BBC_News_Text_Classification

May 13, 2023

1 BBC News Text Classification

1.1 Background

In this machine learning project, the overall topic that will be resolved is in the field of news classification, where it will try to predict the news category whether it's a business, entertainment, politics, sports, or tech topic based on the text news.

1.1.1 1. Import the required libraries

[13]:
```
# library for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud

# library for data processing
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# library for modeling
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dropout, Dense,␣
↪SpatialDropout1D
```

1.1.2 2. Download and preprocess dataset

[14]:
```
# Download the dataset with Kaggle CLI
!kaggle datasets download -d balatmak/newsgroup20bbcnews
```

```
'kaggle' is not recognized as an internal or external command,
operable program or batch file.
```

1.2 3. Data Understanding

1.2.1 3.1 Read dataset with pandas

[15]:
```
# Read dataset with pandas
news_df = pd.read_csv(r'D:\Data Science Course\My Project for Data\BBC News Text Classificationn\bbc-text.CSV')
news_df
```

[15]:

	category	text
0	tech	tv future in the hands of viewers with home th…
1	business	worldcom boss left books alone former worldc…
2	sport	tigers wary of farrell gamble leicester say …
3	sport	yeading face newcastle in fa cup premiership s…
4	entertainment	ocean s twelve raids box office ocean s twelve…
…	…	…
2220	business	cars pull down us retail figures us retail sal…
2221	politics	kilroy unveils immigration policy ex-chatshow …
2222	entertainment	rem announce new glasgow concert us band rem h…
2223	politics	how political squabbles snowball it s become c…
2224	sport	souness delight at euro progress boss graeme s…

[2225 rows x 2 columns]

[16]:
```
print(news_df['text'][1])
```

```
worldcom boss  left books alone  former worldcom boss bernie ebbers  who is
accused of overseeing an $11bn (£5.8bn) fraud  never made accounting decisions
a witness has told jurors.  david myers made the comments under questioning by
defence lawyers who have been arguing that mr ebbers was not responsible for
worldcom s problems. the phone company collapsed in 2002 and prosecutors claim
that losses were hidden to protect the firm s shares. mr myers has already
pleaded guilty to fraud and is assisting prosecutors.  on monday  defence lawyer
reid weingarten tried to distance his client from the allegations. during cross
examination  he asked mr myers if he ever knew mr ebbers  make an accounting
decision  .  not that i am aware of   mr myers replied.  did you ever know mr
ebbers to make an accounting entry into worldcom books   mr weingarten pressed.
no   replied the witness. mr myers has admitted that he ordered false accounting
entries at the request of former worldcom chief financial officer scott
sullivan. defence lawyers have been trying to paint mr sullivan  who has
admitted fraud and will testify later in the trial  as the mastermind behind
worldcom s accounting house of cards.  mr ebbers  team  meanwhile  are looking
to portray him as an affable boss  who by his own admission is more pe graduate
than economist. whatever his abilities  mr ebbers transformed worldcom from a
relative unknown into a $160bn telecoms giant and investor darling of the late
1990s. worldcom s problems mounted  however  as competition increased and the
telecoms boom petered out. when the firm finally collapsed  shareholders lost
about $180bn and 20 000 workers lost their jobs. mr ebbers  trial is expected to
last two months and if found guilty the former ceo faces a substantial jail
```

```
sentence. he has firmly declared his innocence.
```

1.2.2 3.2 Explore dataset information

[17]:
```
news_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2225 entries, 0 to 2224
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   category  2225 non-null   object
 1   text      2225 non-null   object
dtypes: object(2)
memory usage: 34.9+ KB
```

