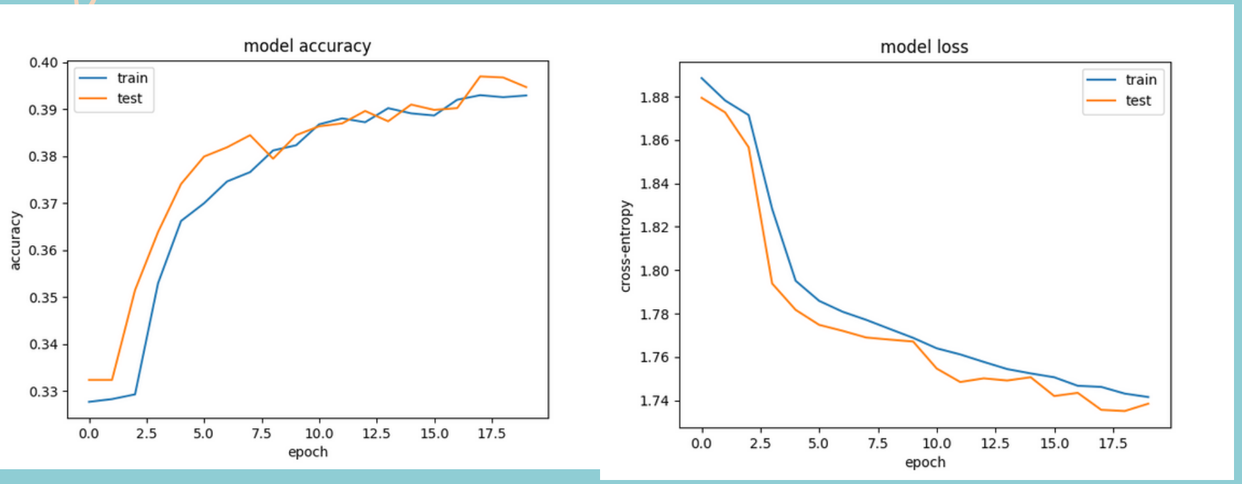




Model Architectures

Bidirectional LSTM or Bi-LSTM allows the neural network to have both backward and forward information about the sequence at every time step. Employing bidirectional will run inputs in two ways, one from past to future and one from future to past, thus preserving information from both past and future at any point in time.

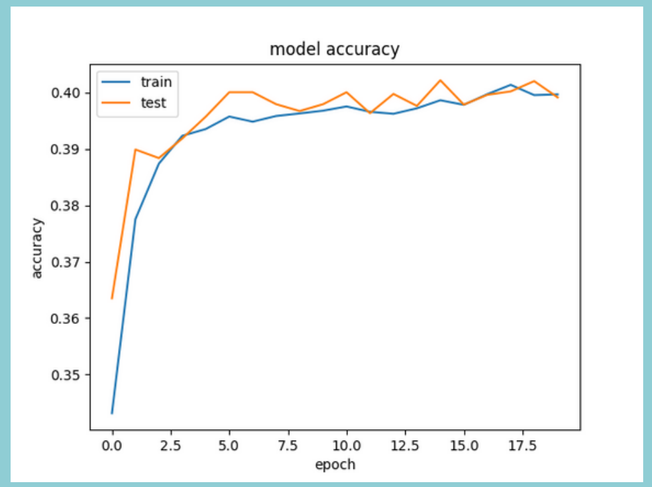
- Vanilla LSTM
- Convolutional LSTM
- Bidirectional LSTM
-
-

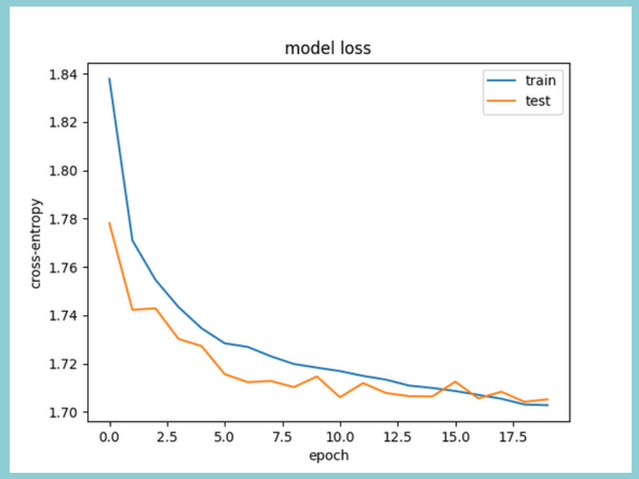


Model Architectures

LSTMs enable RNN to remember inputs over a long time as it contains information in a memory. The attention class takes in a layer and then performs the necessary functions such as initializing a keyworded variable length of arguments to the class, building the layer by adding the necessary weights and biases depending on the input shape and other functions to improve the layer, and then obtain the necessary outputs.

