```
import pandas as pd
import numpy as np

df= pd.read_csv('labeled_data.csv')
df.head()
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class
\
0           0      3            0                   0        3      2

1           1      3            0                   3        0      1

2           2      3            0                   3        0      1

3           3      3            0                   2        1      1

4           4      6            0                   6        0      1


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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24783 entries, 0 to 24782
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          24783 non-null  int64
 1   count               24783 non-null  int64
 2   hate_speech         24783 non-null  int64
 3   offensive_language  24783 non-null  int64
 4   neither             24783 non-null  int64
 5   class               24783 non-null  int64
 6   tweet               24783 non-null  object
dtypes: int64(6), object(1)
memory usage: 1.3+ MB
```

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
import string
```

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
True
```

```
def concat_text(tokens):
    return " ".join([token for token in tokens])

df['tweet'] = df['tweet'].apply(word_tokenize)
df['tweet'] = df['tweet'].apply(concat_text)
df.head()
```

	Unnamed: 0	count	hate_speech	offensive_language	neither	class
0	0	3	0	0	3	2
1	1	3	0	3	0	1
2	2	3	0	3	0	1
3	3	3	0	2	1	1
4	4	6	0	6	0	1

	tweet
0	! ! ! RT @ mayasolovely : As a woman you shoul...
1	! ! ! ! ! RT @ mleew17 : boy dats cold ... tyg...
2	! ! ! ! ! ! ! RT @ UrKindOfBrand Dawg ! ! ! ! ...
3	! ! ! ! ! ! ! ! ! RT @ C_G_Anderson : @ viva_b...
4	! ! ! ! ! ! ! ! ! ! ! ! ! RT @ ShenikaRoberts ...

```
def remove_punc(text):
    removed_text = ""
    for char in str(text.lower()):
        if char not in string.punctuation:
            removed_text+=char
    return removed_text
```

```
df['tweet'] = df['tweet'].apply(remove_punc)
df.head()
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class
\
0           0      3            0                   0        3      2

1           1      3            0                   3        0      1

2           2      3            0                   3        0      1

3           3      3            0                   2        1      1

4           4      6            0                   6        0      1


                                               tweet
0     rt  mayasolovely  as a woman you should nt ...
1       rt  mleew17  boy dats cold  tyga dwn bad ...
2         rt  urkindofbrand dawg     rt  80sbaby4...
3           rt  cganderson   vivabased she look l...
4               rt  shenikaroberts  the shit you ...
```

```
df['tweet'] = df['tweet'].apply(word_tokenize)
df.head()
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class
\
0           0      3            0                   0        3      2

1           1      3            0                   3        0      1

2           2      3            0                   3        0      1

3           3      3            0                   2        1      1

4           4      6            0                   6        0      1


                                               tweet
0  [rt, mayasolovely, as, a, woman, you, should, ...
1  [rt, mleew17, boy, dats, cold, tyga, dwn, bad,...
2  [rt, urkindofbrand, dawg, rt, 80sbaby4life, yo...
3  [rt, cganderson, vivabased, she, look, like, a...
4  [rt, shenikaroberts, the, shit, you, hear, abo...
```

```
def rem_stop(tokens):
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [token for token in tokens if token.lower() not
in stop_words]
    return filtered_tokens
```

```
df['tweet']= df['tweet'].apply(rem_stop)
df.head()
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class
\
0           0      3            0                   0        3      2

1           1      3            0                   3        0      1

2           2      3            0                   3        0      1

3           3      3            0                   2        1      1

4           4      6            0                   6        0      1


                                               tweet
0  [rt, mayasolovely, woman, nt, complain, cleani...
1  [rt, mleew17, boy, dats, cold, tyga, dwn, bad,...
2  [rt, urkindofbrand, dawg, rt, 80sbaby4life, ev...
3    [rt, cganderson, vivabased, look, like, tranny]
4  [rt, shenikaroberts, shit, hear, might, true, ...
```

```
def lemma_tokens(tokens):
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return tokens

df['tweet'] = df['tweet'].apply(lemma_tokens)
df.head()
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class
\
0           0      3            0                   0        3      2

1           1      3            0                   3        0      1

2           2      3            0                   3        0      1

3           3      3            0                   2        1      1

4           4      6            0                   6        0      1


                                               tweet
0  [rt, mayasolovely, woman, nt, complain, cleani...
1  [rt, mleew17, boy, dat, cold, tyga, dwn, bad, ...
2  [rt, urkindofbrand, dawg, rt, 80sbaby4life, ev...
3    [rt, cganderson, vivabased, look, like, tranny]
4  [rt, shenikaroberts, shit, hear, might, true, ...
```

```
df['pre_tweet'] = df['tweet'].apply(concat_text)
df.head()
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class
\
0           0      3            0                   0        3      2

1           1      3            0                   3        0      1

2           2      3            0                   3        0      1

3           3      3            0                   2        1      1

4           4      6            0                   6        0      1


                                                tweet  \
0  [rt, mayasolovely, woman, nt, complain, cleani...
1  [rt, mleew17, boy, dat, cold, tyga, dwn, bad, ...
2  [rt, urkindofbrand, dawg, rt, 80sbaby4life, ev...
3    [rt, cganderson, vivabased, look, like, tranny]
4  [rt, shenikaroberts, shit, hear, might, true, ...

                                           pre_tweet
0  rt mayasolovely woman nt complain cleaning hou...
1  rt mleew17 boy dat cold tyga dwn bad cuffin da...
2  rt urkindofbrand dawg rt 80sbaby4life ever fuc...
3            rt cganderson vivabased look like tranny
4  rt shenikaroberts shit hear might true might f...
```

```
df=df.iloc[:,[6,7,1,2,3,4,5]]
df.head()
```

```
                                                tweet  \
0  [rt, mayasolovely, woman, nt, complain, cleani...
1  [rt, mleew17, boy, dat, cold, tyga, dwn, bad, ...
2  [rt, urkindofbrand, dawg, rt, 80sbaby4life, ev...
3    [rt, cganderson, vivabased, look, like, tranny]
4  [rt, shenikaroberts, shit, hear, might, true, ...

                                           pre_tweet  count
hate_speech  \
0  rt mayasolovely woman nt complain cleaning hou...      3
0
1  rt mleew17 boy dat cold tyga dwn bad cuffin da...      3
0
2  rt urkindofbrand dawg rt 80sbaby4life ever fuc...      3
0
3            rt cganderson vivabased look like tranny      3
0
4  rt shenikaroberts shit hear might true might f...      6
```

```
0
```

	offensive_language	neither	class
0	0	3	2
1	3	0	1
2	3	0	1
3	2	1	1
4	6	0	1

Word2Vec

```
from gensim.models.word2vec import Word2Vec

sg = Word2Vec(df['pre_tweet'].values.tolist(), vector_size=150,
window=5, min_count=2, sg=1)
vocab = sg.wv.index_to_key
def get_mean_vector(model, sentence):
    words = [word for word in sentence if word in vocab]
    if len(words) >= 1:
        return np.mean(model.wv[words], axis=0)
    return np.zeros((150,))
sg_vector = []
for sentence in df['pre_tweet'].values.tolist():
    sg_vector.append(get_mean_vector(sg, sentence))
sg_vector = np.array(sg_vector)
sg_df= pd.DataFrame(sg_vector)
```

```
C:\Users\User\miniconda3\Lib\site-packages\paramiko\transport.py:219:
CryptographyDeprecationWarning: Blowfish has been deprecated
  "class": algorithms.Blowfish,
```

```
from sklearn.model_selection import train_test_split

Xw2v = pd.concat([sg_df, df.iloc[:,2:-1]], axis=1,
join='inner').values
Yw2v= df.iloc[:,-1]
x_trainw2v, x_testw2v, y_trainw2v, y_testw2v = train_test_split(Xw2v,
Yw2v,random_state=42, test_size=0.25)

from sklearn.naive_bayes import GaussianNB
nbw2v = GaussianNB()
nbw2v.fit(x_trainw2v,y_trainw2v)

y_predw2v= nbw2v.predict(x_testw2v)

from sklearn.metrics import accuracy_score

accuracy_score(y_testw2v,y_predw2v)
```

```
0.5468043899289864
```

FastText

```
from gensim.models import FastText
ft=FastText(df['pre_tweet'].values.tolist(),vector_size=100,window=5,m
in_count=1,workers=4)
def get_mean_vector(model, sentence):
    words = [word for word in sentence if word in vocab]
    if len(words) >= 1:
        return np.mean(model.wv[words], axis=0)
    return np.zeros((100,))
ft_vector = []
for sentence in df['pre_tweet'].values.tolist():
    ft_vector.append(get_mean_vector(ft, sentence))
ft_vector = np.array(ft_vector)
ft_df= pd.DataFrame(ft_vector)

from sklearn.model_selection import train_test_split

Xft = pd.concat([ft_df, df.iloc[:,2:-1]], axis=1, join='inner').values
Yft= df.iloc[:,-1]
x_trainft, x_testft, y_trainft, y_testft = train_test_split(Xft,
Yft,random_state=42, test_size=0.25)

from sklearn.naive_bayes import GaussianNB
nbft = GaussianNB()
nbft.fit(x_trainft,y_trainft)

y_predft= nbft.predict(x_testft)

from sklearn.metrics import accuracy_score

accuracy_score(y_testft,y_predft)

0.6826985151710782
```