[18]:
```
# check for missing values
news_df.isna().sum()
```

[18]:
```
category    0
text        0
dtype: int64
```

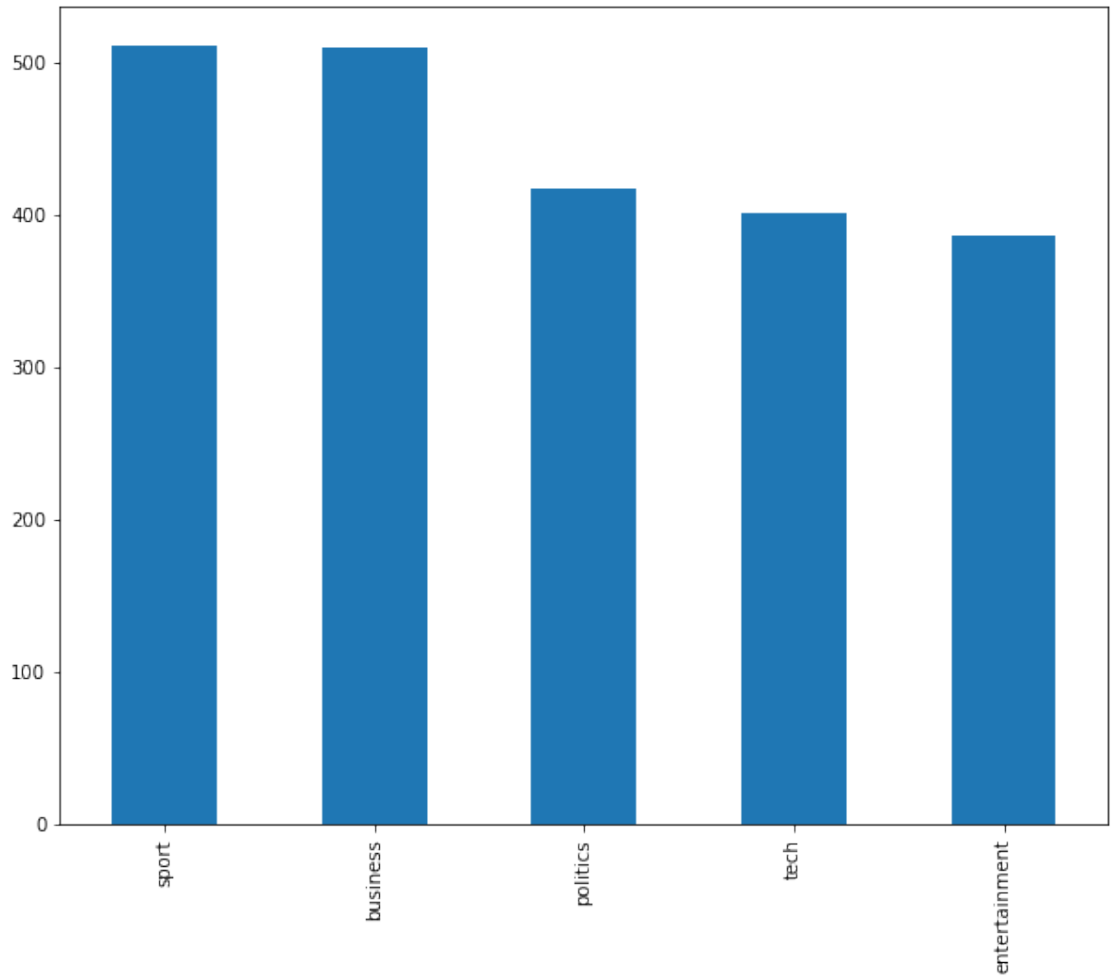
[19]:
```
# check for duplicate row
news_df.duplicated().sum()
```

[19]:
```
99
```

1.2.3 3.3 Data visualization

[20]:
```
news_df['category'].value_counts().plot(kind='bar', figsize=(10, 8))
```

[20]:
```
<AxesSubplot:>
```



1.2.4 3.4 Clean text from stopwords and symbols

[21]:
```
# download nltk stopwords

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\ganna
[nltk_data]     center\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[21]: True

[22]:
```
# create function to clean text from stopwords and symbols using regex and nltk

space = re.compile('[/(){}\[\]\|@,;]')
symbols= re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))
```

```
def clean_text(text):
    text = text.lower()
    text = space.sub(' ', text)
    text = symbols.sub('', text)
    text = text.replace('x', '')
    text = ' '.join(word for word in text.split() if word not in STOPWORDS) #␣
 ↪remove stopwors from text
    return text
```

[23]:
```
# applying function to pandas df

news_df['text'] = news_df['text'].apply(clean_text)
```

[ ]:
```

```

1.3 4. Data Preparation

1.3.1 4.1 Clean duplicated and unecessary word in data

[35]:
```
news_df = news_df.drop_duplicates()
```

[36]:
```
news_df.duplicated().sum()
```

[36]:
```
0
```

[37]:
```
news_df['text'] = [re.sub('said', '', x) for x in news_df['text']]
```

```
C:\Users\ganna center\AppData\Local\Temp\ipykernel_11360\3874003685.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  news_df['text'] = [re.sub('said', '', x) for x in news_df['text']]
```

1.3.2 4.2 Using one hot encoding for category column

[38]:
```
category = pd.get_dummies(news_df['category'])
news_fixed = pd.concat([news_df, category], axis=1)
news_fixed = news_fixed.drop(columns='category')
news_fixed
```

[38]:

	text	business	\
0	tv future hands viewers home theatre systems p…	0	
1	worldcom boss left books alone former worldcom…	1	
2	tigers wary farrell gamble leicester say rushe…	0	

