MALIGNANT COMMENT CLASSFICATION PROJECT

BY: LAKSHITA KAIN
DATA SCIENCE(INTERN)
FLIPROBO TECHNOLOGIES

INTRODUCTION

Internet offers a global ecosystem for social interactions. Modern life revolves around the network, with its status updates, news feeds, comment chains, political advocacy, omnipresent reviews, rankings and ratings.

People have freedom of speech while keeping their anonymity. Someone posts something, and people start commenting negative feedbacks which is far from constructive, it's demotivating and lead to discouragement. Sometimes, it's plain discrimination on the basis of body(body-shaming), social background or just on the basis of race or community. The people who relish this online freedom are called trolls, a term that originally came from a fishing method online thieves use to find victims.

A Pew Research Center survey published found that 70% of 18-to-24-year-olds who use the Internet had experienced harassment, and 26% of women that age said they'd been stalked online. A 2014 study published in the psychology journal Personality and Individual Differences found that the approximately 5% of Internet users who self-identified as trolls scored extremely high in the dark tetrad of personality traits: narcissism, psychopathy, Machiavellianism and, especially, sadism. The pandemic and indirectly the sudden increase in unemployment have made things even worse. Trolls have more time to hide behind the screens and bring people down.

Sentiment Analysis is the task of analyzing people's opinions in textual data (e.g., product reviews, movie reviews, or tweets), and extracting their polarity and viewpoint. The task can be cast as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities.

PROBLEM STATEMENT

The goal of this project is to use deep learning techniques to identify the level toxicity of a comment which could be used to help deter users from posting potentially hurtful comments, engage in more sophisticated arguments and make internet a safer place.

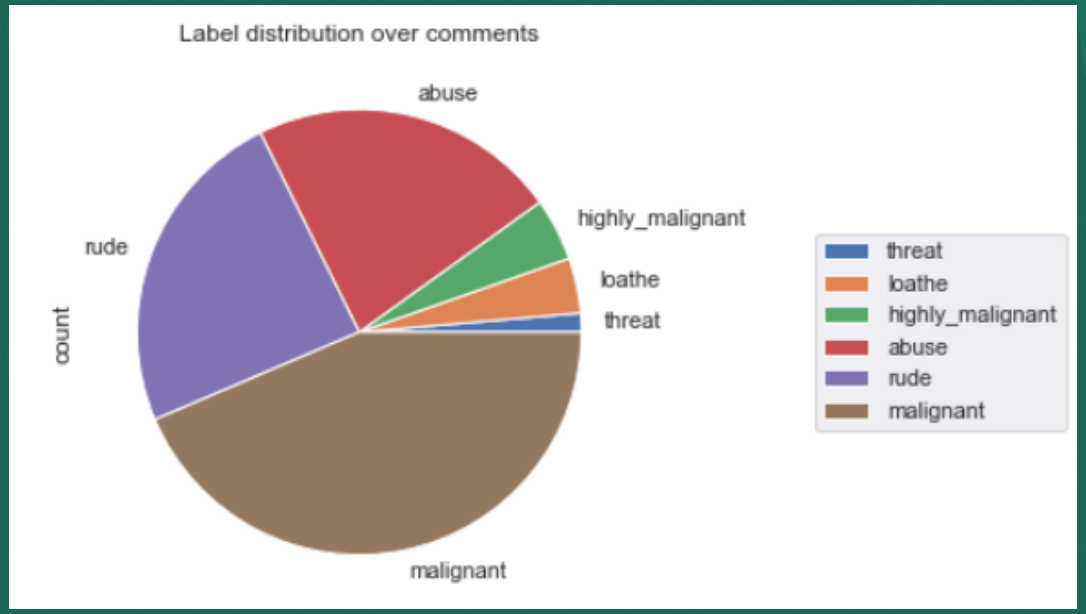
Our dataset consists of 6 labels namely highly malignant, malignant, abuse, threat, loathe and rude. This project aims to implement various Machine Learning algorithms and deep learning algorithms like Multilayer perceptron(MLP), Long Short Term Memory Networks, Multinomial Naïve Baiyes, Logistic Regression, Random Forest Classifier , Linear SVC and Adaptive Boosting.