CNN

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['pre_tweet'])
tokenized_text = tokenizer.texts_to_sequences(df['pre_tweet'])

max_seq_length = max(len(seq) for seq in tokenized_text)
```

```
padded_text = pad_sequences(tokenized_text, maxlen=max_seq_length,
padding='post')

from sklearn.model_selection import train_test_split

x_traincnn, x_testcnn, y_traincnn, y_testcnn =
train_test_split(padded_text, df['class'], test_size=0.25,
random_state=42)
x_traincnn = np.expand_dims(x_traincnn, axis=-1)
x_testcnn = np.expand_dims(x_testcnn, axis=-1)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D, Dense,
Flatten

model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(max_seq_length, 1)),
    GlobalMaxPooling1D(),
    Dense(128, activation='relu'),
    Dense(150)
])

model.compile(optimizer='adam', loss='mse')

model.fit(x_traincnn, y_traincnn, epochs=100, batch_size=32)

embedding_layer_output = model.predict(x_traincnn)
```

```
Epoch 1/100
581/581 [==============================] - 2s 1ms/step - loss:
6586.6890
Epoch 2/100
581/581 [==============================] - 1s 1ms/step - loss: 2.0892
Epoch 3/100
581/581 [==============================] - 1s 1ms/step - loss: 1.3927
Epoch 4/100
581/581 [==============================] - 1s 1ms/step - loss: 1.2197
Epoch 5/100
581/581 [==============================] - 1s 1ms/step - loss: 1.0780
Epoch 6/100
581/581 [==============================] - 1s 2ms/step - loss: 0.9361
Epoch 7/100
581/581 [==============================] - 1s 1ms/step - loss: 0.7962
Epoch 8/100
581/581 [==============================] - 1s 1ms/step - loss: 0.6639
Epoch 9/100
581/581 [==============================] - 1s 2ms/step - loss: 0.5455
Epoch 10/100
581/581 [==============================] - 1s 2ms/step - loss: 0.4454
```

```
Epoch 11/100
581/581 [==============================] - 1s 2ms/step - loss: 0.3659
Epoch 12/100
581/581 [==============================] - 1s 1ms/step - loss: 0.3068
Epoch 13/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2660
Epoch 14/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2401
Epoch 15/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2254
Epoch 16/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2180
Epoch 17/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2149
Epoch 18/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2137
Epoch 19/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 20/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 21/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 22/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 23/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 24/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2617
Epoch 25/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 26/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 27/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 28/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 29/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 30/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 31/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 32/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 33/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 34/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 35/100
```

```
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 36/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 37/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 38/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 39/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 40/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 41/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 42/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 43/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 44/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 45/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 46/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 47/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 48/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 49/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
Epoch 50/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 51/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 52/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 53/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2312
Epoch 54/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2144
Epoch 55/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 56/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 57/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 58/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2134
Epoch 59/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2134
```

```
Epoch 60/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 61/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 62/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 63/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 64/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 65/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 66/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 67/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2135
Epoch 68/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2560
Epoch 69/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 70/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 71/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 72/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 73/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 74/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 75/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 76/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 77/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 78/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 79/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 80/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 81/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 82/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2160
Epoch 83/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 84/100
```

```
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 85/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 86/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 87/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 88/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 89/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 90/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 91/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 92/100
581/581 [==============================] - 1s 2ms/step - loss: 0.2133
Epoch 93/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 94/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 95/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 96/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 97/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 98/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2247
Epoch 99/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
Epoch 100/100
581/581 [==============================] - 1s 1ms/step - loss: 0.2133
581/581 [==============================] - 1s 732us/step
```

```
from sklearn.naive_bayes import GaussianNB
nbcnn = GaussianNB()
nbcnn.fit(embedding_layer_output,y_traincnn)

y_predcnn= nbcnn.predict(model.predict(x_testcnn))
```

```
194/194 [==============================] - 0s 753us/step
```

```
from sklearn.metrics import accuracy_score

accuracy_score(y_testcnn,y_predcnn)
```

```
0.06020012911555842
```

RNN

```
vocab_size = len(tokenizer.word_index) + 1

model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=150,
input_length=max_seq_length),
    LSTM(units=64)
])

model.compile(optimizer='adam', loss='mse')

model.fit(x_traincnn, y_traincnn, epochs=10, batch_size=32)

embedding_layer_output = model.predict(x_traincnn)
```

```
Epoch 1/10
581/581 [==============================] - 55s 90ms/step - loss:
0.2657
Epoch 2/10
581/581 [==============================] - 51s 88ms/step - loss:
0.2259
Epoch 3/10
581/581 [==============================] - 51s 87ms/step - loss:
0.2256
Epoch 4/10
581/581 [==============================] - 49s 85ms/step - loss:
0.2256
Epoch 5/10
581/581 [==============================] - 49s 84ms/step - loss:
0.2255
Epoch 6/10
581/581 [==============================] - 49s 84ms/step - loss:
0.2255
Epoch 7/10
581/581 [==============================] - 49s 84ms/step - loss:
0.2255
Epoch 8/10
581/581 [==============================] - 49s 84ms/step - loss:
0.2255
Epoch 9/10
581/581 [==============================] - 49s 84ms/step - loss:
0.2255
Epoch 10/10
581/581 [==============================] - 49s 85ms/step - loss:
0.2255
581/581 [==============================] - 6s 9ms/step
```

```
from sklearn.naive_bayes import GaussianNB
nbrnn = GaussianNB()
```

```
nbrnn.fit(embedding_layer_output,y_traincnn)

y_predrnn= nbrnn.predict(model.predict(x_testcnn))

194/194 [==============================] - 2s 9ms/step

from sklearn.metrics import accuracy_score

accuracy_score(y_testcnn,y_predrnn)

0.05842479018721756
```