```
3     yeading face newcastle fa cup premiership side…         0
4     ocean twelve raids bo office ocean twelve crim…         0
…                                                     …       …
2220  cars pull us retail figures us retail sales fe…         1
2221  kilroy unveils immigration policy echatshow ho…         0
2222  rem announce new glasgow concert us band rem a…         0
2223  political squabbles snowball become commonplac…         0
2224  souness delight euro progress boss graeme soun…         0

      entertainment  politics  sport  tech
0                 0         0      0     1
1                 0         0      0     0
2                 0         0      1     0
3                 0         0      1     0
4                 1         0      0     0
…               …         …      …     …
2220              0         0      0     0
2221              0         1      0     0
2222              1         0      0     0
2223              0         1      0     0
2224              0         0      1     0

[2120 rows x 6 columns]
```

1.3.3 4.3 Split train and test data

[39]:
```
text = news_fixed['text'].values
label = news_fixed[['business', 'entertainment', 'politics', 'sport', 'tech']].
 ↪values
```

[40]:
```
# Train test split data

text_train, text_test, label_train, label_test = train_test_split(text, label,␣
 ↪test_size=0.2)
```

1.3.4 4.4 Pre-modeling steps

[41]:
```
# set the necessary variables

vocab_size = 50000
embedding_dim = 100
max_length = 3000
trunc_type='post'
oov_tok = "<OOV>"
```

[42]:
```
# using text preprocessing tokenizer and sequence preprocessing padsequences
```

```
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(text_train)
tokenizer.fit_on_texts(text_test)

train_sequences = tokenizer.texts_to_sequences(text_train)
test_sequences = tokenizer.texts_to_sequences(text_test)

train_padsequences = pad_sequences(train_sequences, maxlen=max_length,␣
 ↪truncating=trunc_type)
test_padsequences = pad_sequences(test_sequences, maxlen=max_length)
```

1.4 5. Modeling

1.4.1 5.1 Using Sequential model

[43]:
```
# using sequential model with embedding, LSTM, and Dense layers

model = Sequential([
                    Embedding(input_dim=vocab_size, output_dim=embedding_dim,␣
 ↪input_length=max_length),
                    SpatialDropout1D(0.2),
                    LSTM(64, dropout=0.4, recurrent_dropout=0.3),
                    Dropout(0.5),
                    Dense(5, activation='softmax')
                    ])
```

1.4.2 5.2 Compile model

[44]:
```
# Compile model

model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

1.4.3 5.3 Create callback class

[45]:
```
# Add calbacks on_epoch_end

class myCallback(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('accuracy')>0.96 and logs.get('val_accuracy')>0.96):
      print("\nModel accuracy and validation accuracy > 96%!")
      self.model.stop_training = True
callbacks = myCallback()
```