- Vanilla LSTM
- Convolutional LSTM
- Bidirectional LSTM
- LSTM with Attention
-





Model Architectures

We introduce an embedding layer and then pass the Bidirectional LSTM layer to the attention class to form the final attention layer.

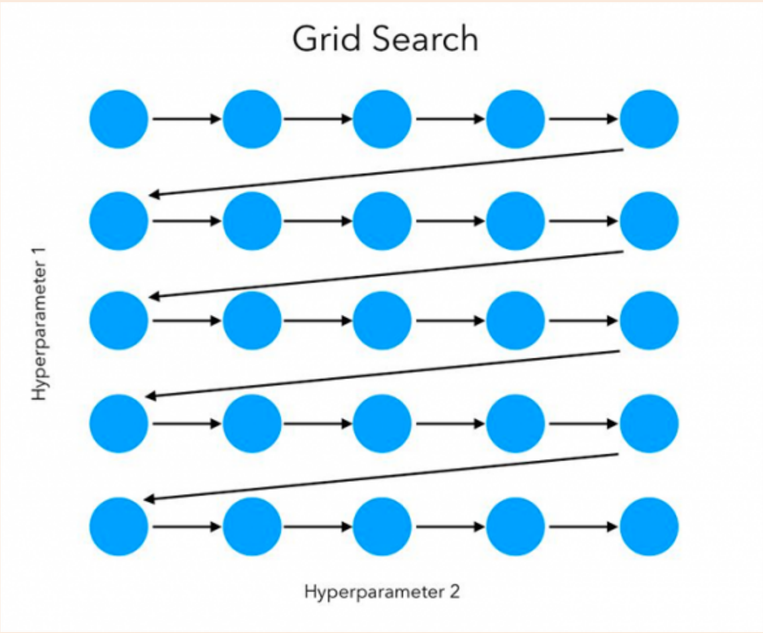
- Vanilla LSTM
- Convolutional LSTM
- Bidirectional LSTM
- LSTM with Attention
- Bidirectional with Attention

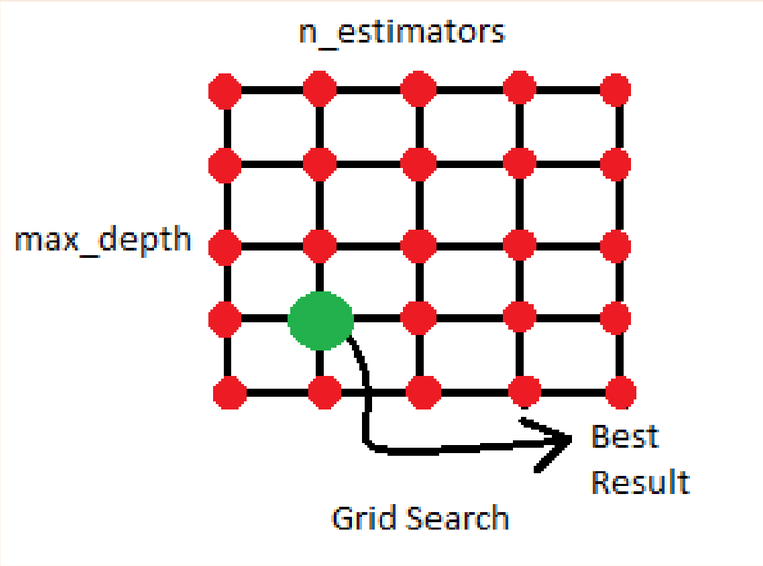
PART 3: HYPERPARAMETER TUNING

GridSearchCV | K - fold Cross Validation | Regularization

GridSearchCV

- Process of performing hyperparameter tuning in order to determine the optimal values for a given model.
- Pass predefined values for hyperparameters to the GridSearchCV function.
- GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method