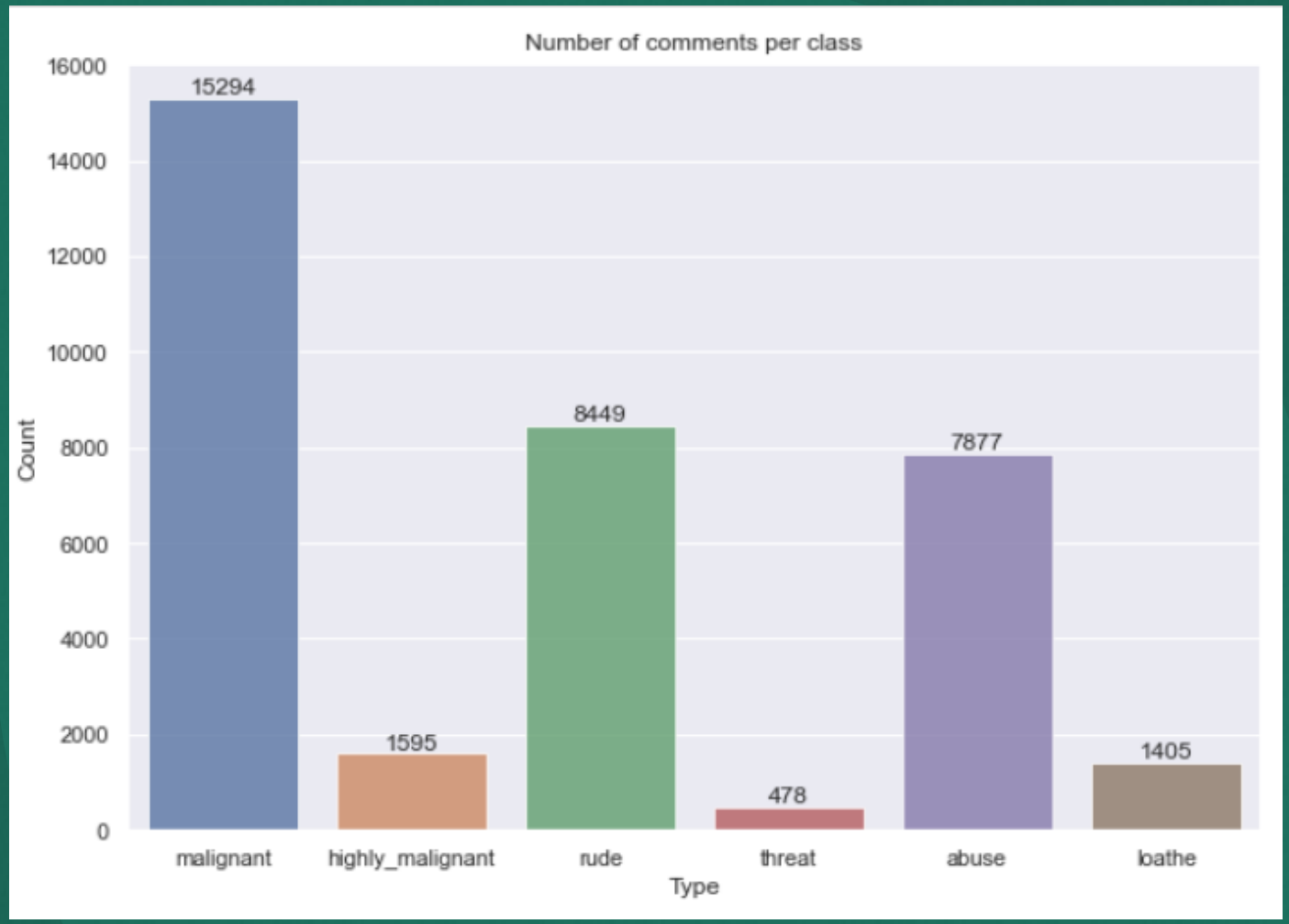
EXPLORATORY DATA ANALYSIS

- Shape of our train csv : records - `159571` , `features - 8`
- Shape of our train csv : records - `153164` , `features - 2`
- There are no null values
- There are no duplicated fields or attributes.

Target Labels	Number of Occurrences in our train dataset
Malignant	15294
Highly Malignant	1595
Rude	8449
Threat	478
Abuse	7877
Loathe	1405

Labels Data Distribution





Statistical Description of our Labels

Correlation Matrix

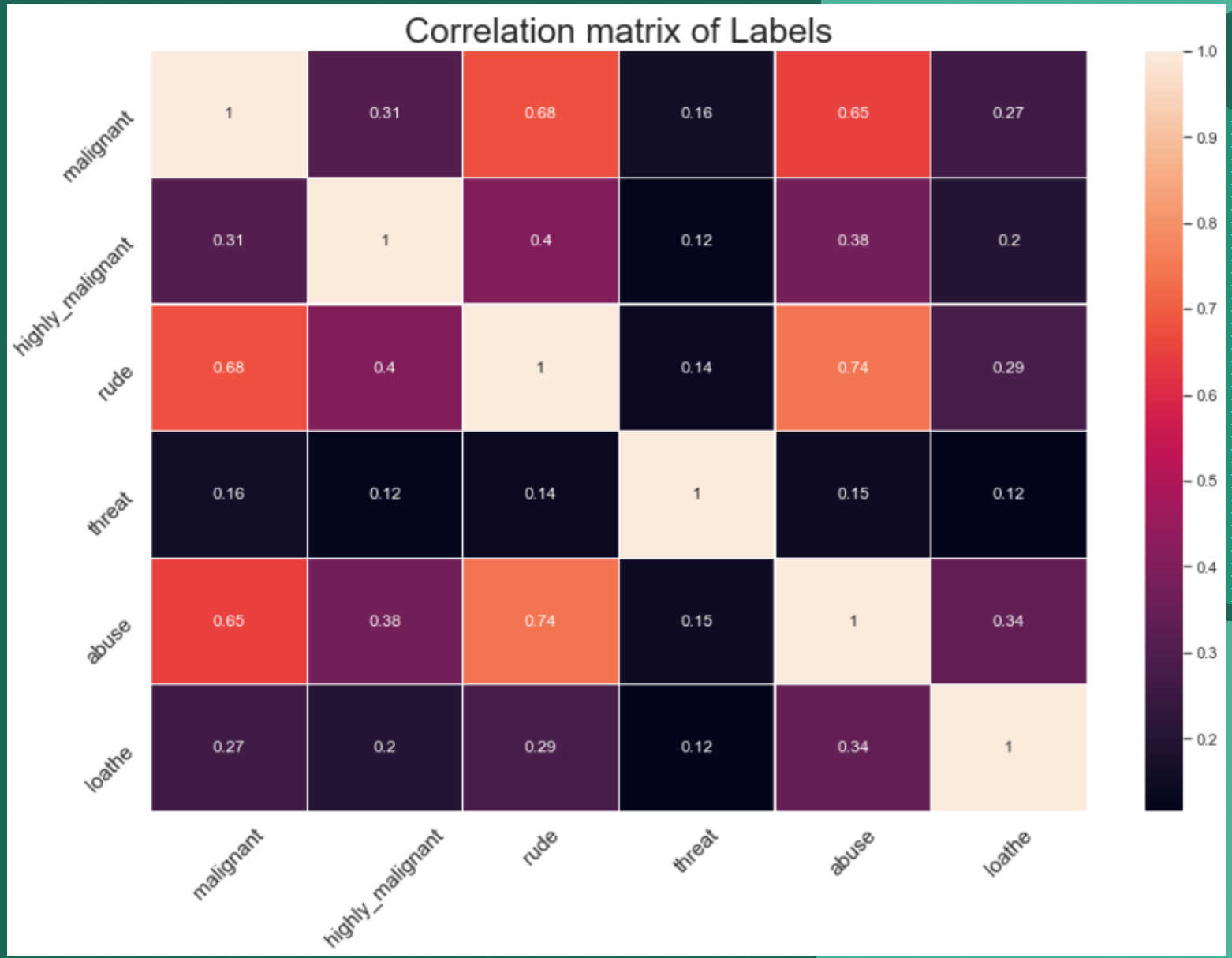


Table of content

Data Normalization

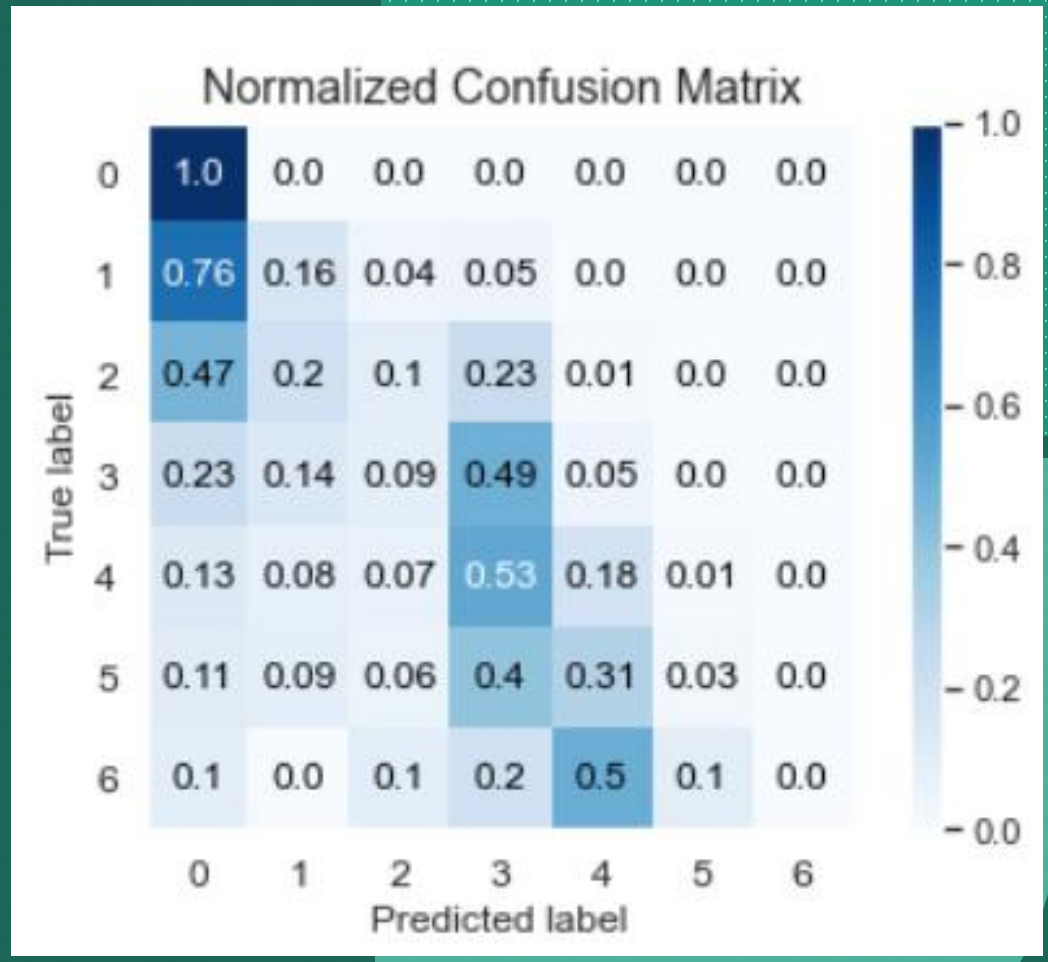
- Lower casing of all the sentences.
- Tokenization : converting sentences into separate word tokens.
- Removal of numerics, special characters, symbols etc.
- Removal of stopwords from english (I, they, we, are , they, a , so etc)
- Lematization: reduces the inflected words properly ensuring that the root word belongs to english language.
- TFID vectorization : provide adequate weight to a word in proportion of the impact it has on the meaning of a sentence. The score is a product of 2 independent scores, term frequency(tf) and inverse document frequency (idf)
- Embedding Padding : In our LSTM model pre-padding is used. “embedded_docs” is the input list of sentences which is one hot encoded and every sentence is made of the same length.

Logistic Regression

Accuracy Score : 91.82%
Hamming Loss : 0.08175
Cross – Validation Score : 91.96%

Classification Logistic Regression :

	precision	recall	f1-score	support
0	0.95	1.00	0.97	42950
1	0.36	0.16	0.22	1947
2	0.30	0.10	0.15	1051
3	0.49	0.49	0.49	1275
4	0.43	0.18	0.25	527
5	0.43	0.03	0.05	112
6	0.00	0.00	0.00	10
accuracy			0.92	47872
macro avg	0.42	0.28	0.30	47872
weighted avg	0.89	0.92	0.90	47872

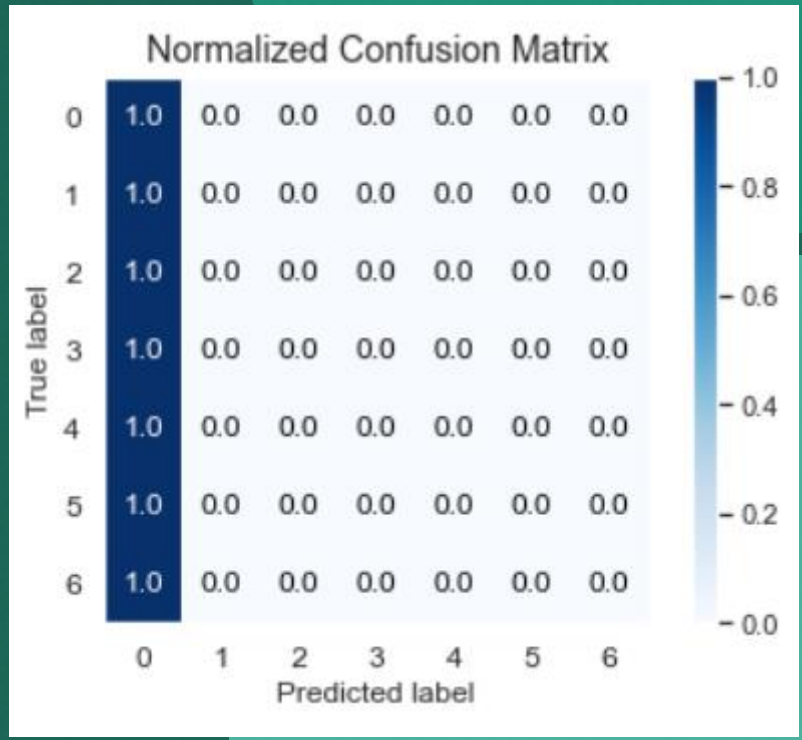


Random Forest Classifier

- Accuracy Score : 89.72%
- Hamming Loss : 0.1028
- Cross – Validation Score : 89.83%

Classification Random Forest Classifier :

	precision	recall	f1-score	support
0	0.90	1.00	0.95	42950
1	0.00	0.00	0.00	1947
2	0.00	0.00	0.00	1051
3	0.00	0.00	0.00	1275
4	0.00	0.00	0.00	527
5	0.00	0.00	0.00	112
6	0.00	0.00	0.00	10
accuracy			0.90	47872
macro avg	0.13	0.14	0.14	47872
weighted avg	0.80	0.90	0.85	47872

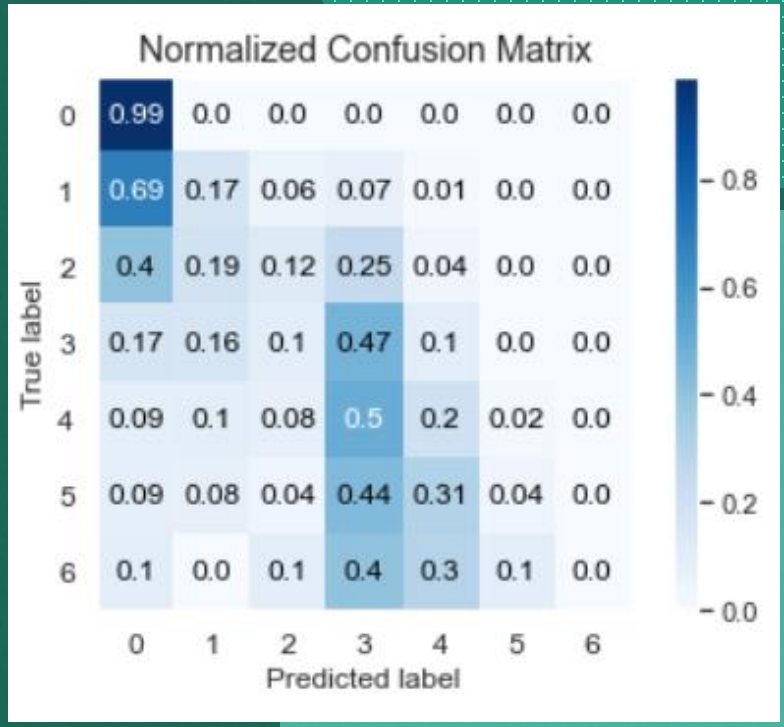


Linear SVC

- Accuracy Score : 91.67%
- Hamming Loss : 0.0833
- Cross – Validation Score : 91.9%

Classification MLinear SVC :

	precision	recall	f1-score	support
0	0.95	0.99	0.97	42950
1	0.34	0.17	0.23	1947
2	0.27	0.12	0.16	1051
3	0.45	0.47	0.46	1275
4	0.32	0.20	0.25	527
5	0.15	0.04	0.06	112
6	0.00	0.00	0.00	10
accuracy			0.92	47872
macro avg	0.36	0.28	0.30	47872
weighted avg	0.89	0.92	0.90	47872

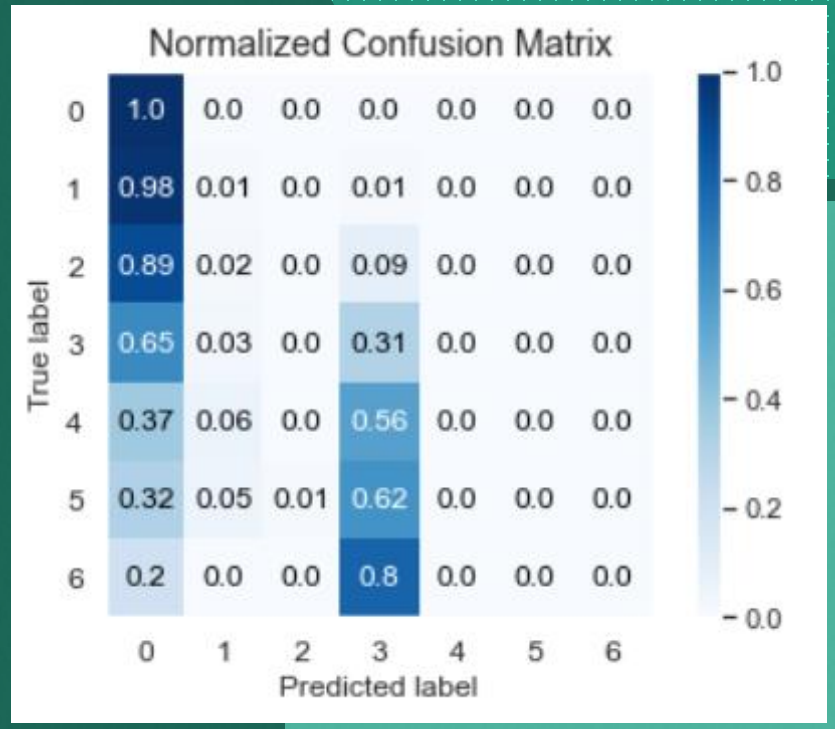


Multinomial Naive Bayes

- Accuracy Score : 90.57%
- Hamming Loss : 0.1028
- Cross – Validation Score : 90.81%

Classification Multinomial Naive Bayes Classifier :

	precision	recall	f1-score	support
0	0.92	1.00	0.96	42950
1	0.11	0.01	0.01	1947
2	0.22	0.00	0.00	1051
3	0.45	0.31	0.37	1275
4	0.50	0.00	0.01	527
5	0.00	0.00	0.00	112
6	0.00	0.00	0.00	10
accuracy			0.91	47872
macro avg	0.31	0.19	0.19	47872
weighted avg	0.85	0.91	0.87	47872

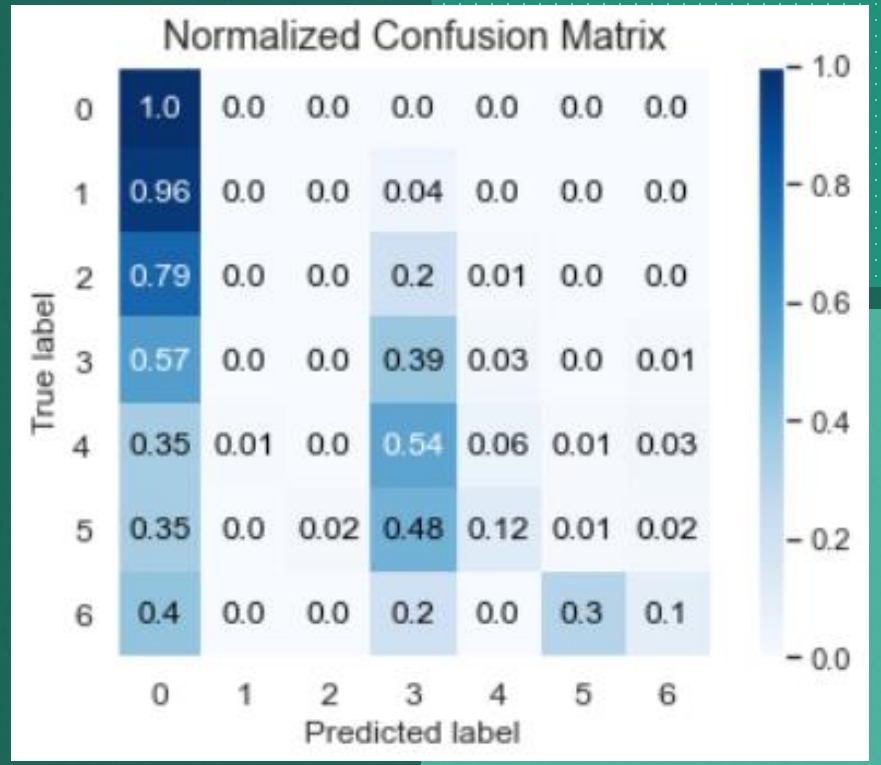


Adaptive Boosting

- Accuracy Score : 90.71%
- Hamming Loss : 0.09289
- Cross – Validation Score : 90.79%

Classification Adaptive Boost Classifier :

	precision	recall	f1-score	support
0	0.92	1.00	0.96	42950
1	0.18	0.00	0.01	1947
2	0.00	0.00	0.00	1051
3	0.42	0.39	0.41	1275
4	0.33	0.06	0.10	527
5	0.08	0.01	0.02	112
6	0.03	0.10	0.05	10
accuracy			0.91	47872
macro avg	0.28	0.22	0.22	47872
weighted avg	0.85	0.91	0.87	47872



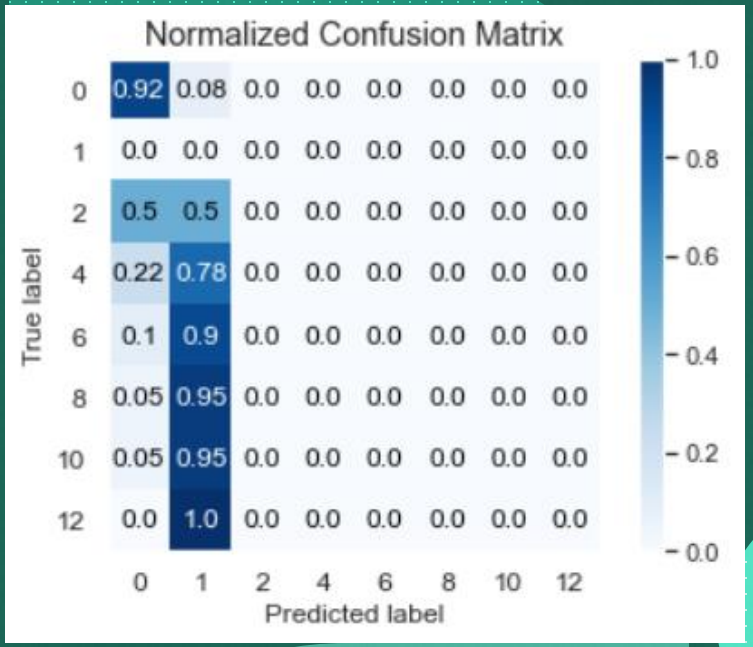
Long Short Term Memory

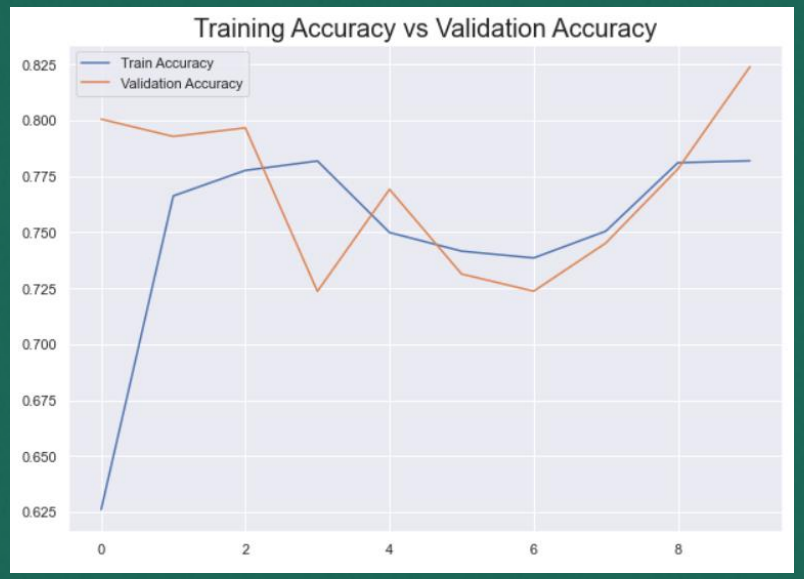
Tradition neural networks suffer from short-term memory. Also, a big drawback is the vanishing gradient problem. (While backpropagation the gradient becomes so small that it tends to 0 and such a neuron is of no use in further processing.) LSTMs efficiently improves performance by memorizing the relevant information that is important and finds the pattern.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 40)            200000
_________________________________________________________________
lstm (LSTM)                  (None, 100)               56400
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 256,501
Trainable params: 256,501
Non-trainable params: 0
_________________________________________________________________
None
```

Long Short Term Memory

Accuracy Score : 82.39%
Hamming Loss : 0.1016





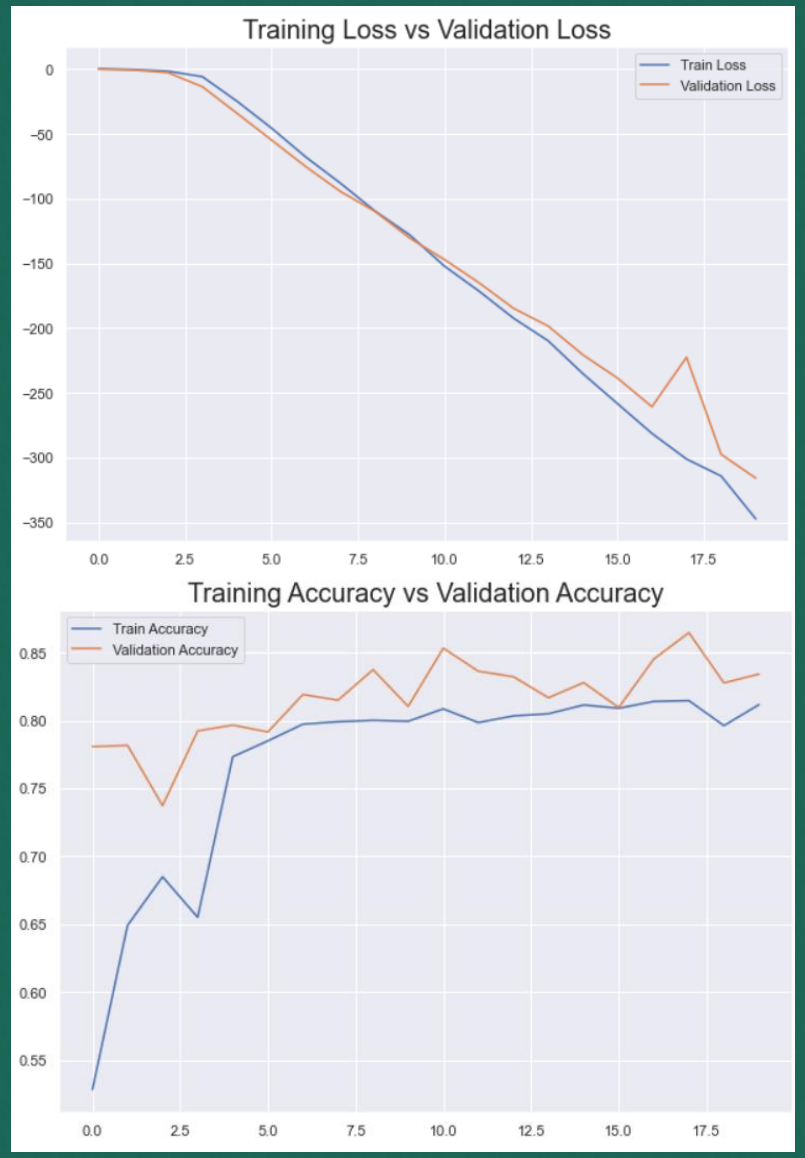
	precision	recall	f1-score	support
0	0.90	1.00	0.95	43005
2	0.00	0.00	0.00	1870
4	0.00	0.00	0.00	1067
6	0.00	0.00	0.00	1272
8	0.00	0.00	0.00	533
10	0.00	0.00	0.00	121
12	0.00	0.00	0.00	4
accuracy			0.90	47872
macro avg	0.13	0.14	0.14	47872
weighted avg	0.81	0.90	0.85	47872

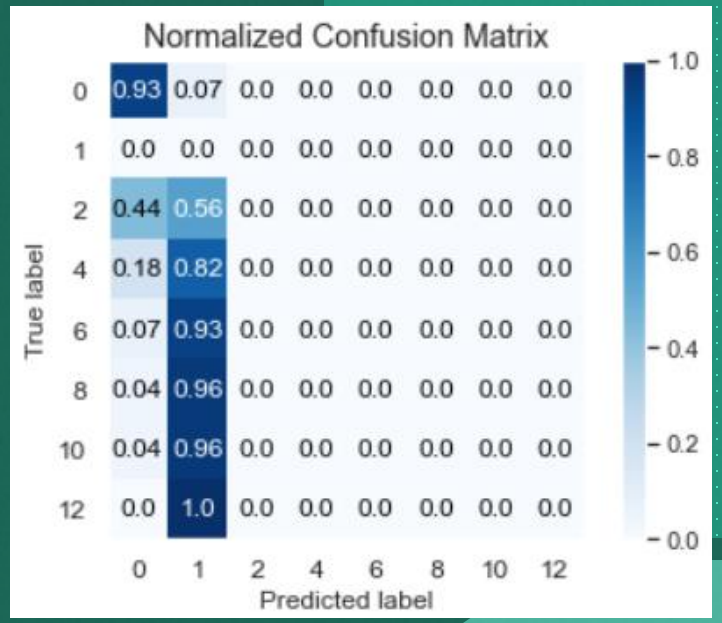
LSTM With Dropout

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 50, 40)            200000
_________________________________________________________________
dropout (Dropout)            (None, 50, 40)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               56400
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 256,501
Trainable params: 256,501
Non-trainable params: 0
_________________________________________________________________
None
```

- Accuracy Score : 89.83%
- Hamming Loss : 0.1657

LSTM With Dropout



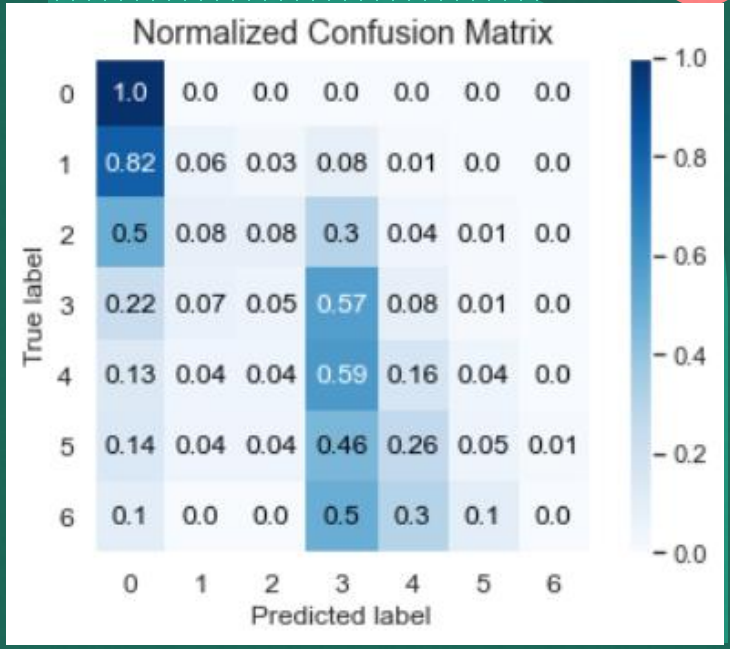


	precision	recall	f1-score	support
0	0.90	1.00	0.95	43005
2	0.00	0.00	0.00	1870
4	0.00	0.00	0.00	1067
6	0.00	0.00	0.00	1272
8	0.00	0.00	0.00	533
10	0.00	0.00	0.00	121
12	0.00	0.00	0.00	4
accuracy			0.90	47872
macro avg	0.13	0.14	0.14	47872
weighted avg	0.81	0.90	0.85	47872

	Prediction Accuracy (%)	Hamming Loss	Cross - Validation Score (%)
Logistic Regression	91.82	0.0817	91.96
Random Forest Classifier	89.72	0.1028	89.83
Linear SVC	91.67	0.0833	91.9
Multinomial NB	90.57	0.1028	90.81
Adaptive Boosting Regressor	90.71	0.0920	90.79
LSTM	82.39	0.1016	-
LSTM with dropout	89.83	0.1657	-

Hyper Parameter Tuning of Linear SVC

```
params = {'penalty': ['l1','l2'],
          'loss':['hinge','squared_hinge'],
          'max_iter':[1000,2000,3000],
          'multi_class': ['ovr','crammer_singer'],
          'fit_intercept' : ['True', 'False'],
          'random_state': [10,20,50,82,4]}
```



Accuracy Score : 91.67%
Hamming Loss : 0.0831

```
{'random_state': 20,
 'penalty': 'l2',
 'multi_class': 'ovr',
 'max_iter': 1000,
 'loss': 'hinge',
 'fit_intercept': 'True'}
```

Classification Linear SVC model Classifier :

	precision	recall	f1-score	support
0	0.95	1.00	0.97	42950
1	0.34	0.06	0.10	1947
2	0.34	0.08	0.12	1051
3	0.46	0.57	0.51	1275
4	0.29	0.16	0.20	527
5	0.12	0.05	0.07	112
6	0.00	0.00	0.00	10
accuracy			0.92	47872
macro avg	0.36	0.27	0.28	47872
weighted avg	0.88	0.92	0.89	47872

Thank You!


mirror_mod.use_x
mirror_mod.use_y
mirror_mod.use_z
elif operation ==
mirror_mod.use_x
mirror_mod.use_y
#selection at the end add back the deselected mirror
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the