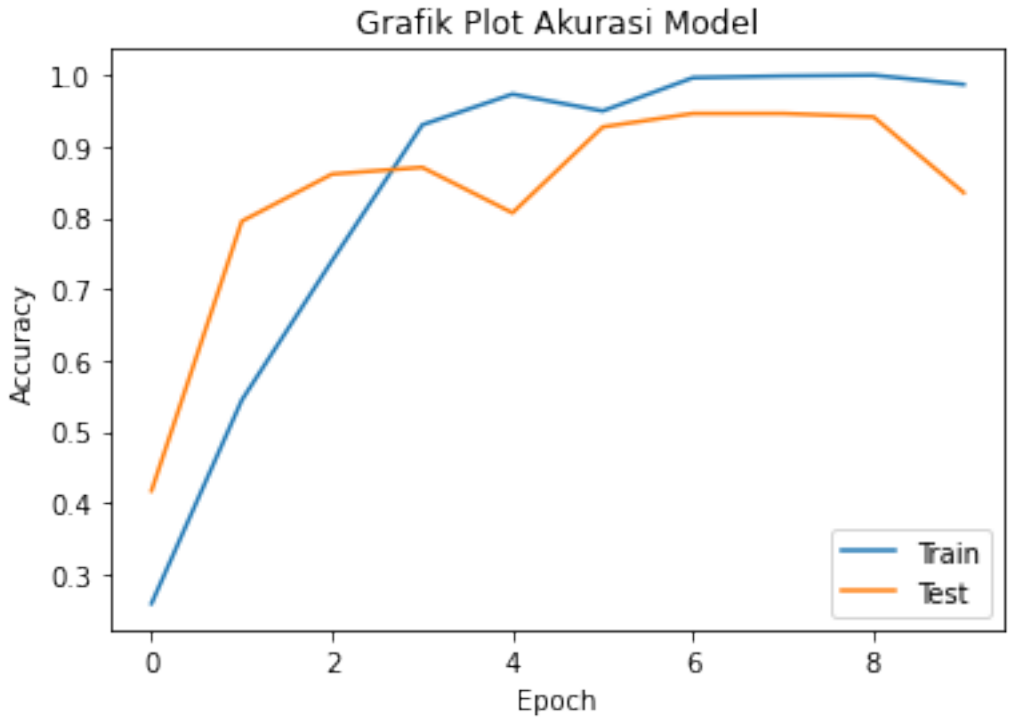
1.4.4 5.4 Fit model

[46]:
```
# Fit model

hist = model.fit(
    train_padsequences,
    label_train,
    epochs=10,
    validation_data=(test_padsequences, label_test),
    callbacks=[callbacks]
)
```

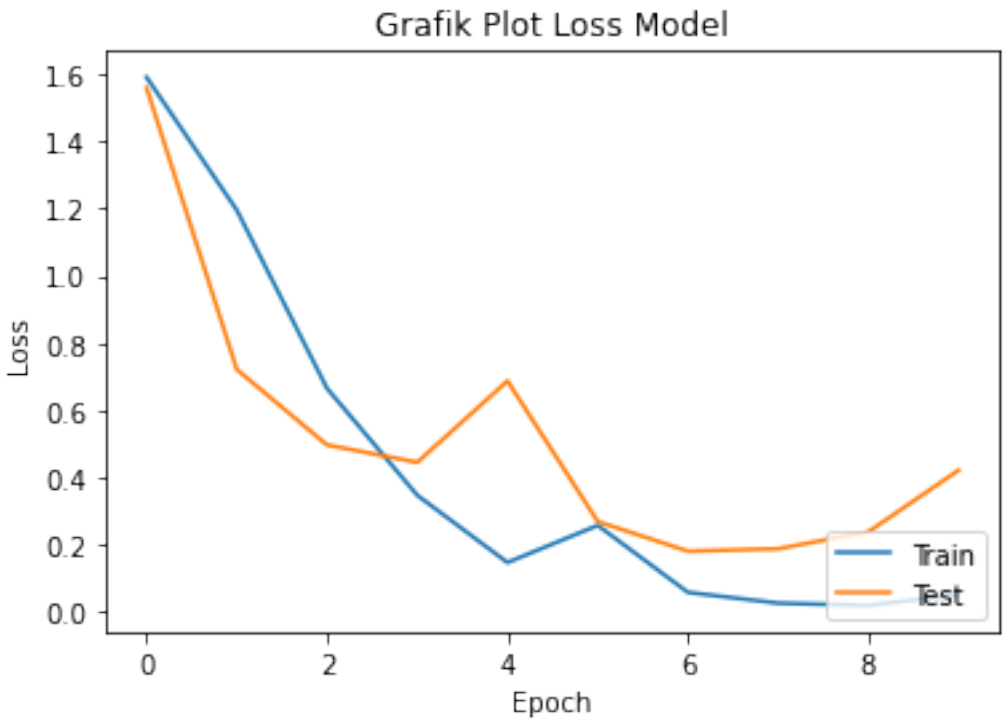
```
Epoch 1/10
53/53 [==============================] - 110s 2s/step - loss: 1.5919 - accuracy:
0.2594 - val_loss: 1.5607 - val_accuracy: 0.4175
Epoch 2/10
53/53 [==============================] - 106s 2s/step - loss: 1.1960 - accuracy:
0.5442 - val_loss: 0.7213 - val_accuracy: 0.7948
Epoch 3/10
53/53 [==============================] - 106s 2s/step - loss: 0.6649 - accuracy:
0.7394 - val_loss: 0.4964 - val_accuracy: 0.8608
Epoch 4/10
53/53 [==============================] - 107s 2s/step - loss: 0.3464 - accuracy:
0.9298 - val_loss: 0.4445 - val_accuracy: 0.8703
Epoch 5/10
53/53 [==============================] - 107s 2s/step - loss: 0.1460 - accuracy:
0.9729 - val_loss: 0.6870 - val_accuracy: 0.8066
Epoch 6/10
53/53 [==============================] - 107s 2s/step - loss: 0.2566 - accuracy:
0.9493 - val_loss: 0.2680 - val_accuracy: 0.9269
Epoch 7/10
53/53 [==============================] - 106s 2s/step - loss: 0.0569 - accuracy:
0.9959 - val_loss: 0.1797 - val_accuracy: 0.9458
Epoch 8/10
53/53 [==============================] - 106s 2s/step - loss: 0.0249 - accuracy:
0.9982 - val_loss: 0.1869 - val_accuracy: 0.9458
Epoch 9/10
53/53 [==============================] - 108s 2s/step - loss: 0.0180 - accuracy:
0.9994 - val_loss: 0.2376 - val_accuracy: 0.9410
Epoch 10/10
53/53 [==============================] - 106s 2s/step - loss: 0.0511 - accuracy:
0.9864 - val_loss: 0.4213 - val_accuracy: 0.8349
```