GridSearchCV

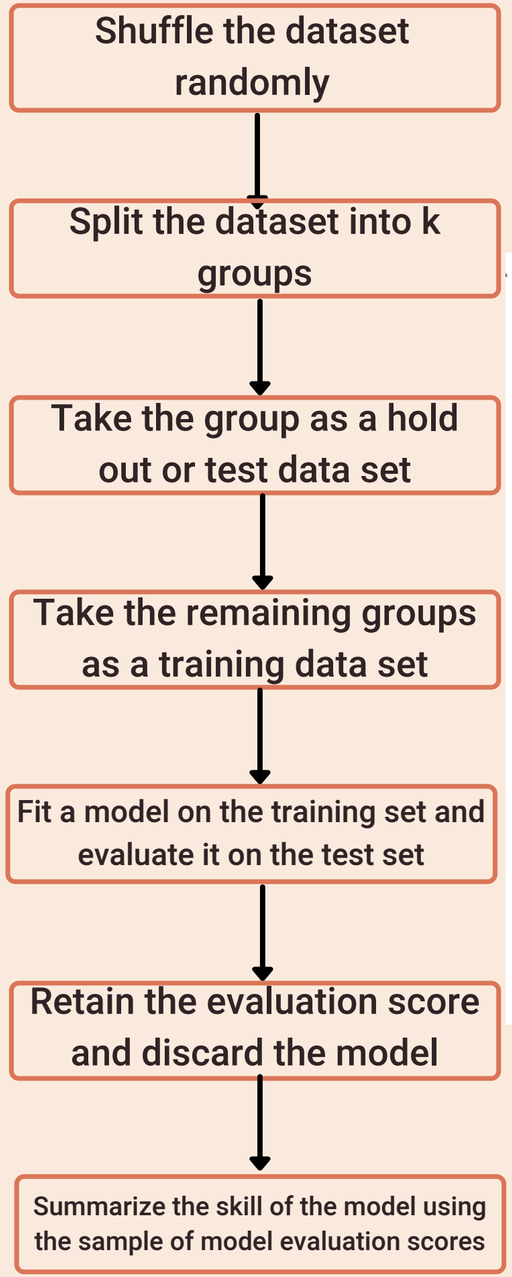
```
Best: 0.398902 using {'batch_size': 512, 'epochs': 50}
0.391135 (0.007981) with: {'batch_size': 512, 'epochs': 25}
0.398902 (0.002040) with: {'batch_size': 512, 'epochs': 50}
0.392662 (0.003586) with: {'batch_size': 512, 'epochs': 100}
0.371149 (0.004605) with: {'batch_size': 512, 'epochs': 150}
0.360945 (0.006863) with: {'batch_size': 512, 'epochs': 200}
0.398349 (0.005134) with: {'batch_size': 256, 'epochs': 25}
0.395522 (0.002901) with: {'batch_size': 256, 'epochs': 50}
0.373879 (0.003137) with: {'batch_size': 256, 'epochs': 100}
0.350254 (0.003785) with: {'batch_size': 256, 'epochs': 150}
0.330528 (0.003027) with: {'batch_size': 256, 'epochs': 200}
0.395912 (0.003288) with: {'batch_size': 128, 'epochs': 25}
```

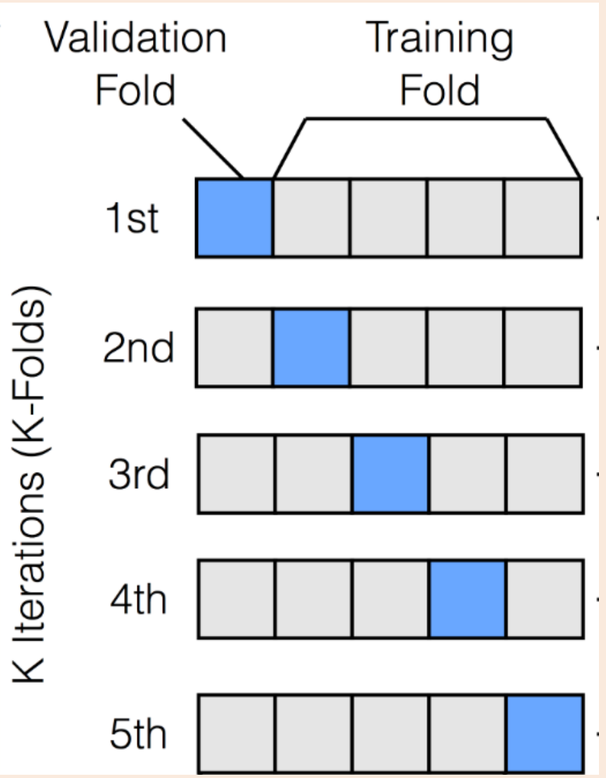
```
Best: 0.397699 using {'learn_rate': 0.001}
0.397699 (0.003676) with: {'learn_rate': 0.001}
0.377941 (0.005663) with: {'learn_rate': 0.005}
0.386228 (0.006978) with: {'learn_rate': 0.01}
0.345996 (0.015788) with: {'learn_rate': 0.03}
0.339010 (0.007537) w Best: 0.398869 using {'optimizer': 'Adam'}
0.270666 (0.082122) w 0.328253 (0.000697) with: {'optimizer': 'SGD'}
0.212860 (0.096336) w 0.389283 (0.008262) with: {'
0.327473 (0.004015) w 0.328253 (0.000697) with: {'
0.261470 (0.084287) w 0.328253 (0.000697) with: {'
0.398869 (0.001703) with: {'
0.392564 (0.004109) with: {'
0.385383 (0.011123) with: {'
```

```
Best: 0.399649 using {'dropout_rate': 0.5, 'weight_constraint': 4}
0.388990 (0.004083) with: {'dropout_rate': 0.0, 'weight_constraint': 1}
0.391590 (0.003208) with: {'dropout_rate': 0.0, 'weight_constraint': 2}
0.391947 (0.002021) with: {'dropout_rate': 0.0, 'weight_constraint': 3}
0.393702 (0.003955) with: {'dropout_rate': 0.0, 'weight_constraint': 4}
0.392630 (0.002987) with: {'dropout_rate': 0.0, 'weight_constraint': 5}
0.394742 (0.004233) with: {'dropout_rate': 0.1, 'weight_constraint': 1}
0.397764 (0.002848) with: {'dropout_rate': 0.1, 'weight_constraint': 2}
0.397504 (0.003854) with: {'dropout_rate': 0.1, 'weight_constraint': 3}
0.397829 (0.000186) with: {'dropout_rate': 0.1, 'weight_constraint': 4}
0.395912 (0.001457) with: {'dropout_rate': 0.1, 'weight_constraint': 5}
0.397309 (0.002565) with: {'dropout_rate': 0.2, 'weight_constraint': 1}
0.395229 (0.001156) with: {'dropout_rate': 0.2, 'weight_constraint': 2}
0.398089 (0.002239) with: {'dropout_rate': 0.2, 'weight_constraint': 3}
```

K-Fold Cross Validation

- Resampling procedure used to evaluate machine learning models on a limited data sample
- Has a single parameter called k that refers to the number of groups that a given data sample is to be split into
- Generally results in a less biased or less optimistic estimate of the model skill





Regularization

- Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.
- A form of regression that shrinks the coefficient estimates towards zero.
- Works by adding a penalty or complexity term or shrinkage term with Residual Sum of Squares (RSS) to the complex model.
- Bayesian Ridge has been employed
- Introduce a small amount of bias, known as Ridge regression penalty so that we can get better long-term predictions (L2)

L1 Regularization

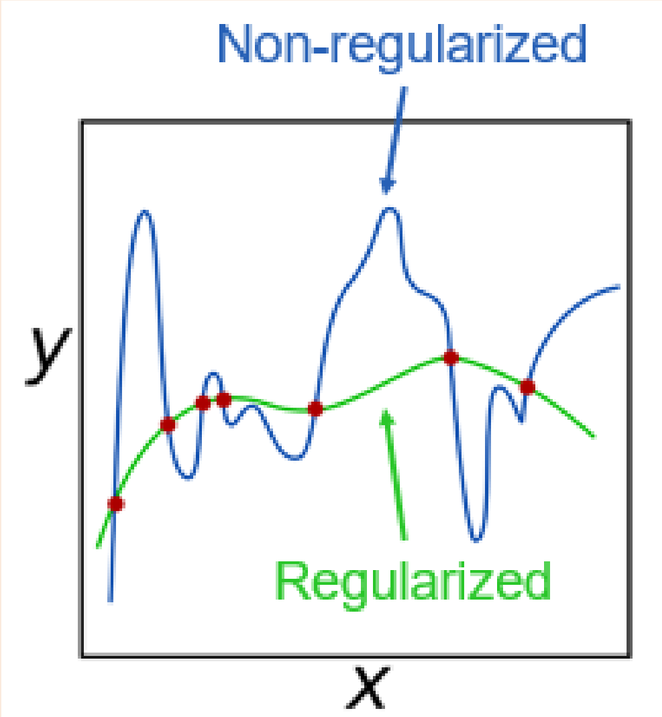
$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} W_j^2$$

