VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

J Component Report

Programme : M. Tech Integrated in CSE with Specialization in Business Analytics

Course Title : Social Media Analytics

Course Code : CSE4069

Slot : D1

TITLE

Twitter Hate Speech and Offensive Language Detection

Team Members:

19MIA1039 | PBSS Jaswanth

19MIA1074 | S. Vamsidhar

19MIA1094 | Mansoor Khan Lodi

19MIA1105 | Shashank R

Faculty: Dr. Priyadarshini

Sign:
Date:

INDEX

ACKNOWLEDGMENT

Primarily, we would like to thank the almighty for all the blessings he showered over us to complete this project without any flaws. The success and final outcome of this assignment required a lot of guidance and assistance from many people, and we are extremely fortunate to have gotthis all along with the completion of our project. Whatever we have doneis only due to such guidance and assistance from our faculty, Dr. Priyadarshini, to whom we are thankful for giving us an opportunity to do this project. Last but not least, we are grateful to all our fellow classmatesand our friends for the suggestions and support given to us throughout the completion of our project.

ABSTRACT

In this project, we aim to develop a hate speech detector to classify tweets on social media platform Twitter. The spread of hate speech online has become a major concern in recent years, and social media platforms such as Twitter have been identified as a major arena for the spread of such harmful content. The use of paralinguistic signals and poorly written text in social media posts makes the task of detecting hate speech more difficult. The proposed hate speech detector will be implemented using the Python programming language and will be trained to identify and classify tweets as hate speech or not. The goal of this project is to contribute to the efforts to combat the spread of hate speech online and help create a safer and more inclusive environment on social media platforms

INTRODUCTION

This project is focused on addressing the pressing issue of toxic and abusive language on social media, specifically on the widely used platform, Twitter. With a user base of over 372 million individuals worldwide, Twitter has become a significant arena for the spread of hate speech and offensive language. As the dependence on technology for communication continues to increase, the use of social media platforms such as Twitter is at an all-time high. However, the exponential growth in usage has also brought about a rise in the spread of toxic and abusive language. This not only harms individuals who become victims of such language but also disrupts the harmony of any community.

The aim of this project is to develop a robust method to classify tweets as toxic or non-toxic, using advanced techniques in natural language processing and machine learning. By providing a tool that can automatically identify and flag toxic language, this project aims to make social media a safer and more inclusive space for all users. The ultimate goal is to create a positive impact on society by reducing the spread of hate speech and offensive language on the internet.

ABOUT THE DATASET

For our project, we have used two datasets. They are:

1) Hate Speech and Offensive Language Dataset
2) Twitter Sentiment Analysis

Here's some brief information of the datasets:

Hate Speech and Offensive Language Dataset:

This dataset is based on tweets circulated among an organization named CrowdFlower where the tweets were tweeted by the employees belonging to that organization.

Here are the features of the dataset:

Index - Index

Count - Number of CrowdFlower (CF) employees who tweeted each tweet.

Hate_speech - Number of CF employees who judged a tweet to be hate speech.

Offensive_language - Number of CF employees who judged a tweet to be offensive.

Neither - Number of CF employees who judged a tweet to be neither offensive nor non-offensive

Class - Class label for majority of CF employees. 0 corresponds to hate speech, 1 corresponds to offensive language and 2 corresponds to neither.

Text Tweet – The text content of a tweet

Dataset Link: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset

Twitter Sentiment Analysis:

This dataset is regarding a set of tweets and their class labels whether they are racist/sexual or not.

Here are the features of the dataset:

Id – Id assigned to the tweet

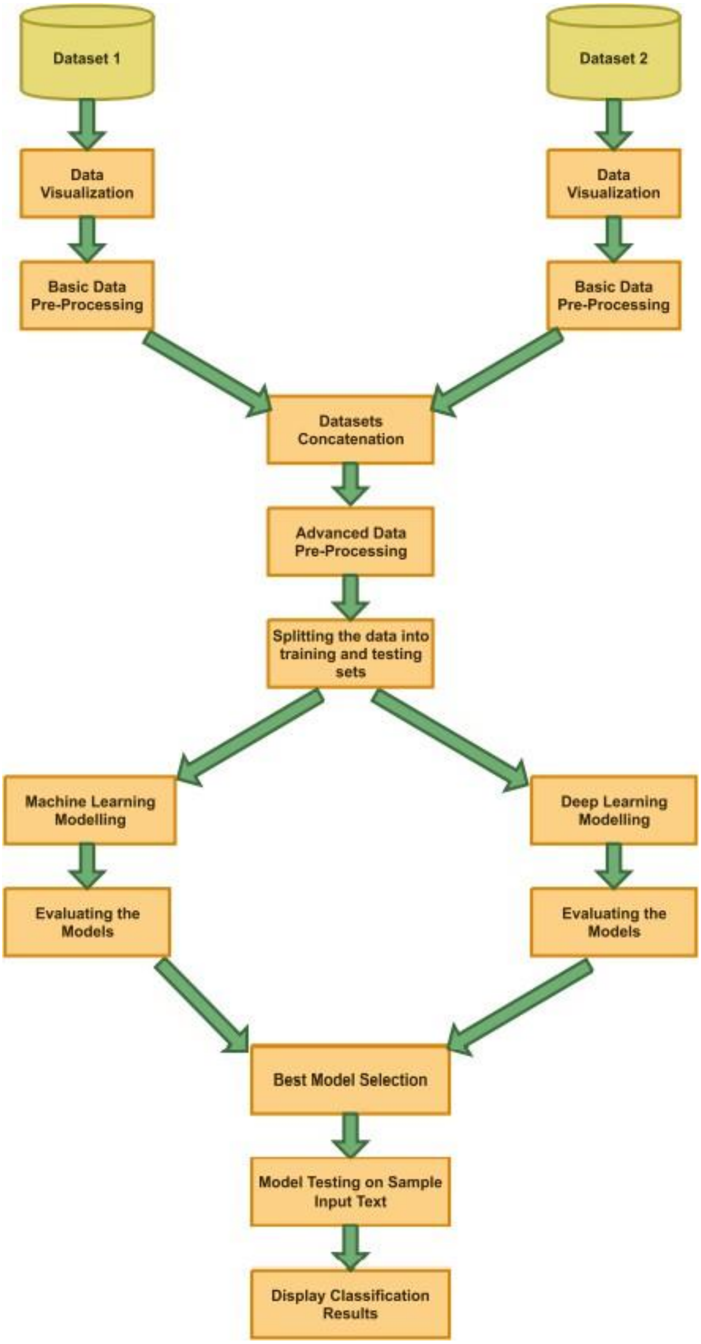
Label – Class label for the tweet

Tweet – The text content of the tweet

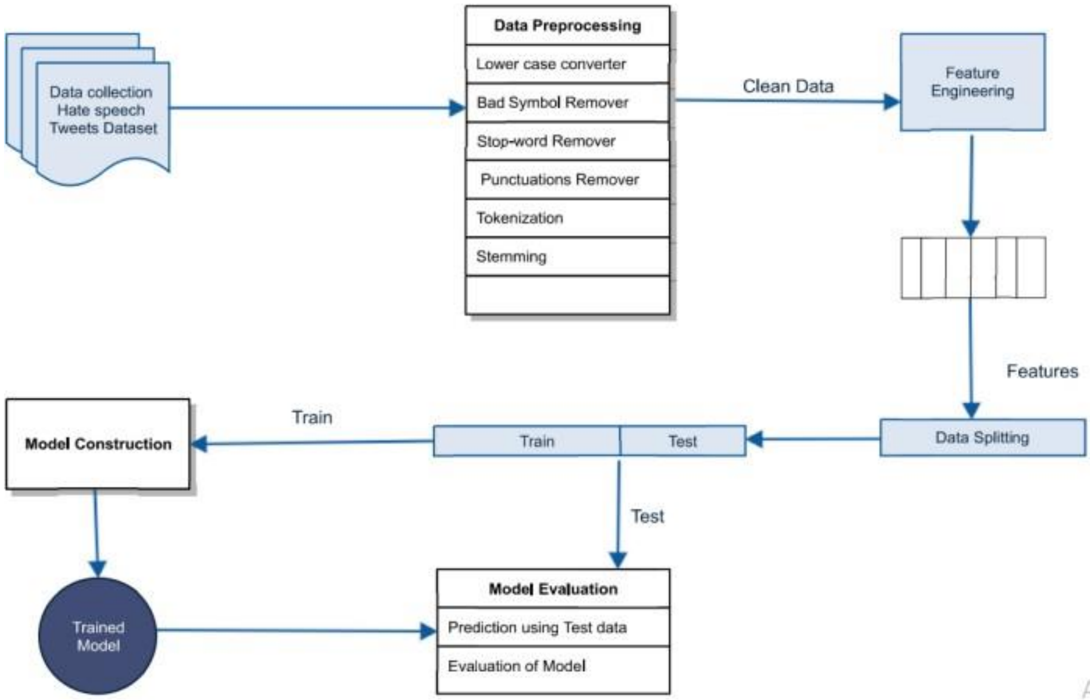
Dataset Link: https://www.kaggle.com/datasets/arkhoshghalb/twitter-sentiment-analysis-hatred-speech

DIAGRAMS CONCERNING THE PROJECT

Block Diagram:



Class Diagram:



Acti

WORKFLOW OF THE PROJECT

We take two different types of datasets and perform basic data visualization concerning the output labels and then concatenate them using basic data preprocessing techniques.

With the merged dataset in hand, we then perform advanced data preprocessing techniques in order to extract features from text i.e., splitting each and every word present in the text by removing certain unnecessary characters. We further use libraries like CountVectorizer and TfidfTransformer to enhance this process.

We then move to the modeling phase where we use machine learning models like Naïve Bayes and XG Boost on the data transformed by CountVectorizer and TfidfTransformer and compare the results of the models.

We then move on to the deep learning modeling phase where we use a LSTM model. In order to use the model, we first use a tokenizer that converts texts to sequences on the base training data as well as on the base testing data. We then run the model and then check the metrics ofthe model like accuracy score, confusion matrix etc.

On comparing the confusion matrices of the machine and deep learning models, we can come to know that the deep learning model performs well. So, we plan to save the model and use it for further testing purposes.

We then run the model with three separate test instances in order to classify whether they belong to the hate and offensive category ornot.

CODE SCREENSHOTS

```
[ ] # Importing Required Libraries
```

```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sns
    import re
    import nltk
    from nltk.corpus import stopwords
    import string
    from wordcloud import WordCloud, STOPWORDS
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.feature_extraction.text import TfidfTransformer
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.metrics import classification_report
    from sklearn.metrics import confusion_matrix
    import xgboost as xgb
    from keras.models import Model
    from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding,SpatialDropout1D
    from tensorflow.keras.optimizers import RMSprop
    from keras.preprocessing.text import Tokenizer
    from keras.preprocessing import sequence
    from tensorflow.keras.utils import to_categorical
    from keras.callbacks import EarlyStopping
    from keras.models import Sequential
    from keras.callbacks import EarlyStopping,ModelCheckpoint
    import keras
    import plotly.express as px
    import plotly.io as pio
    import pickle
```

```
[ ] # Reading the first dataset
```

```
[ ] df_twitter = pd.read_csv("/content/drive/MyDrive/NLP Project/train.csv")
```

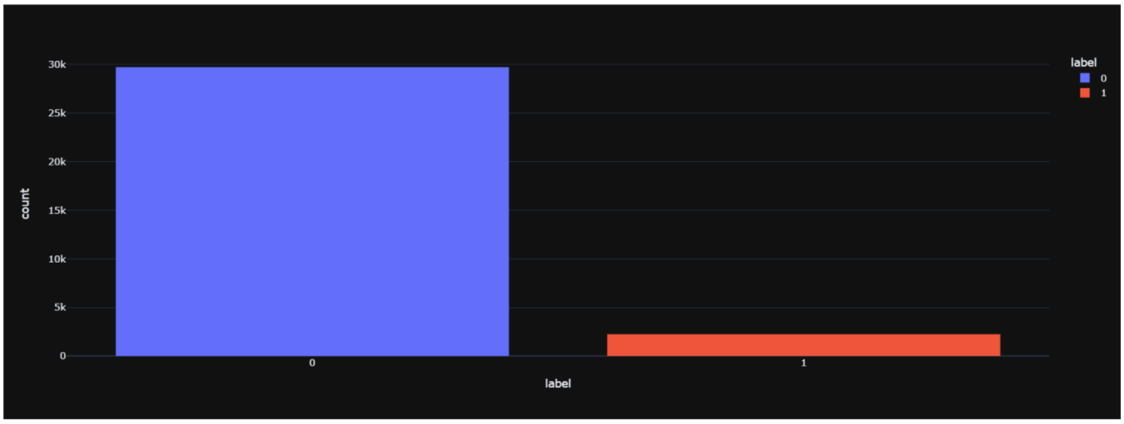
```
[ ] df_twitter.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

```
[ ] df_twitter1 = df_twitter
```

```
[ ] df_twitter1 = df_twitter1["label"].astype('str')
```

```
[ ] pio.templates.default = "plotly_dark"
    px.histogram(df_twitter1, x = "label", color = "label")
```




```
[ ] df_twitter.shape
```

```
(31962, 3)
```

```
[ ] df_twitter.isnull().sum()
```

```
id       0
label    0
tweet    0
dtype: int64
```

```
[ ] df_twitter.drop('id', axis = 1, inplace = True)
```

```
[ ] df_offensive = pd.read_csv("/content/drive/MyDrive/NLP Project/labeled_data.csv")
```

```
[ ] df_offensive.head()
```

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

```
[ ] df_offensive.shape
```

```
(24783, 7)
```

```
[ ] df_offensive.isnull().sum()
```

```
Unnamed: 0            0
count                 0
hate_speech           0
offensive_language    0
neither               0
class                 0
tweet                 0
dtype: int64
```

```
[ ] df_offensive.drop(['Unnamed: 0','count','hate_speech','offensive_language','neither'], axis = 1, inplace = True)
```

```
[ ] df_offensive.head(10)
```

	class	tweet
0	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
5	1	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...
6	1	!!!!!!"@__BrighterDays: I can not just sit up ...
7	1	!!!!“@selfiequeenbri: cause I'm tired of...
8	1	" & you might not get ya bitch back & ...
9	1	" @rhythmixx_ :hobbies include: fighting Maria...

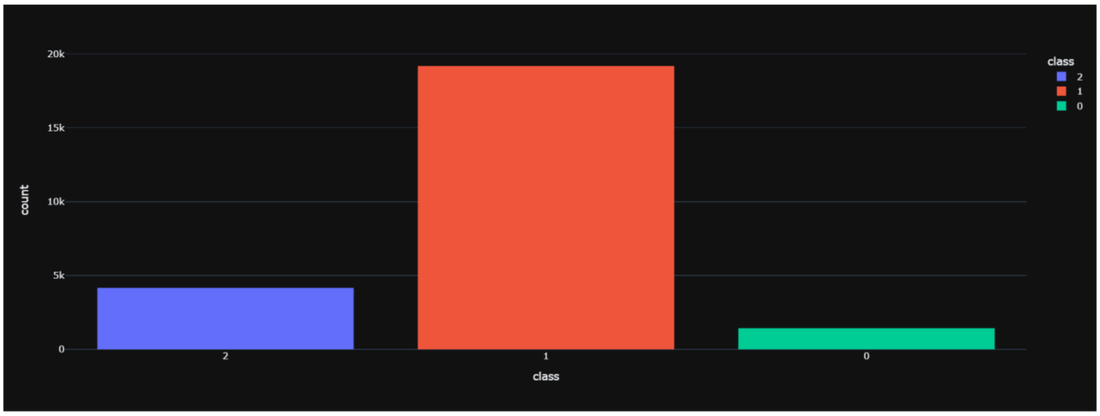
```
[ ] df_offensive['class'].unique()
```

```
array([2, 1, 0])
```

```
[ ] df_offensive1 = df_offensive
```

```
[ ] df_offensive1 = df_offensive1['class'].astype('str')
```

```
[ ] pio.templates.default = "plotly_dark"
    px.histogram(df_offensive1, x = "class", color = "class")
```



```
[ ] df_offensive[df_offensive['class'] == 0]
```

	class	tweet
85	0	"@Blackman38Tide: @WhaleLookyHere @HowdyDowdy1...
89	0	"@CB_Baby24: @white_thunduh alsarabsss" hes a ...
110	0	"@DevilGrimz: @VigxRArts you're fucking gay, b...
184	0	"@MarkRoundtreeJr: LMFAOOOO I HATE BLACK PEOPL...
202	0	"@NoChillPaz: "At least I'm not a nigger" http...
...	...	...
24576	0	this guy is the biggest faggot omfg
24685	0	which one of these names is more offensive kik...
24751	0	you a pussy ass nigga and I know it nigga.
24776	0	you're all niggers
24777	0	you're such a retard i hope you get type 2 dia...

1430 rows × 2 columns

```
[ ] df_offensive["class"].replace({0:1}, inplace = True)
```

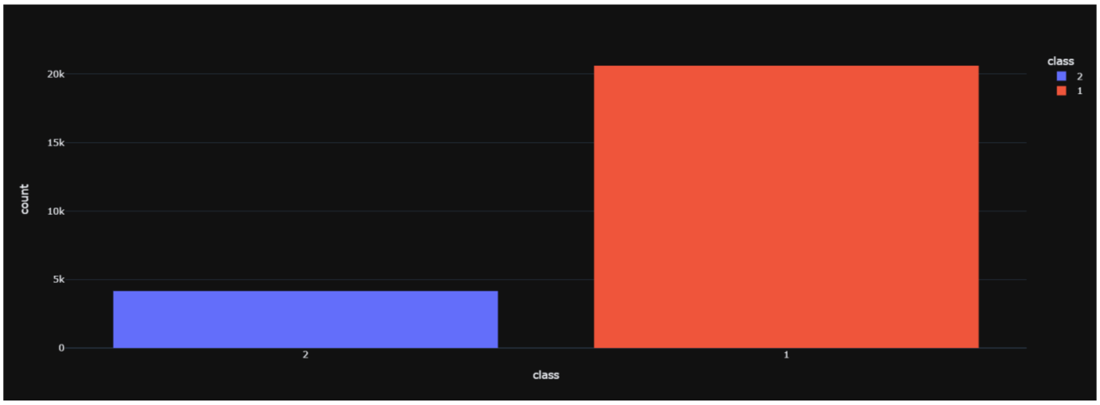
```
[ ] df_offensive['class'].unique()
```

```
array([2, 1])
```

```
[ ] df_offensive2 = df_offensive
```

```
[ ] df_offensive2 = df_offensive2['class'].astype('str')
```

```
[ ] pio.templates.default = "plotly_dark"
    px.histogram(df_offensive2, x = "class", color = "class")
```



```
[ ]  df_offensive[df_offensive['class'] == 0]
```

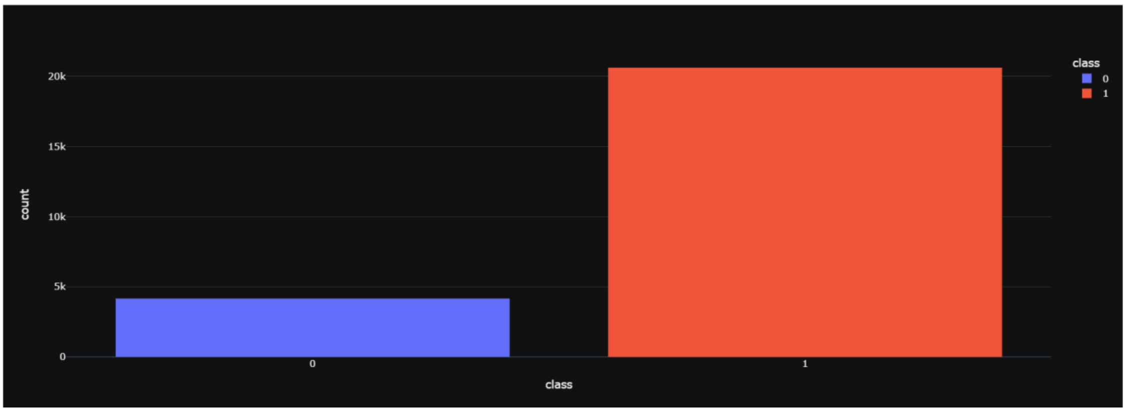
class	tweet

```
[ ]  df_offensive["class"].replace({2:0}, inplace = True)
```

```
[ ]  df_offensive3 = df_offensive
```

```
[ ]  df_offensive3 = df_offensive3['class'].astype('str')
```

```
[ ] pio.templates.default = "plotly_dark"
    px.histogram(df_offensive3, x = "class", color = "class")
```




```
[ ]  df_offensive.rename(columns = {'class':'label'}, inplace = True)
```

```
[ ]  df_offensive.head()
```

	label	tweet
0	0	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

```
[ ]  df_offensive.iloc[0]['tweet']

'!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. &amp; as a man you should always take the trash out...'
```

```
[ ]  df_offensive.iloc[5]['tweet']

'!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with hoes! &#128514;&#128514;&#128514;"'
```

```
[ ]  frame = [df_twitter,df_offensive]
     df = pd.concat(frame)
```

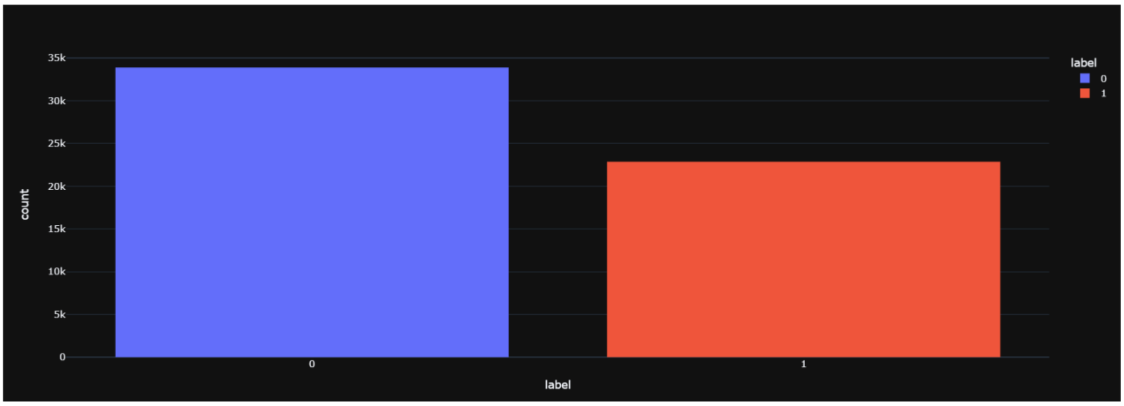
```
[ ] df.head()
```

	label	tweet
0	0	@user when a father is dysfunctional and is s...
1	0	@user @user thanks for #lyft credit i can't us...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...
4	0	factsguide: society now #motivation

```
[ ] df1 = df
```

```
[ ] df1 = df1['label'].astype('str')
```

```
[ ] pio.templates.default = "plotly_dark"
    px.histogram(df1, x = "label", color = "label")
```




```
[ ] df.shape
```

```
(56745, 2)
```

```
[ ] nltk.download('stopwords')
    stemmer = nltk.SnowballStemmer("english")
    stopword = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[ ] def make_wordcloud(df):
        comment_words=""
        for val in df.tweet:
            val = str(val).lower()
            comment_words += " ".join(val)+" "
        print(comment_words[0:100])
        wordcloud = WordCloud(width = 800, height = 800,
                    background_color ='white',
                    stopwords = stopwords,min_font_size = 10).generate(comment_words)

        plt.figure(figsize = (8, 8), facecolor = None)
        plt.imshow(wordcloud)
        plt.axis("off")
        plt.tight_layout(pad = 0)
        plt.show()
```

```
def clean_text(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text = " ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text = " ".join(text)
    return text
```

```
df['tweet'] = df['tweet'].apply(clean_text)
```

```
df.head()
```

	label	tweet
0	0	user father dysfunct selfish drag kid dysfunc...
1	0	user user thank lyft credit cant use caus dont...
2	0	bihday majesti
3	0	model love u take u time urð□□± ð□□□ð□□□ð□□□...
4	0	factsguid societi motiv

```
x = df['tweet']
y = df['label']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 42)
print(len(x_train), len(y_train))
print(len(x_test), len(y_test))
```

```
42558 42558
14187 14187
```

```
count = CountVectorizer(stop_words = 'english', ngram_range = (1,5))
x_train_vectorizer = count.fit_transform(x_train)
```

```
x_test_vectorizer = count.transform(x_test)
```

```
[ ] count.vocabulary_
```

```
 'got hoe': 264878,
 'boy ryanbowersob got': 87737,
 'ryanbowersob got hoe': 573407,
 'boy ryanbowersob got hoe': 87738,
 'hate': 286834,
 'aliv': 11793,
 'hater': 288324,
 'orlandounit': 480071,
 'hate aliv': 286847,
 'aliv hater': 11819,
 'hater orlandounit': 288353,
 'hate aliv hater': 286848,
 'aliv hater orlandounit': 11820,
 'hate aliv hater orlandounit': 286849,
 'book': 83638,
 'endur': 192647,
 'convict': 131799,
 'fred': 231715,
 'korematsu': 359709,
 'amp': 17807,
 'quest': 523179,
 'justic': 348074,
 'humanright': 313763,
 'htâ': 312750,
 'book endur': 83763,
 'endur convict': 192648,
 'convict fred': 131800,
 'fred korematsu': 231716,
 'korematsu amp': 359710,
```

```
[ ] tfidf = TfidfTransformer()
    x_train_tfidf = tfidf.fit_transform(x_train_vectorizer)
    x_test_tfidf = tfidf.transform(x_test_vectorizer)
```

```
[ ] model_vectorizer = MultinomialNB().fit(x_train_vectorizer, y_train)
    prediction_vectorizer = model_vectorizer.predict(x_test_vectorizer)
    print(confusion_matrix(y_test,prediction_vectorizer))
    print(classification_report(y_test, prediction_vectorizer))
```

```
[[7878  575]
 [ 458 5276]]
              precision    recall  f1-score   support

           0       0.95      0.93      0.94      8453
           1       0.90      0.92      0.91      5734

    accuracy                           0.93     14187
   macro avg       0.92      0.93      0.92     14187
weighted avg       0.93      0.93      0.93     14187
```

```
model_tfidf = MultinomialNB().fit(x_train_tfidf, y_train)
prediction_tfidf = model_tfidf.predict(x_test_tfidf)
print(confusion_matrix(y_test,prediction_tfidf))
print(classification_report(y_test, prediction_tfidf))
```

```
[[8213  240]
 [ 860 4874]]
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      8453
           1       0.95      0.85      0.90      5734

    accuracy                           0.92     14187
   macro avg       0.93      0.91      0.92     14187
weighted avg       0.92      0.92      0.92     14187
```

```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(
        learning_rate = 0.1,
        max_depth = 7,
        n_estimators = 80,
        use_label_encoder = False,
        eval_metric ='auc' )
```

```
xgb_model_vectorizer = xgb_model.fit(x_train_vectorizer, y_train)
xgb_predictions_vectorizer = xgb_model_vectorizer.predict(x_test_vectorizer)
print(confusion_matrix(y_test,xgb_predictions_vectorizer))
print(classification_report(y_test, xgb_predictions_vectorizer))
```

```
[[8367   86]
 [ 923 4811]]
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      8453
           1       0.98      0.84      0.91      5734

    accuracy                           0.93     14187
   macro avg       0.94      0.91      0.92     14187
weighted avg       0.93      0.93      0.93     14187
```

```
xgb_model = xgb_model.fit(x_train_tfidf, y_train)
xgb_predictions = xgb_model.predict(x_test_tfidf)
print(confusion_matrix(y_test,xgb_predictions))
print(classification_report(y_test, xgb_predictions))
```

```
[[8358   95]
 [ 939 4795]]
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      8453
           1       0.98      0.84      0.90      5734

    accuracy                           0.93     14187
   macro avg       0.94      0.91      0.92     14187
weighted avg       0.93      0.93      0.93     14187
```

```
[ ] max_words = 50000
    max_len = 300
    tokenizer = Tokenizer(num_words=max_words)
    tokenizer.fit_on_texts(x_train)
    sequences = tokenizer.texts_to_sequences(x_train)
    sequences_matrix = sequence.pad_sequences(sequences,maxlen = max_len)
```

```
[ ] model = Sequential()
    model.add(Embedding(max_words, 100, input_length = max_len))
    model.add(SpatialDropout1D(0.2))
    model.add(LSTM(100, dropout = 0.2, recurrent_dropout = 0.2))
    model.add(Dense(1, activation = 'sigmoid'))
    model.summary()
    model.compile(loss = 'binary_crossentropy', optimizer = RMSprop(), metrics = ['accuracy'])
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 100)          5000000

 spatial_dropout1d (SpatialD  (None, 300, 100)         0
 ropout1D)

 lstm (LSTM)                 (None, 100)               80400

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 5,080,501
Trainable params: 5,080,501
Non-trainable params: 0
```

```
[ ] stop = EarlyStopping(
        monitor = 'val_accuracy',
        mode = 'max',
        patience = 5
    )

    checkpoint = ModelCheckpoint(
        filepath ='./',
        save_weights_only = True,
        monitor ='val_accuracy',
        mode ='max',
        save_best_only = True)
```

```
[ ] history = model.fit(sequences_matrix, y_train, batch_size = 1024, epochs = 15,
              validation_split = 0.2, callbacks=[stop,checkpoint])
```

```
Epoch 1/15
34/34 [==============================] - 48s 1s/step - loss: 0.4828 - accuracy: 0.7921 - val_loss: 0.2979 - val_accuracy: 0.8977
Epoch 2/15
34/34 [==============================] - 44s 1s/step - loss: 0.2348 - accuracy: 0.9218 - val_loss: 0.2029 - val_accuracy: 0.9286
Epoch 3/15
34/34 [==============================] - 45s 1s/step - loss: 0.1471 - accuracy: 0.9488 - val_loss: 0.1643 - val_accuracy: 0.9409
Epoch 4/15
34/34 [==============================] - 44s 1s/step - loss: 0.1110 - accuracy: 0.9620 - val_loss: 0.1657 - val_accuracy: 0.9394
Epoch 5/15
34/34 [==============================] - 45s 1s/step - loss: 0.0906 - accuracy: 0.9687 - val_loss: 0.1633 - val_accuracy: 0.9456
Epoch 6/15
34/34 [==============================] - 45s 1s/step - loss: 0.0770 - accuracy: 0.9744 - val_loss: 0.1614 - val_accuracy: 0.9418
Epoch 7/15
34/34 [==============================] - 44s 1s/step - loss: 0.0675 - accuracy: 0.9774 - val_loss: 0.1746 - val_accuracy: 0.9429
Epoch 8/15
34/34 [==============================] - 45s 1s/step - loss: 0.0584 - accuracy: 0.9811 - val_loss: 0.1890 - val_accuracy: 0.9420
Epoch 9/15
34/34 [==============================] - 44s 1s/step - loss: 0.0516 - accuracy: 0.9834 - val_loss: 0.1856 - val_accuracy: 0.9416
Epoch 10/15
34/34 [==============================] - 45s 1s/step - loss: 0.0457 - accuracy: 0.9850 - val_loss: 0.1996 - val_accuracy: 0.9326
```

```
[ ] test_sequences = tokenizer.texts_to_sequences(x_test)
    test_sequences_matrix = sequence.pad_sequences(test_sequences, maxlen = max_len)
```

```
[ ] accr = model.evaluate(test_sequences_matrix, y_test)
```

```
444/444 [==============================] - 37s 82ms/step - loss: 0.2226 - accuracy: 0.9281
```

```
[ ] lstm_prediction = model.predict(test_sequences_matrix)
```

```
[ ] res = []
    for prediction in lstm_prediction:
        if prediction[0] < 0.5:
            res.append(0)
        else:
            res.append(1)
```

```
[ ] print(confusion_matrix(y_test, res))
```

```
[[7871  582]
 [ 438 5296]]
```

```
[ ] with open('tokenizer.pickle', 'wb') as handle:
        pickle.dump(tokenizer, handle, protocol = pickle.HIGHEST_PROTOCOL)
```

```
[ ] model.save("hate&abusive_model.h5")
```

```
[ ] load_model = keras.models.load_model("./hate&abusive_model.h5")
    with open('tokenizer.pickle', 'rb') as handle:
        load_tokenizer = pickle.load(handle)
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

```
[ ] test = 'I love NLP!!!'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test = [clean_text(test)]
    seq = load_tokenizer.texts_to_sequences(test)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and offensive")
```

```
I love NLP!!!
i love nlp
pred [[0.19120216]]
no hate
```

```
[ ] test1 = 'I hate you'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test1 = [clean_text(test1)]
    seq = load_tokenizer.texts_to_sequences(test1)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and offensive")
```

```
I hate you
i hate you
pred [[0.6952494]]
hate and offensive
```

```
[ ] test2 = 'You are a bloody bitch!!!'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test2 = [clean_text(test2)]
    seq = load_tokenizer.texts_to_sequences(test2)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and abusive")
```

```
You are a bloody bitch!!!
you are a bloody bitch
pred [[0.99358785]]
hate and abusive
```

```
[ ] test3 = 'I hate people who are dumb'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test3 = [clean_text(test3)]
    seq = load_tokenizer.texts_to_sequences(test3)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and abusive")
```

```
I hate people who are dumb
i hate people who are dumb
pred [[0.9158933]]
hate and abusive
```

RESULTS AND CONCLUSION

After the modeling phase in our project, we implemented four test cases where the selected best model evaluated every case accurately. So,the model can be associated even with the real time use cases.

Here are the test cases and the predictions:

```
[ ] test = 'I love NLP!!!'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test = [clean_text(test)]
    seq = load_tokenizer.texts_to_sequences(test)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and offensive")
```

```
I love NLP!!!
i love nlp
pred [[0.19120216]]
no hate
```

```
[ ] test1 = 'I hate you'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test1 = [clean_text(test1)]
    seq = load_tokenizer.texts_to_sequences(test1)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and offensive")
```

```
I hate you
i hate you
pred [[0.6952494]]
hate and offensive
```

```
[ ] test2 = 'You are a bloody bitch!!!'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test2 = [clean_text(test2)]
    seq = load_tokenizer.texts_to_sequences(test2)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and abusive")
```

```
You are a bloody bitch!!!
you are a bloody bitch
pred [[0.99358785]]
hate and abusive
```

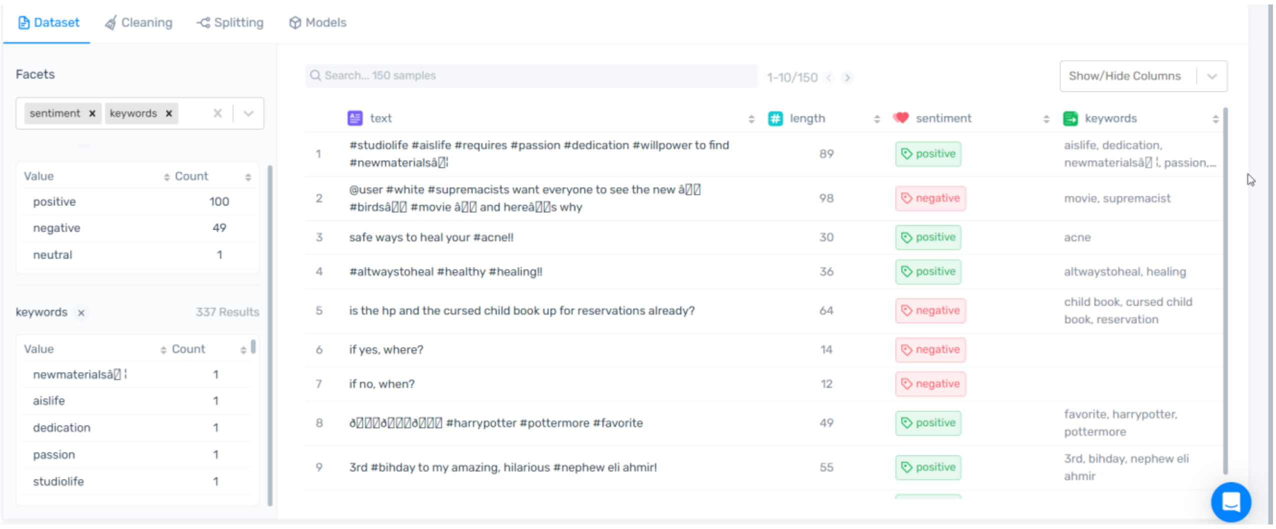
```
[ ] test3 = 'I hate people who are dumb'
    def clean_text(text):
        print(text)
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        print(text)
        text = [word for word in text.split(' ') if word not in stopword]
        text=" ".join(text)
        text = [stemmer.stem(word) for word in text.split(' ')]
        text=" ".join(text)
        return text
    test3 = [clean_text(test3)]
    seq = load_tokenizer.texts_to_sequences(test3)
    padded = sequence.pad_sequences(seq, maxlen=300)
    pred = load_model.predict(padded)
    print("pred", pred)
    if pred < 0.5:
        print("no hate")
    else:
        print("hate and abusive")

    I hate people who are dumb
    i hate people who are dumb
    pred [[0.9158933]]
    hate and abusive
```

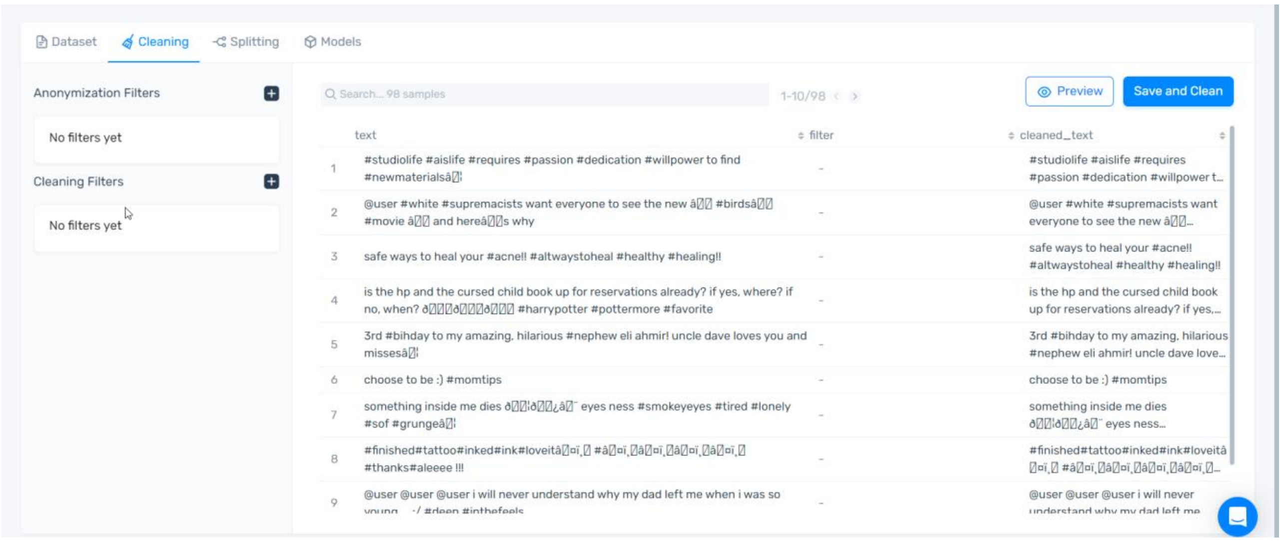
So, we conclude our project by creating a **Hate and Offensive Tweets Classifier** which can be relied on for classifying tweets in real time.

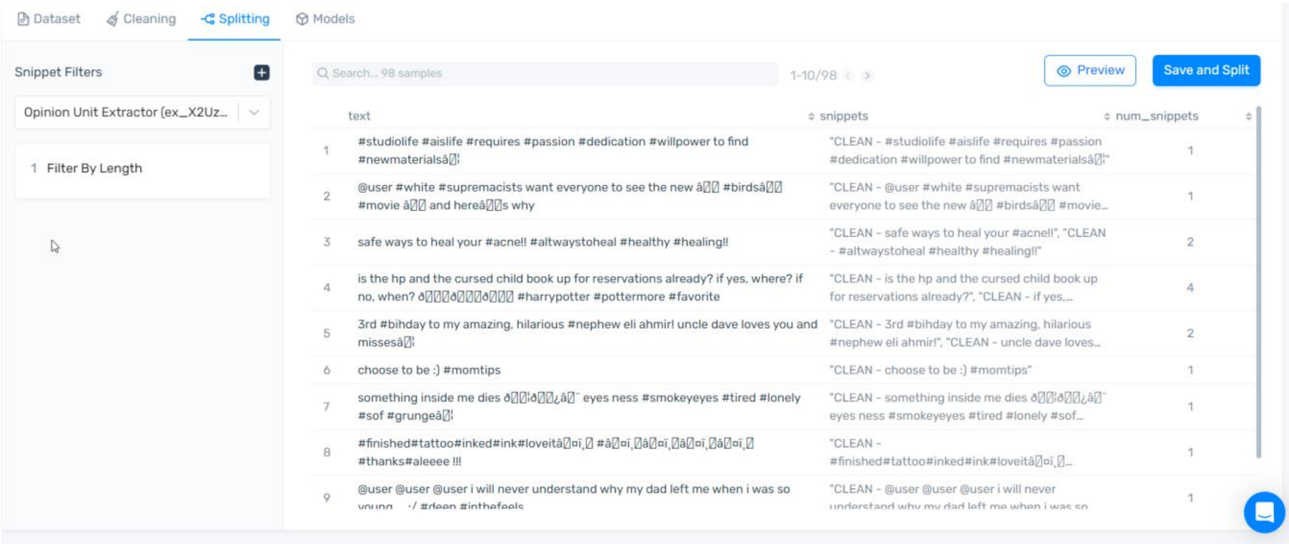
<u>Analytics using Social media tool</u>

Tool used: Monkey Learn

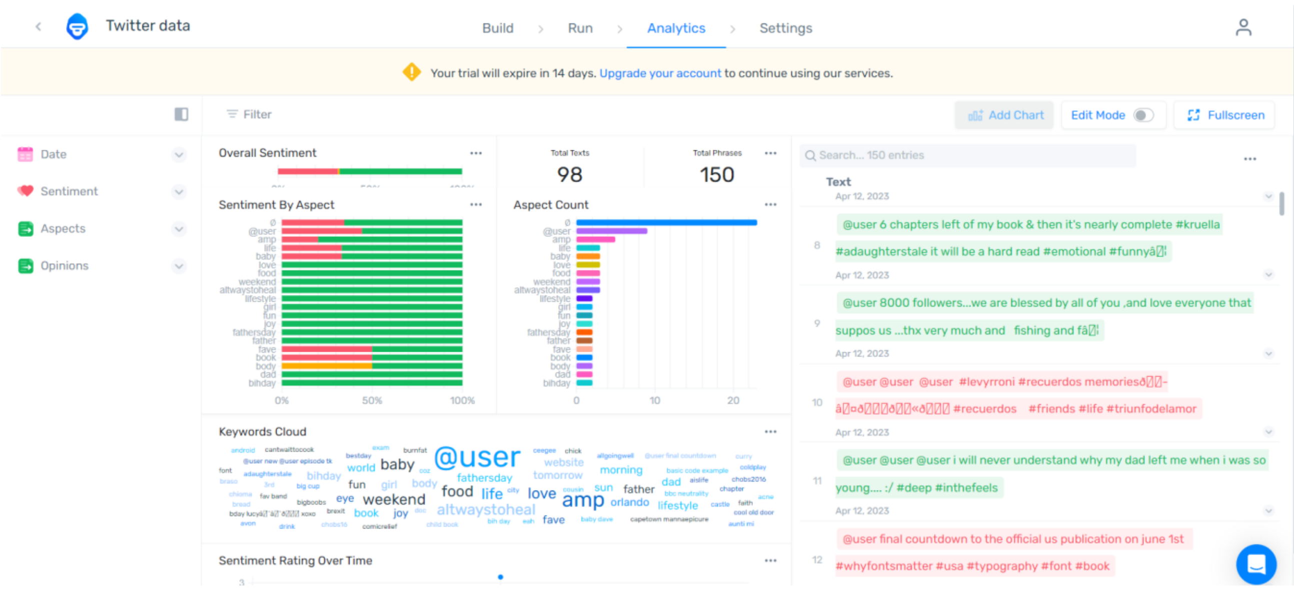
[Dataset, Cleaning, Splitting]

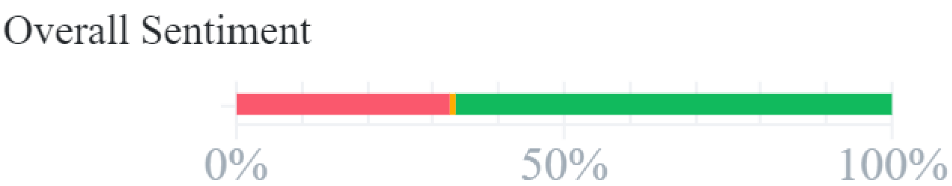




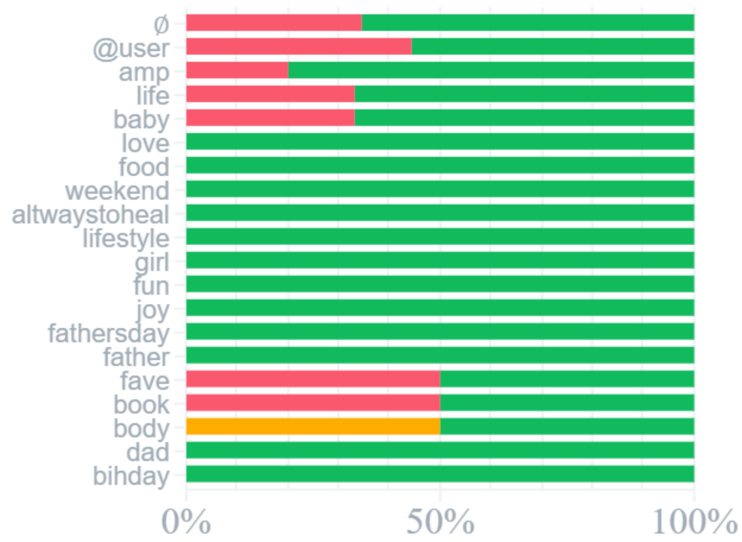

Dataset
Cleaning
Splitting
Models
Snippet Filters
Opinion Unit Extractor (ex_X2Uz...
1 Filter By Length
Search... 98 samples
1-10/98
Preview
Save and Split
text
snippets
num_snippets
#studiolife #aislife #requires #passion #dedication #willpower to find #newmaterialsâ
"CLEAN - #studiolife #aislife #requires #passion #dedication #willpower to find #newmaterialsâ"
safe ways to heal your #acne!! #altwaystoheal #healthy #healing!!
"CLEAN - safe ways to heal your #acne!!", "CLEAN - #altwaystoheal #healthy #healing!!"
is the hp and the cursed child book up for reservations already? if yes, where? if no, when? #harrypotter #pottermore #favorite
"CLEAN - is the hp and the cursed child book up for reservations already?", "CLEAN - if yes,...
3rd #bihday to my amazing, hilarious #nephew eli ahmir! uncle dave loves you and missesâ
"CLEAN - 3rd #bihday to my amazing, hilarious #nephew eli ahmir!", "CLEAN - uncle dave loves...
choose to be :) #momtips
"CLEAN - choose to be :) #momtips"
@user @user @user i will never understand why my dad left me when i was so
"CLEAN - @user @user @user i will never