1.5 6. Model Evaluation

[49]:
```
# Plot accuracy and loss model

plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Grafik Plot Akurasi Model')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='lower right')
plt.show()
```



[50]:
```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Grafik Plot Loss Model')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='lower right')
plt.show()
```



[51]:
```
# create function to predict text
labels = ['Business','Entertainment','Politics','Sports','Tech']

def predictText(text):
  texts = map(clean_text, text) # apply clean text function
  seq = tokenizer.texts_to_sequences(texts)
  padded = pad_sequences(seq, maxlen=max_length)
  pred = model.predict(padded)
  df = pd.DataFrame({'category' : labels, 'percentage' : pred[0]})
  print(df)
  print('\nThe text is classified as', labels[np.argmax(pred)], 'category')
```

[52]:
```
```

```
news = ['It was announced on Tuesday that Disney Channel movie with tourmate␣
 ↪Demi Lovato, "Camp Rock 2: The Final Jam," will premiere on September 3 at 8␣
 ↪p.m. ET. On July 27, long before they watch the sequel to the 2008 flick,␣
 ↪fans can pick up the soundtrack, featuring 15 original songs that a press␣
 ↪release promises will span genres from hip-hop to rock to pop. The flick␣
 ↪will not only have more summer lovin\' between real-life couple Lovato and␣
 ↪Joe Jonas as Mitchie and Shane, but there will also be a little friendly␣
 ↪rivalry between the Camp Rockers and a group of musicians at another summer␣
 ↪camp, Camp Star, including a love interest for Nick Jonas, played by Chloe␣
 ↪Bridges. The JoBros promise the movie\'s music will be every bit as␣
 ↪entertaining as its plot, which has been kept a secret since the movie was␣
 ↪shot. "The songs are really cool," Joe told MTV News.']
predictText(news)
```

```
1/1 [==============================] - 0s 319ms/step
        category  percentage
0       Business    0.031390
1  Entertainment    0.895199
2       Politics    0.019047
3         Sports    0.031809
4           Tech    0.022555

The text is classified as Entertainment category
```

From the result of the prediction above, the model has alreaady achieved a good accuracy that the news has the correct prediction

2 Create a GUI

[53]:
```
# Import required libraries
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
from tkinter import *
from tkinter import filedialog

# load the dataset
df = pd.read_csv(r'D:\Data Science Course\My Project for Data\BBC News Text␣
 ↪Classificationn\bbc-text.CSV')

# preprocess the text
```

```
nltk.download('stopwords')
stopwords_english = stopwords.words('english')
stemmer = SnowballStemmer('english')

def preprocess(text):
    text = text.lower()
    text = " ".join([word for word in text.split() if word not in␣
 ↪stopwords_english])
    text = " ".join([stemmer.stem(word) for word in text.split()])
    return text

df['text'] = df['text'].apply(preprocess)

# convert text into a bag of words
count_vectorizer = CountVectorizer()
X = count_vectorizer.fit_transform(df['text'])

# split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, df['category'],␣
 ↪test_size=0.2, random_state=42)

# train a machine learning model using Naive Bayes
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)

# evaluate the performance of the model
y_pred = naive_bayes.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# create a GUI to allow user input
root = Tk()

def predict_category():
    text = entry.get()
    text = preprocess(text)
    X = count_vectorizer.transform([text])
    prediction = naive_bayes.predict(X)
    label.config(text="Category: {}".format(prediction[0]))

label1 = Label(root, text="BBC News Text Classification")
label1.pack()

entry = Entry(root)
entry.pack()
```

```
button = Button(root, text="Predict", command=predict_category)
button.pack()

label = Label(root, text="")
label.pack()

root.mainloop()

print('Accuracy: {:.2f}%'.format(accuracy*100))
print('Precision: {:.2f}%'.format(precision*100))
print('Recall: {:.2f}%'.format(recall*100))
print('F1 Score: {:.2f}%'.format(f1*100))
```

```
[nltk_data] Downloading package stopwords to C:\Users\ganna
[nltk_data]     center\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
Accuracy: 96.63%
Precision: 96.77%
Recall: 96.63%
F1 Score: 96.62%
```

[ ]: