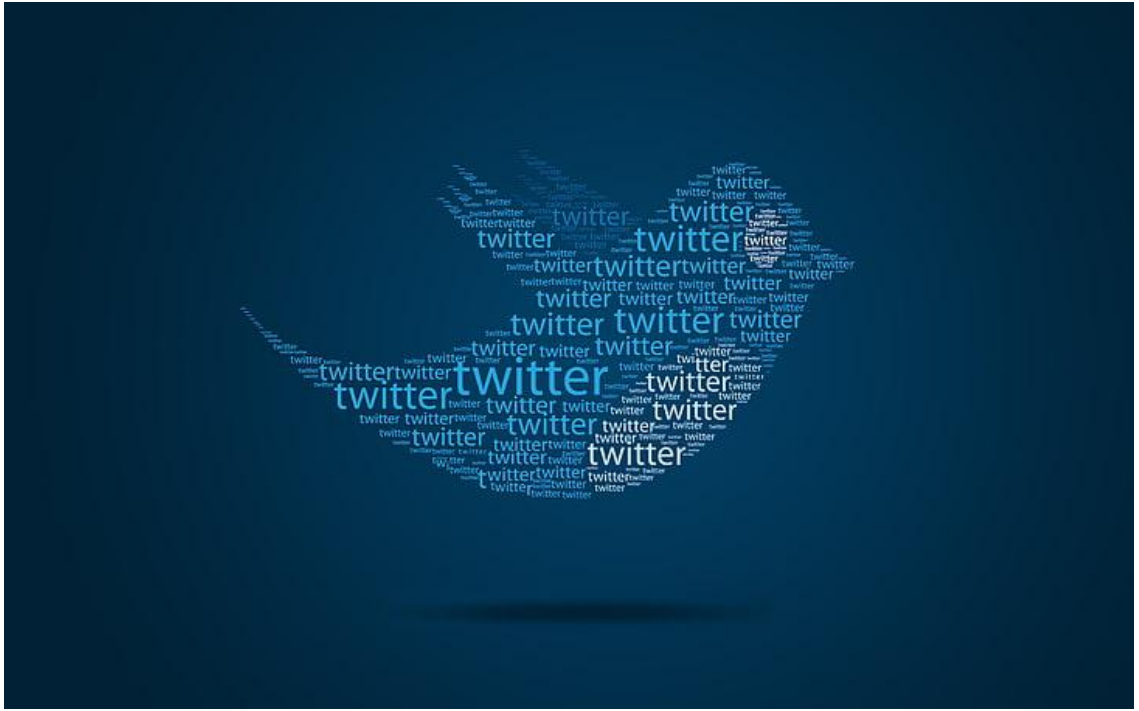
Tweet Sentiment Predictor

Andrew Y

Problem and Data Description

Problem

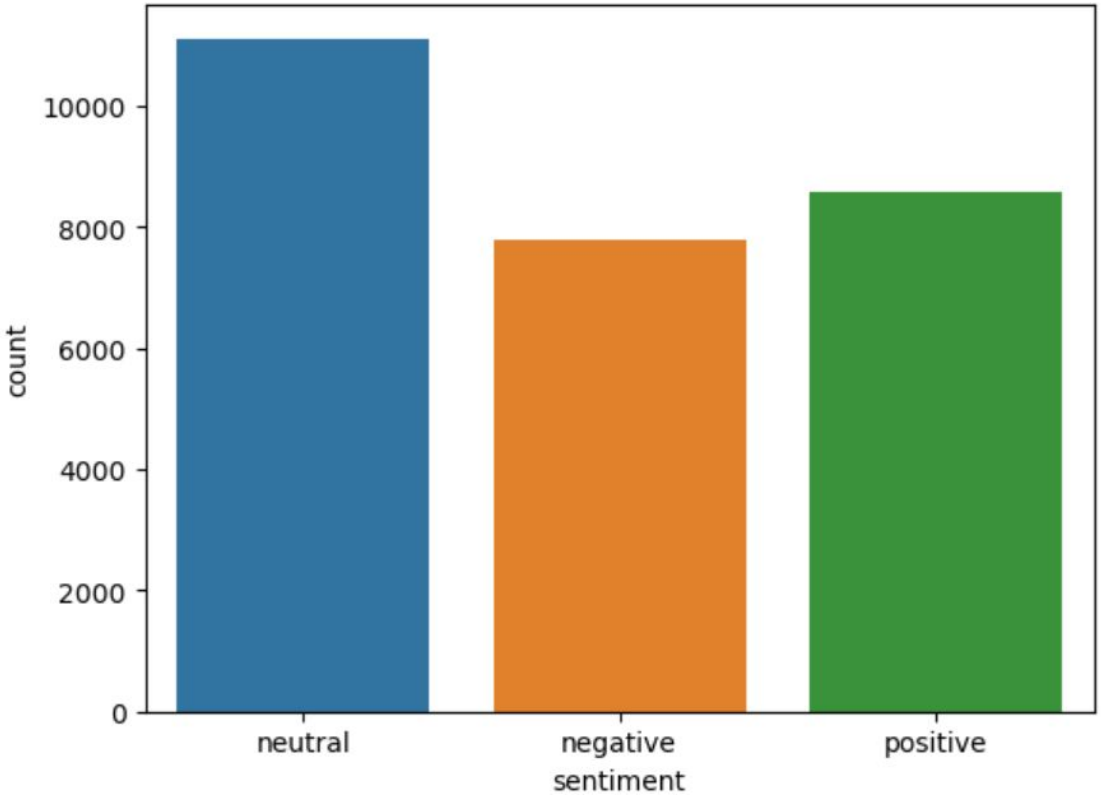


Data Description

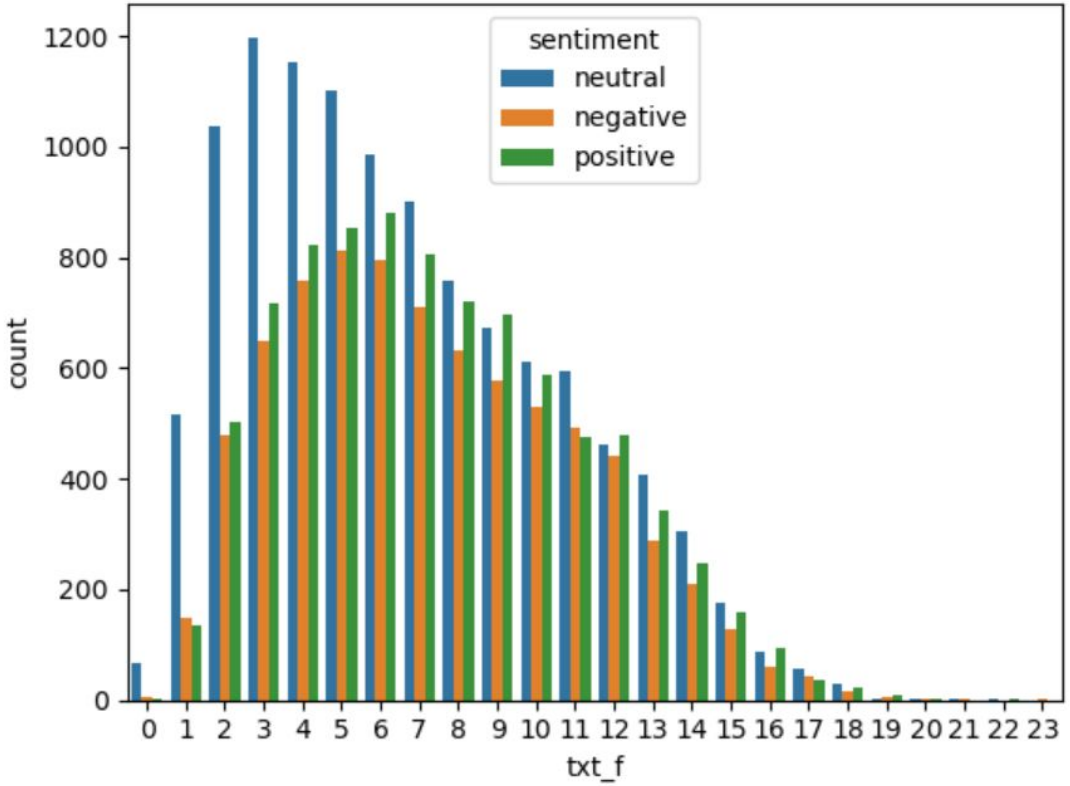
	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative

EDA and Data Preprocessing

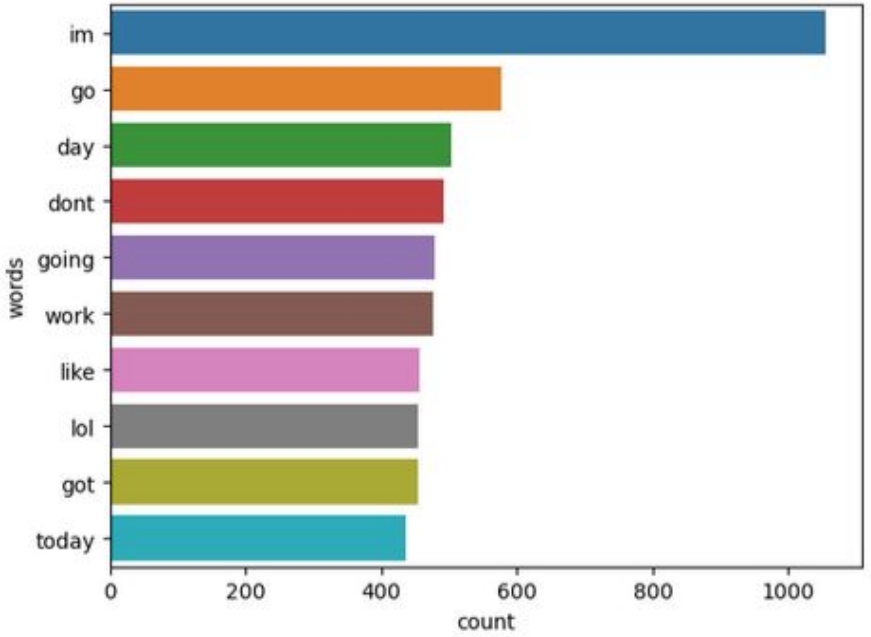
EDA - Label Distribution



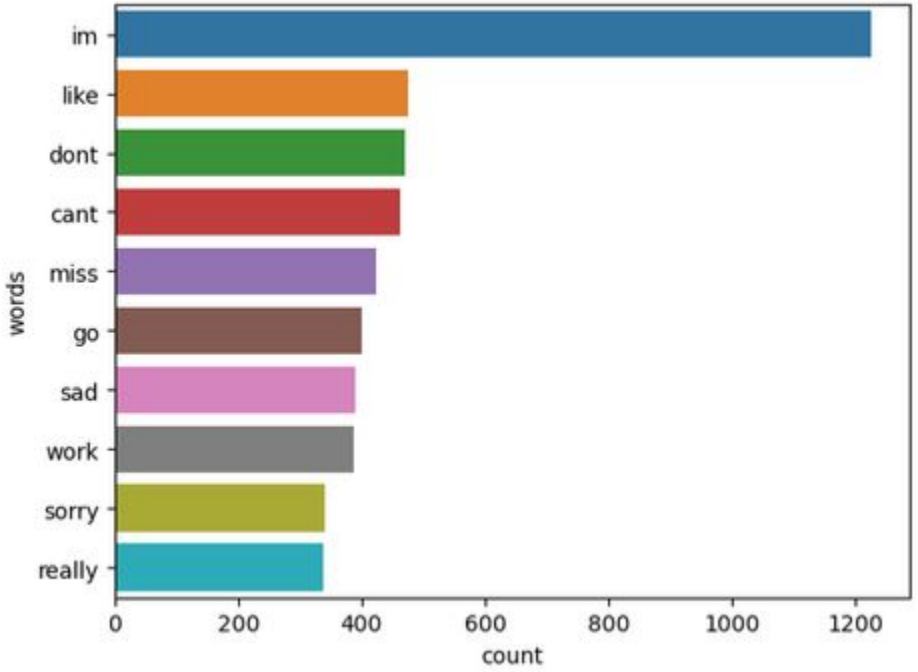
EDA - Word Count per Label



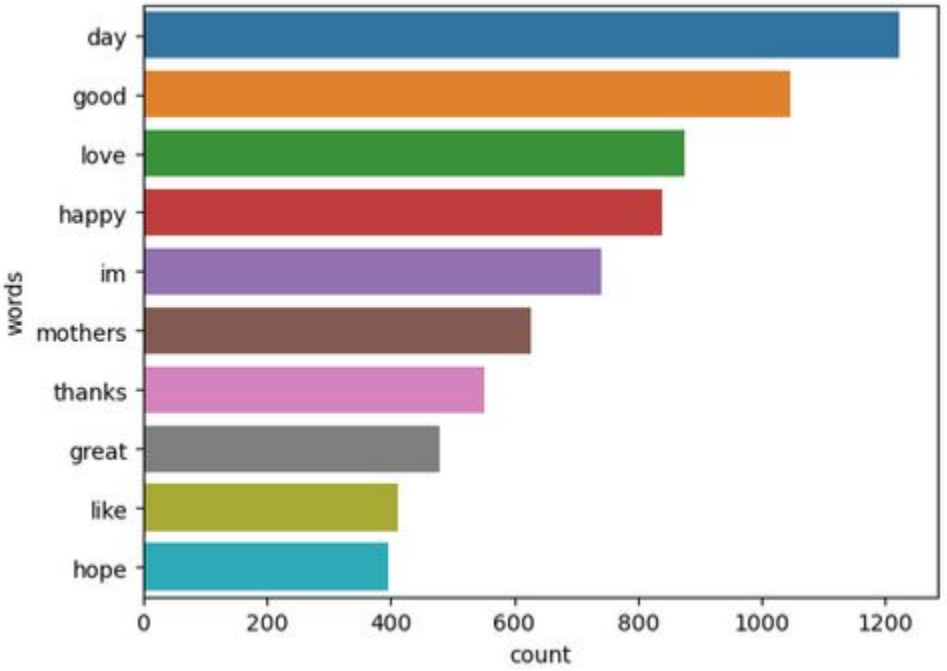
EDA - Neutral Words



EDA - Negative Words



EDA - Positive Words



Data Preprocessing

	text	txt	sentiment	sentiment
0	I`d have responded, if I were going	id responded going	neutral	0
1	Sooo SAD I will miss you here in San Diego!!!	sooo sad miss san diego	negative	-1
2	my boss is bullying me...	boss bullying	negative	-1
3	what interview! leave me alone	interview leave alone	negative	-1
4	Sons of ****, why couldn`t they put them on t...	sons couldnt put releases already bought	negative	-1

Model Architecture

Model

- SimpleRNN
- LSTM
- Bidirectional LSTM
- GRU

1. Model - SimpleRNN

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(SimpleRNN(50))
model.add(Dense(3, activation = 'sigmoid'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_1 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

2. Model - LSTM

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(LSTM(50))
model.add(Dense(3, activation = 'sigmoid'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_2 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

3. Model - Bidirectional LSTM

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(Bidirectional(LSTM(50)))
model.add(Dense(3, activation = 'sigmoid'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_3 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

4. Model - GRU

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(GRU(50))
model.add(Dense(3, activation = 'sigmoid'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_4 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

5. HPT - Multiple LSTM

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(LSTM(50, return_sequences = True))
model.add(LSTM(25))

model.add(Dense(3, activation = 'sigmoid'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_5 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

6. HPT - Multiple Dense Layers

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(LSTM(50, return_sequences = True))
model.add(LSTM(25))
model.add(Dense(50, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_6 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

7. HPT - Spatial Dropout

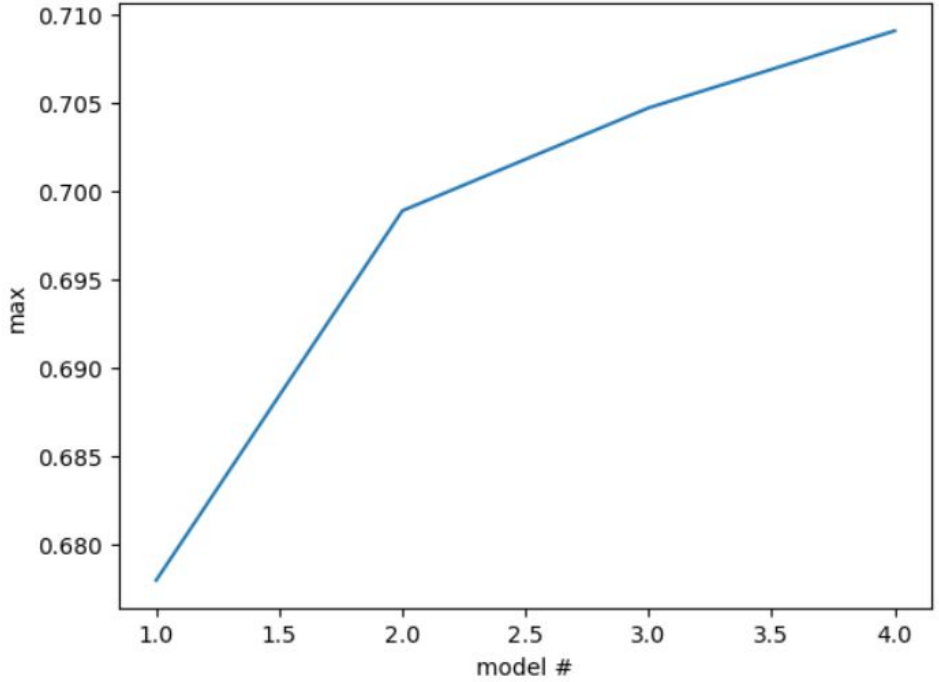
```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(50, activation = 'relu'))
model.add(Dense(3, activation='softmax'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_7 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

8. HPT - Multiple Dropout Layers

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64
dropout = 0.2

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(Dropout(dropout))
model.add(LSTM(50))
model.add(Dropout(dropout))
model.add(Dense(50, activation = 'relu'))
model.add(Dropout(dropout))
model.add(Dense(3, activation = 'softmax'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_8 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

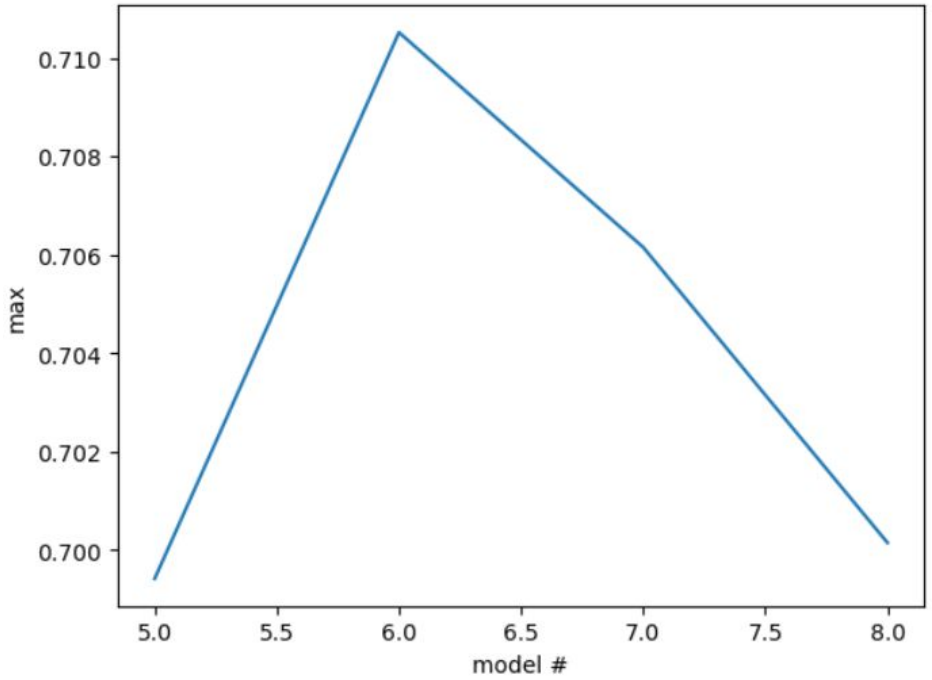
Results and Analysis

Model - Results



	model	val_accuracy
3	4	0.709061
2	3	0.704694
1	2	0.698872
0	1	0.677948

HPT - Results



	model	val_accuracy
1	6	0.710517
2	7	0.706150
3	8	0.700146
0	5	0.699418

Conclusion

Best Model

```
dictionary_length = len(tokenizer.word_index.keys()) + 1
sequence_len = 64

model = Sequential()

model.add(Embedding(dictionary_length, sequence_len))
model.add(LSTM(50, return_sequences = True))
model.add(LSTM(25))
model.add(Dense(50, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

model.compile(Adam(learning_rate = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_6 = model.fit(X_train, y_train,
                      epochs = 5,
                      batch_size = 32,
                      validation_data = (X_test, y_test))
```

- Stacking LSTM layers
- Stacking Dense layers
- Softmax activation function on output

Best Model

```
1 model_f.predict(X_test)
```

```
172/172 [==============================] - 2s 6ms/step

array([[1.1596869e-03, 9.1401963e-03, 9.8970008e-01],
       [9.9487358e-01, 4.1722842e-03, 9.5408713e-04],
       [2.4897162e-02, 9.4309795e-01, 3.2004874e-02],
       ...,
       [4.2054015e-03, 9.8808903e-01, 7.7055553e-03],
       [1.6851228e-03, 1.0124150e-02, 9.8819077e-01],
       [9.4549119e-01, 5.1044062e-02, 3.4646972e-03]], dtype=float32)
```

```
1 y_test
```

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

Credits

Github

- https://github.com/awyeh64/Sentiment-Predictor

Data

- https://www.kaggle.com/datasets/yasserh/twitter-tweets-sentiment-dataset/data