Analytics

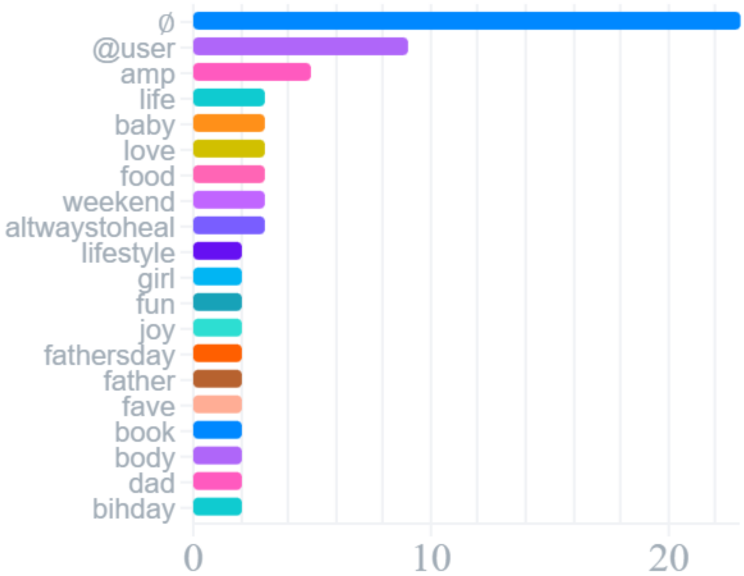

Twitter data
Build
Run
Analytics
Settings
Your trial will expire in 14 days. Upgrade your account to continue using our services.
Filter
Add Chart
Edit Mode
Fullscreen
Date
Sentiment
Aspects
Opinions
Overall Sentiment
Total Texts
98
Total Phrases
150
Sentiment By Aspect
Aspect Count
Keywords Cloud
Sentiment Rating Over Time
Search... 150 entries
Text
@user 6 chapters left of my book & then it's nearly complete #kruella
#adaughterstale it will be a hard read #emotional #funny
@user 8000 followers...we are blessed by all of you ,and love everyone that
suppos us ...thx very much and fishing and f
@user @user @user #levyrroni #recuerdos memories
#recuerdos #friends #life #triunfodelamor
@user @user @user i will never understand why my dad left me when i was so
young.... :/ #deep #inthefeels
@user final countdown to the official us publication on june 1st
#whyfontsmatter #usa #typography #font #book