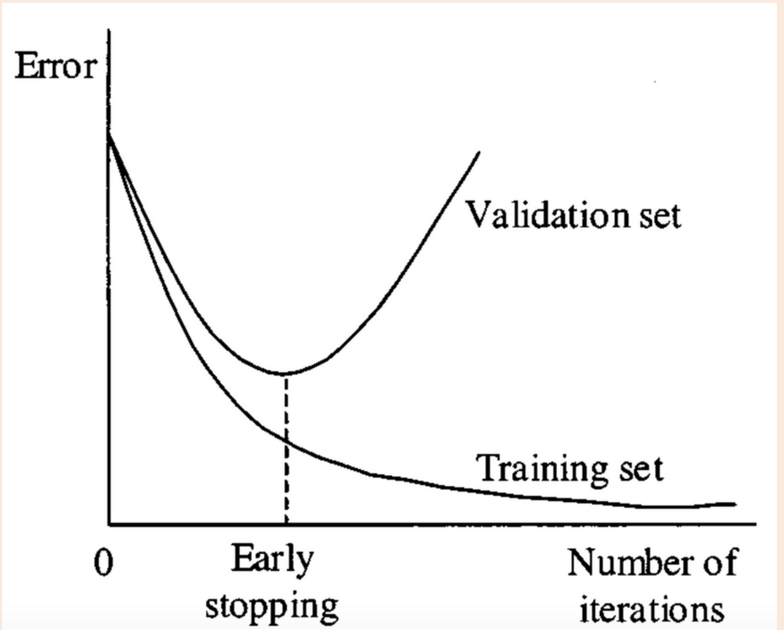
Loss function

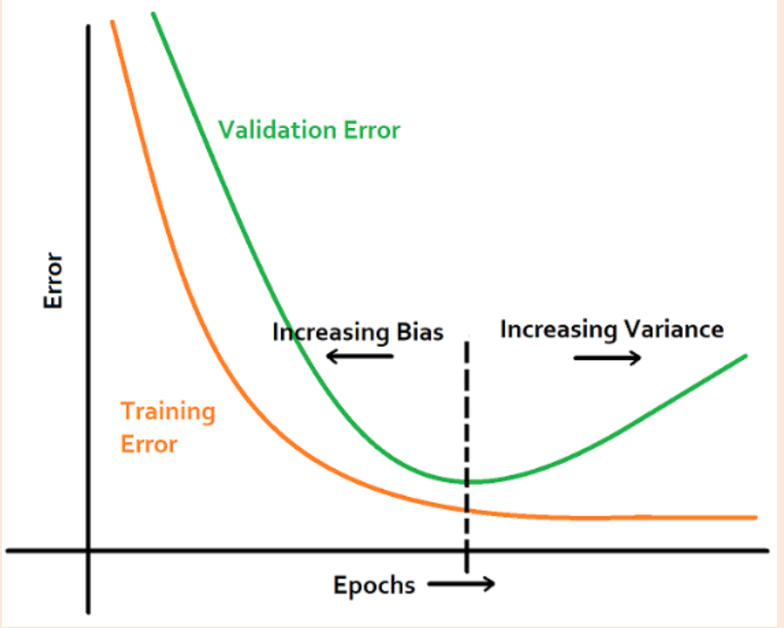
Regularization Term



Early Stopping

- Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset
- Keras supports the early stopping of training via a callback called EarlyStopping
- Specifies the performance measure to monitor, the trigger, and once triggered, it will stop the training process

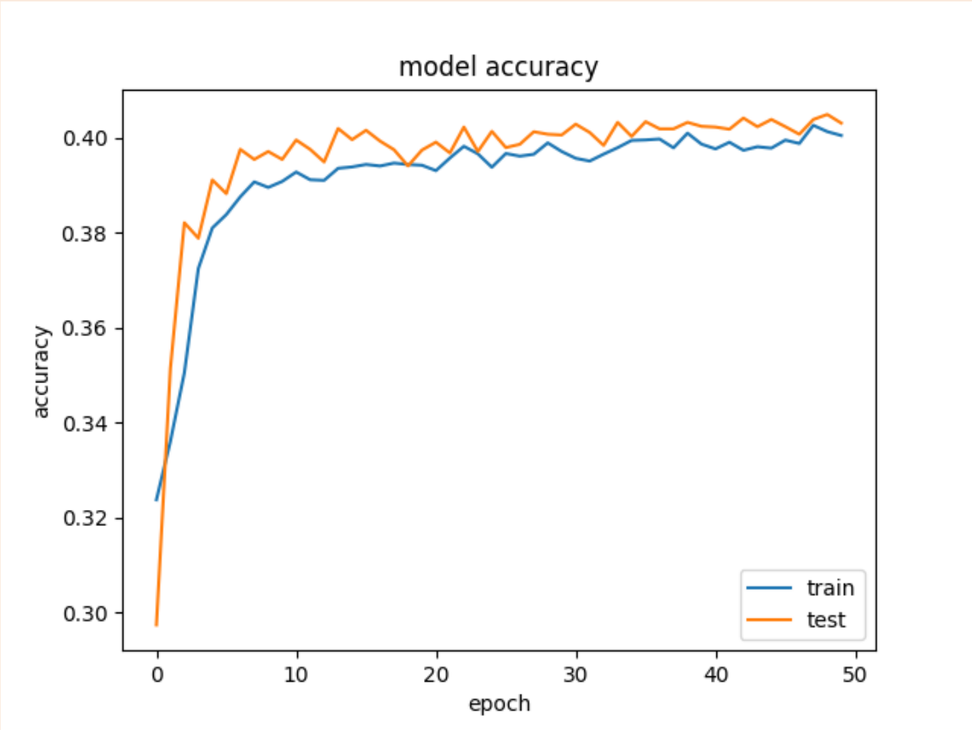




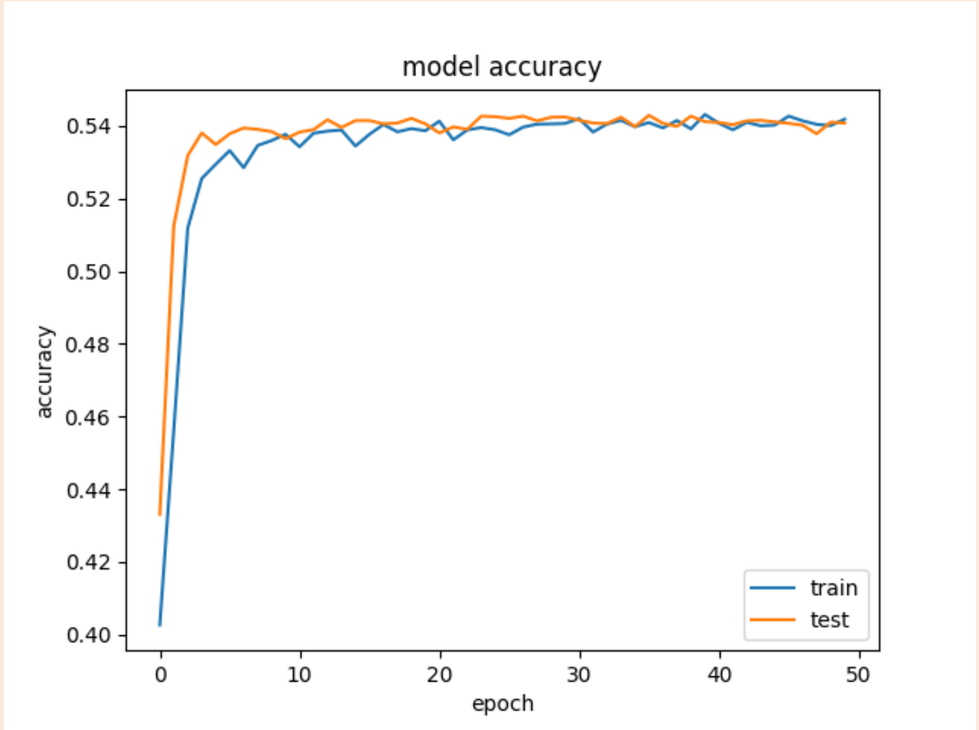
PART 4: FINAL MODEL

Performance Comparison | Best model discussion

Bidirectional LSTM



9 labels

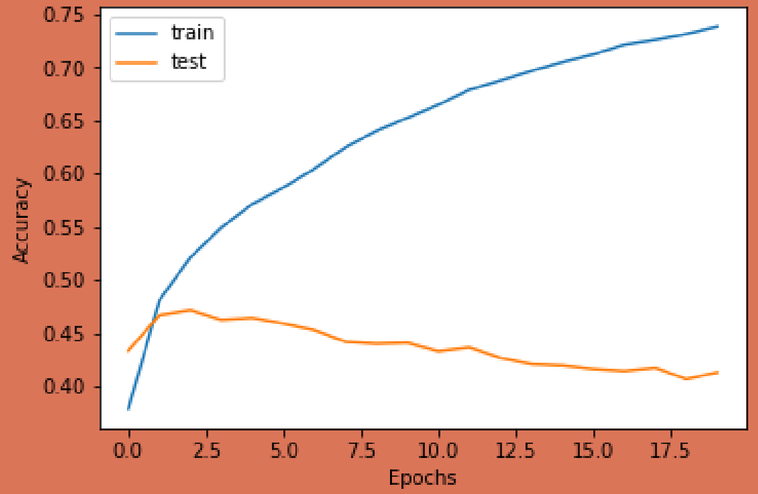


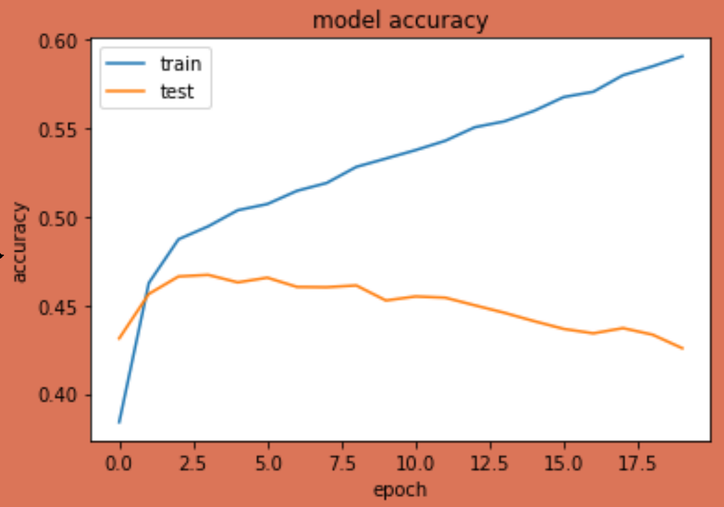
3 labels

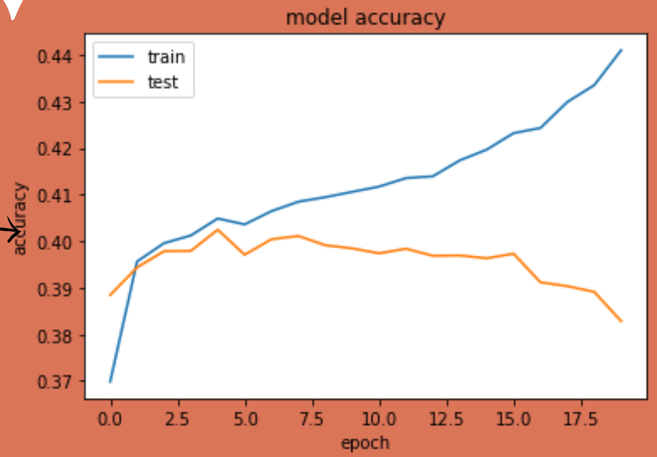
PART 5: RESULT DISCUSSION

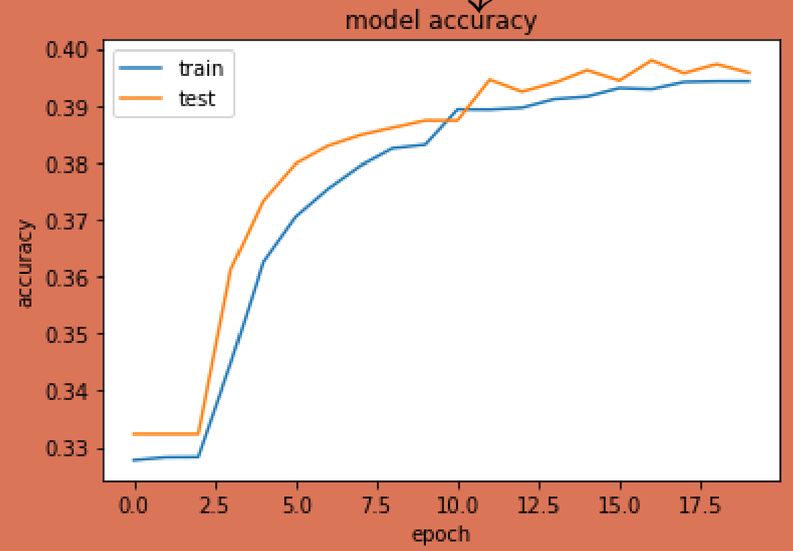
Accuracy Scores Achieved | Lessons Learnt

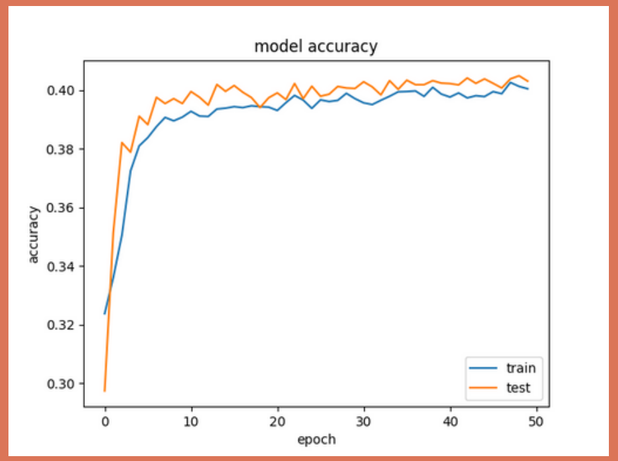
Model Progression

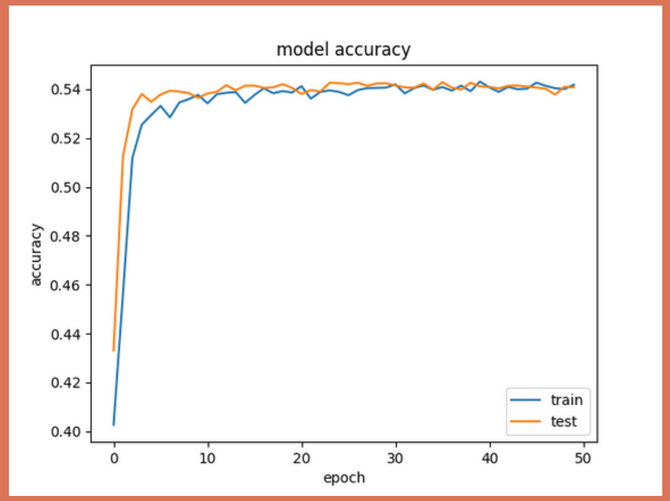






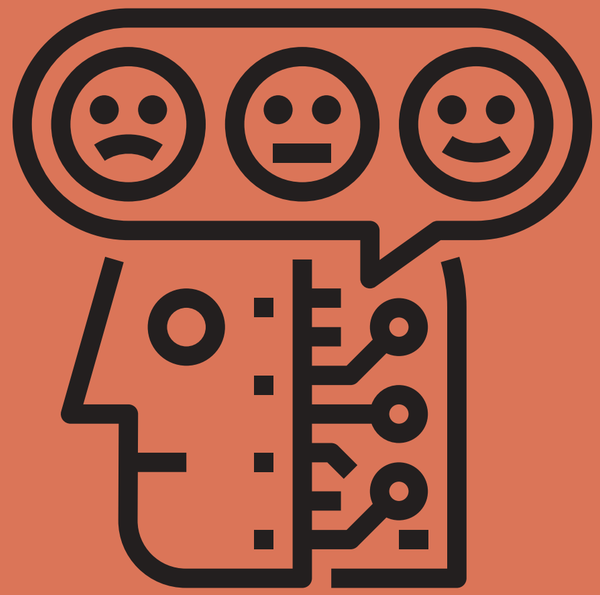


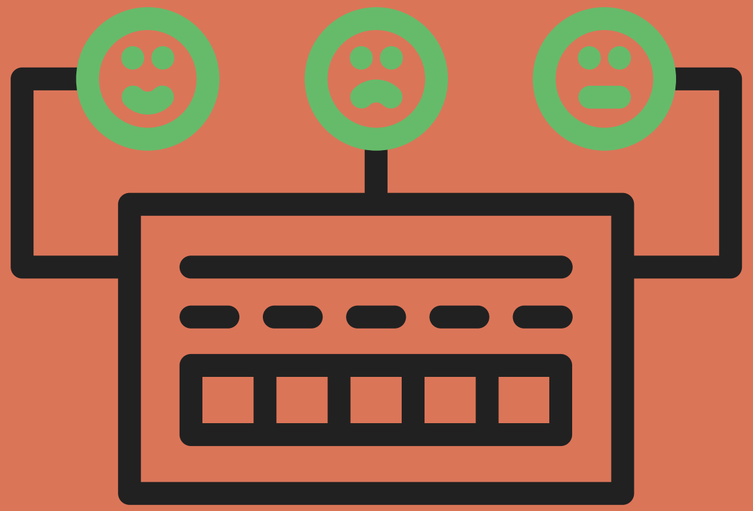




Lessons Learnt

- Importance of dataset cleaning and pre - processing of data
- Problem of overfitting
- Importance of good quanitity of data
- Types of Neural Network Architecutres
- Hyperparameter tuning techniques



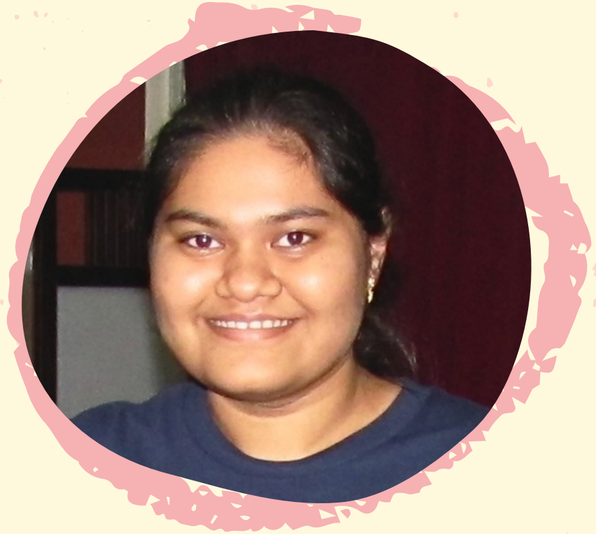


PART 6: CONCLUSION

Meet the Team| Key Takeaways + Learning

Introducing the team

Abhinaya



Aishwarya

Kartikeya

Abhinaya

- Feature Engineering
- Types of layers
- Solving overfitting
- Hyperparameter tuning
- Using GPU Clusters

Aishwarya

* Model Progression
* Importance of EDA
* Working of LSTM
* Hyperparameter Tuning
* Result Analysis

Kartikeya

* Learning about LSTM
* Types of LSTM
* Data Preparation
* Relevant features
* Model Architectures

Thank you