Overall Sentiment
0%
50%
100%

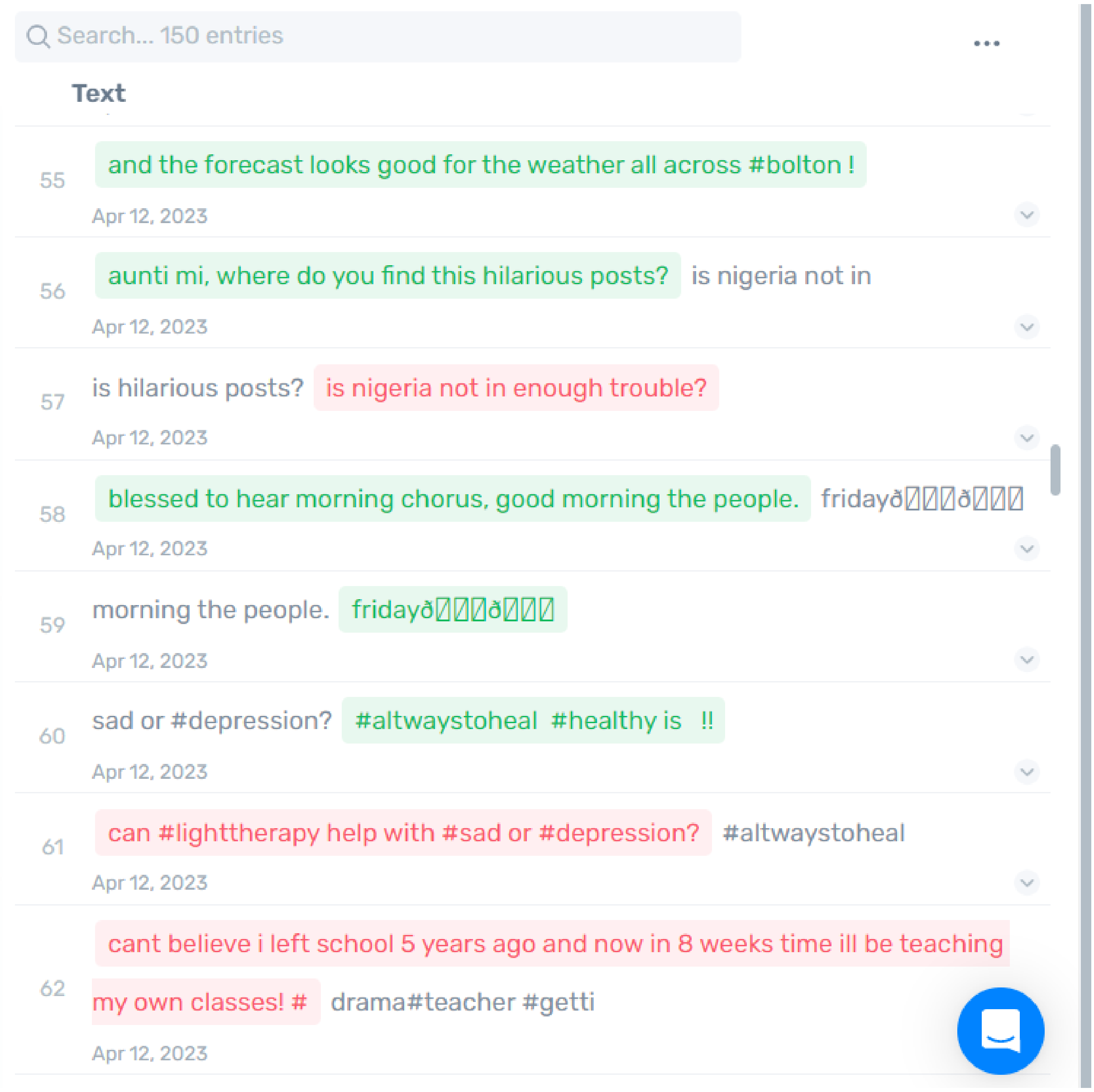
Sentiment by Aspect



Aspect Count

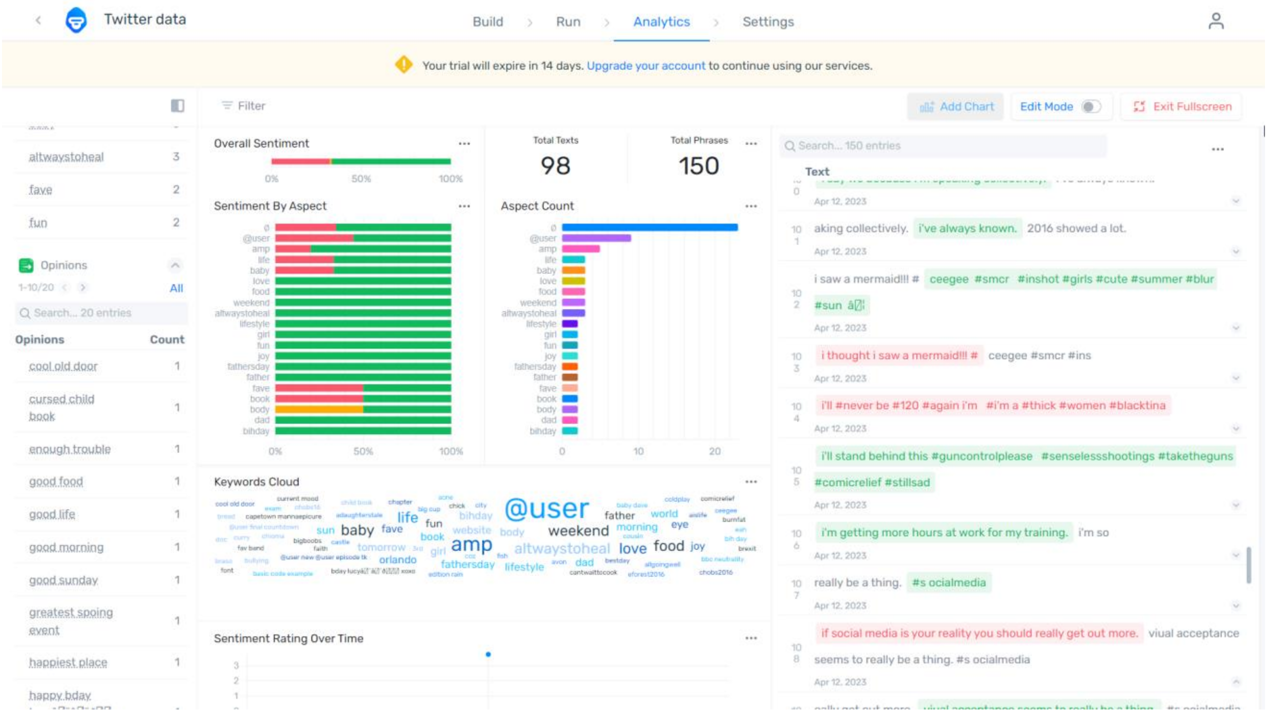


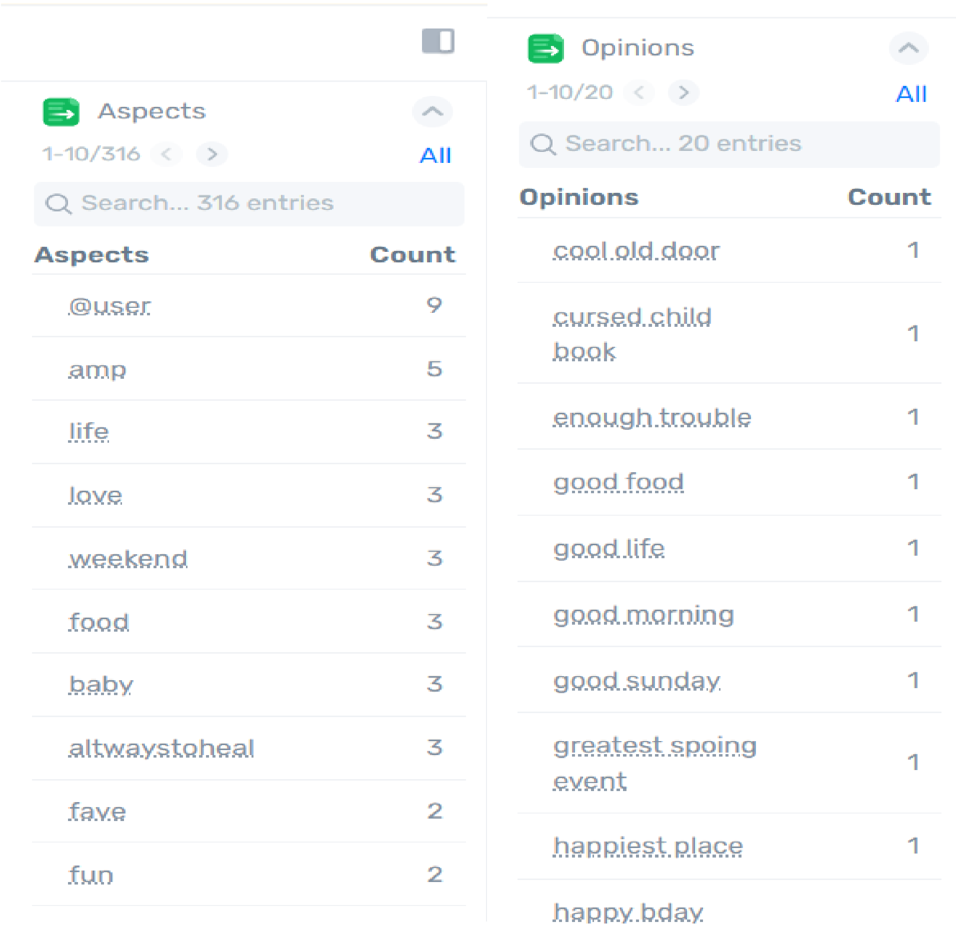
Keywords Cloud

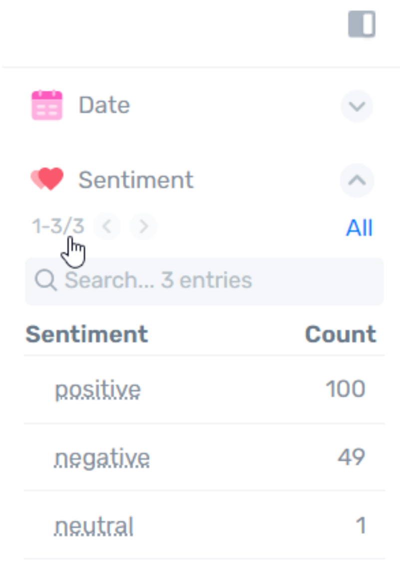


Search... 150 entries

	Text
55	and the forecast looks good for the weather all across #bolton ! Apr 12, 2023
56	aunti mi, where do you find this hilarious posts? is nigeria not in Apr 12, 2023
57	is hilarious posts? is nigeria not in enough trouble? Apr 12, 2023
58	blessed to hear morning chorus, good morning the people. fridayð□□□ð□□□ Apr 12, 2023
59	morning the people. fridayð□□□ð□□□ Apr 12, 2023
60	sad or #depression? #altwaystoheal #healthy is !! Apr 12, 2023
61	can #lighttherapy help with #sad or #depression? #altwaystoheal Apr 12, 2023
62	cant believe i left school 5 years ago and now in 8 weeks time ill be teaching my own classes! # drama#teacher #getti Apr 12, 2023


Twitter data
Build
Run
Analytics
Settings
Your trial will expire in 14 days. Upgrade your account to continue using our services.
Filter
Add Chart
Edit Mode
Exit Fullscreen
Overall Sentiment
Total Texts
98
Total Phrases
150
Sentiment By Aspect
Aspect Count
Keywords Cloud
Sentiment Rating Over Time
Opinions
Count
Text


Aspects
1-10/316
All
Search... 316 entries
Aspects
Count
@user 9
amp 5
life 3
love 3
weekend 3
food 3
baby 3
altwaystoheal 3
fave 2
fun 2
Opinions
1-10/20
All
Search... 20 entries
Opinions
Count
cool old door 1
cursed child book 1
enough trouble 1
good food 1
good life 1
good morning 1
good sunday 1
greatest spoing event 1
happiest place 1
happy bday



Sentiment	Count
positive	100
negative	49
neutral	1

REFERENCES

Datasets –

https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset

https://www.kaggle.com/datasets/arkhoshghalb/twitter-sentiment-analysis-hatred-speech

CountVectorizer and TfidfTransformer –

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

Stopwords –

https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

Models –

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

https://xgboost.readthedocs.io/en/stable/python/python_api.html

https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM