In [1]:
```
import pandas as pd
import numpy as np

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

import re
import unicodedata
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

import torch
import torch.nn as nn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from transformers import BertTokenizer, BertForSequenceClassification, AdamW

import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score

from sklearn.svm import SVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report, accuracy_score
from sklearn.utils import shuffle

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
d1 = pd.read_csv(r"Scraped_Car_Review_maserati.csv")
d2 = pd.read_csv(r"Scraped_Car_Review_fiat.csv")
d3 = pd.read_csv(r"Scraped_Car_Review_lotus.csv")
d4 = pd.read_csv(r"Scraped_Car_Review_genesis.csv")
d5 = pd.read_csv(r"Scraped_Car_Review_maybach.csv")
d6 = pd.read_csv(r"Scraped_Car_Review_maserati.csv")
```

In [3]:
```python
# Merge the datasets vertically (concatenate rows)
data = pd.concat([d1, d2, d3, d4, d5, d6], ignore_index=True)

# Print the merged dataset
data
```

Out[3]:

	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating
0	0	on 01/01/05 19:28 PM (PST)	Jeff	2004 Maserati Spyder Convertible Cambiocorsa 2...	Maserati Thrill Ride	I bought my Maserati Spyder Cambiocorsa \rin ...	4.875
1	1	on 12/05/04 06:07 AM (PST)	Francie	2004 Maserati Spyder Convertible Cambiocorsa 2...	My second Spyder	I got the very first Maserati Spyder delivere...	4.750
2	2	on 12/04/04 15:28 PM (PST)	West8	2004 Maserati Spyder Convertible GT 2dr Conver...	Fast & Fun	After owning several comparable cars in the p...	4.375
3	3	on 10/28/04 00:00 AM (PDT)	KLAMIA2001	2004 Maserati Spyder Convertible Cambiocorsa 2...	ITALIANS RULE	THIS IS MY THIRD MASERATI, A 2002, 2003 \r& A...	5.000
4	4	on 09/22/04 00:00 AM (PDT)	d blair	2004 Maserati Spyder Convertible Cambiocorsa 2...	Midlife Crisis?? I don't think so!	Ok so I am a 55 year old man who has	NaN
...	...	...	...	...	...	...	...
1262	234	on 09/04/02 00:00 AM (PDT)	Jonathan Epstein	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	A gorgeous Italian with Ferrari blood	A gorgeous car on the exterior with the\rbest...	4.500
1263	235	on 08/31/02 00:00 AM (PDT)	calabro	2002 Maserati Coupe Coupe GT 2dr Coupe (4.2L 8...	Awesom	Excelllent all around car,,Finally an \rItali...	4.750
1264	236	on 08/26/02 00:00 AM (PDT)	adam tool	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	A return of a classic	Plain and simple, fast as hell and you \rwon'...	5.000
1265	237	on 07/31/02 00:00 AM (PDT)	Woodman	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	Italy at its Best	This car has been fantastic, a real joy to dr...	4.875
1266	238	on 05/19/02 00:00 AM (PDT)	zerotoninetyfast	2002 Maserati Coupe Coupe GT 2dr Coupe (4.2L 8...	Grown-up Performance ! ! !	So far, this car has been quite \rimpressive!...	4.500

1267 rows × 7 columns

In [4]:
```python
# Check for null values in the entire DataFrame
null_values = data.isnull().sum()

# Print the count of null values for each column
null_values
```

Out[4]:
```
Unnamed: 0        1
Review_Date     120
Author_Name     155
Vehicle_Title   155
Review_Title    155
Review          155
Rating          190
dtype: int64
```

In [5]:
```python
# Fill null values with a specific value, for example, 0
data = data.fillna(0)

# Alternatively, you can fill null values with the mean of each column
df_filled_mean = data.fillna(data.mean())

# Print the DataFrame with filled values
data
df_filled_mean
```

Out[5]:

	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating
0	0	on 01/01/05 19:28 PM (PST)	Jeff	2004 Maserati Spyder Convertible Cambiocorsa 2...	Maserati Thrill Ride	I bought my Maserati Spyder Cambiocorsa \rin ...	4.875
1	1	on 12/05/04 06:07 AM (PST)	Francie	2004 Maserati Spyder Convertible Cambiocorsa 2...	My second Spyder	I got the very first Maserati Spyder delivere...	4.750
2	2	on 12/04/04 15:28 PM (PST)	West8	2004 Maserati Spyder Convertible GT 2dr Conver...	Fast & Fun	After owning several comparable cars in the p...	4.375
3	3	on 10/28/04 00:00 AM (PDT)	KLAMIA2001	2004 Maserati Spyder Convertible Cambiocorsa 2...	ITALIANS RULE	THIS IS MY THIRD MASERATI, A 2002, 2003 \r& A...	5.000
4	4	on 09/22/04 00:00 AM (PDT)	d blair	2004 Maserati Spyder Convertible Cambiocorsa 2...	Midlife Crisis?? I don't think so!	Ok so I am a 55 year old man who has	0.000
...	...	...	...	...	...	...	...
1262	234	on 09/04/02 00:00 AM (PDT)	Jonathan Epstein	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	A gorgeous Italian with Ferrari blood	A gorgeous car on the exterior with the\rbest...	4.500
1263	235	on 08/31/02 00:00 AM (PDT)	calabro	2002 Maserati Coupe Coupe GT 2dr Coupe (4.2L 8...	Awesom	Excelllent all around car,,Finally an \rItali...	4.750
1264	236	on 08/26/02 00:00 AM (PDT)	adam tool	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	A return of a classic	Plain and simple, fast as hell and you \rwon'...	5.000
1265	237	on 07/31/02 00:00 AM (PDT)	Woodman	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	Italy at its Best	This car has been fantastic, a real joy to dr...	4.875
1266	238	on 05/19/02 00:00 AM (PDT)	zerotoninetyfast	2002 Maserati Coupe Coupe GT 2dr Coupe (4.2L 8...	Grown-up Performance ! ! !	So far, this car has been quite \rimpressive!...	4.500

1267 rows × 7 columns

In [6]:
```python
data = data.drop(columns=['Unnamed: 0'])

# Print the DataFrame after dropping the "Unnamed" column
data
```

Out[6]:

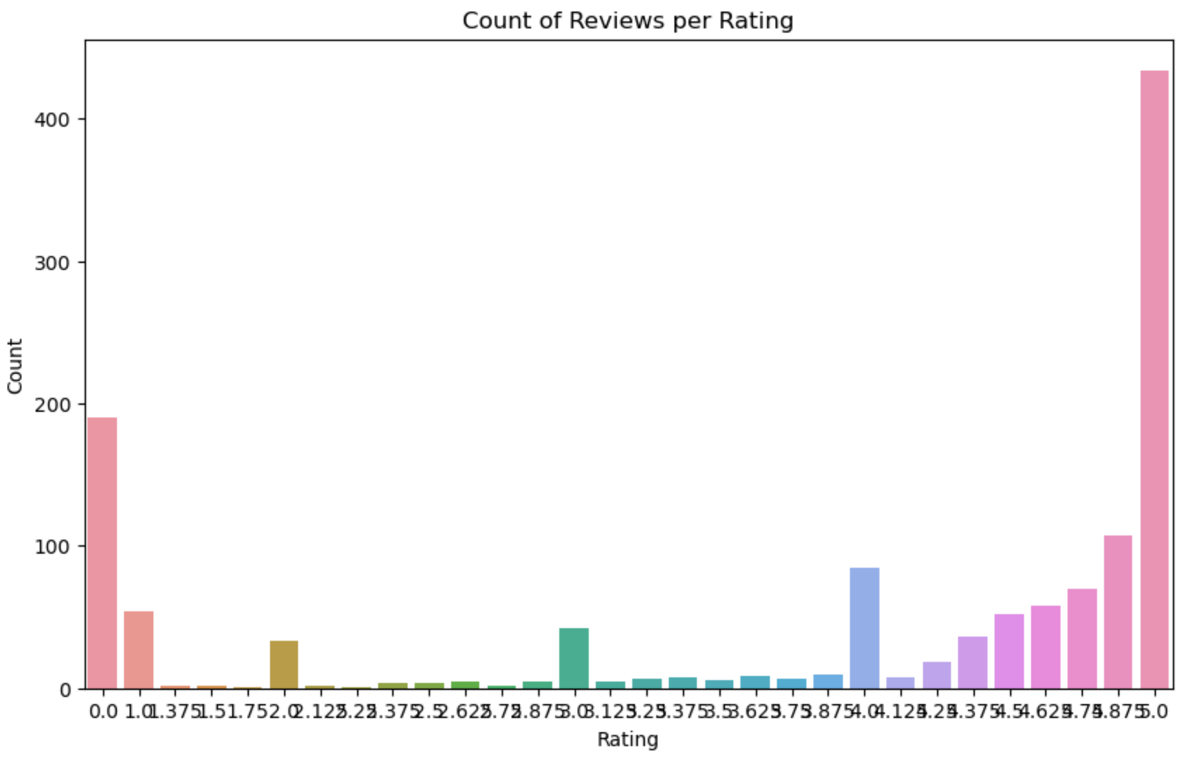
	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating
0	on 01/01/05 19:28 PM (PST)	Jeff	2004 Maserati Spyder Convertible Cambiocorsa 2...	Maserati Thrill Ride	I bought my Maserati Spyder Cambiocorsa \rin ...	4.875
1	on 12/05/04 06:07 AM (PST)	Francie	2004 Maserati Spyder Convertible Cambiocorsa 2...	My second Spyder	I got the very first Maserati Spyder delivere...	4.750
2	on 12/04/04 15:28 PM (PST)	West8	2004 Maserati Spyder Convertible GT 2dr Conver...	Fast & Fun	After owning several comparable cars in the p...	4.375
3	on 10/28/04 00:00 AM (PDT)	KLAMIA2001	2004 Maserati Spyder Convertible Cambiocorsa 2...	ITALIANS RULE	THIS IS MY THIRD MASERATI, A 2002, 2003 \r& A...	5.000
4	on 09/22/04 00:00 AM (PDT)	d blair	2004 Maserati Spyder Convertible Cambiocorsa 2...	Midlife Crisis?? I don't think so!	Ok so I am a 55 year old man who has	0.000
...	...	...	...	...	...	...
1262	on 09/04/02 00:00 AM (PDT)	Jonathan Epstein	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	A gorgeous Italian with Ferrari blood	A gorgeous car on the exterior with the\rbest...	4.500
1263	on 08/31/02 00:00 AM (PDT)	calabro	2002 Maserati Coupe Coupe GT 2dr Coupe (4.2L 8...	Awesom	Excelllent all around car,,Finally an \rItali...	4.750
1264	on 08/26/02 00:00 AM (PDT)	adam tool	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	A return of a classic	Plain and simple, fast as hell and you \rwon'...	5.000
1265	on 07/31/02 00:00 AM (PDT)	Woodman	2002 Maserati Coupe Coupe Cambiocorsa 2dr Coup...	Italy at its Best	This car has been fantastic, a real joy to dr...	4.875
1266	on 05/19/02 00:00 AM (PDT)	zerotoninetyfast	2002 Maserati Coupe Coupe GT 2dr Coupe (4.2L 8...	Grown-up Performance ! ! !	So far, this car has been quite \rimpressive!...	4.500

1267 rows × 6 columns

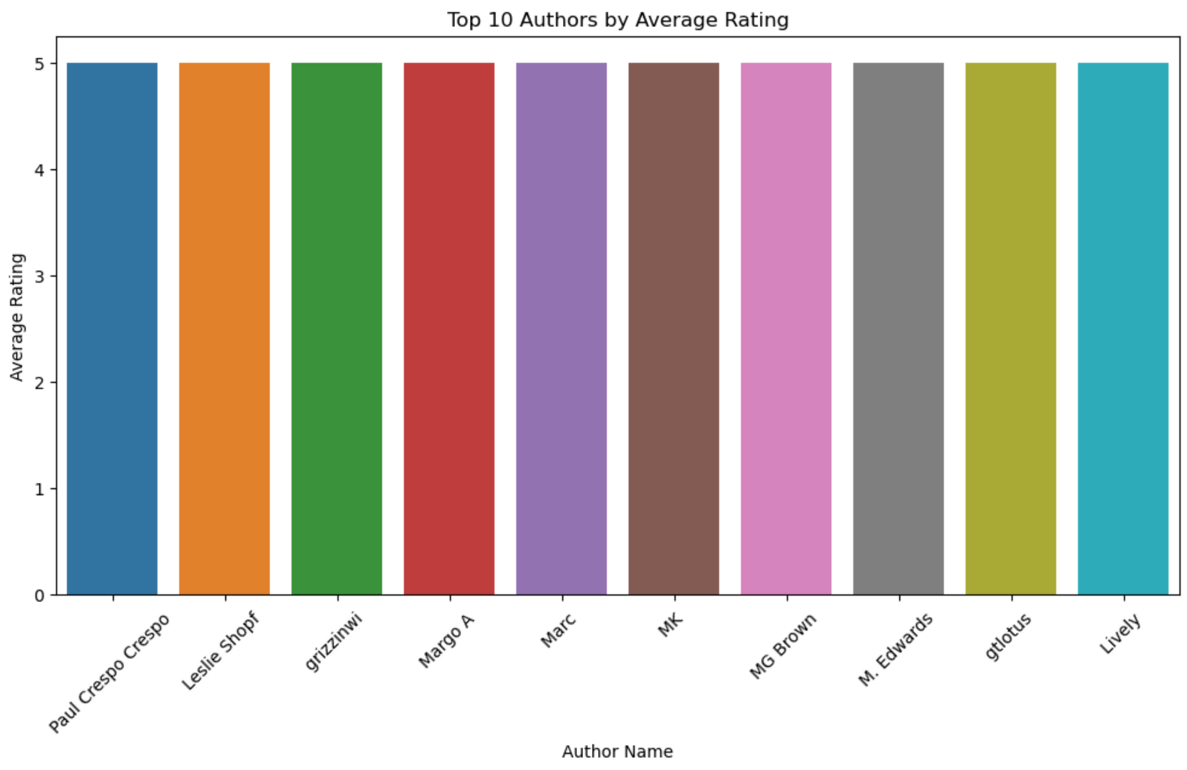
In [7]:
```python
data.dtypes
```

Out[7]:
```
Review_Date       object
Author_Name       object
Vehicle_Title     object
Review_Title      object
Review            object
Rating           float64
dtype: object
```

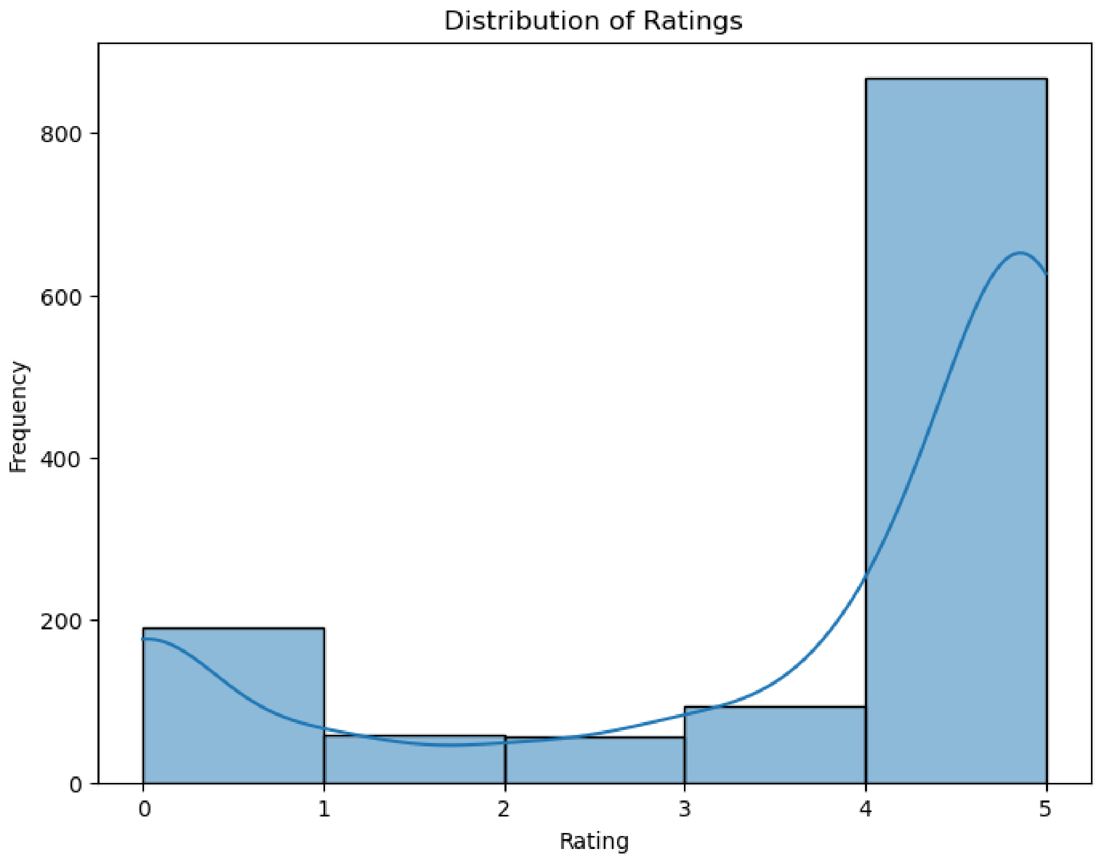
In [8]:
```python
# Count of reviews per rating
plt.figure(figsize=(10, 6))
sns.countplot(x='Rating', data=data)
plt.title('Count of Reviews per Rating')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



In [9]:
```python
# Average rating distribution by author
plt.figure(figsize=(12, 6))
sns.barplot(x='Author_Name', y='Rating', data=data.groupby('Author_Name')['Rating'].mean().reset_i
plt.title('Top 10 Authors by Average Rating')
plt.xlabel('Author Name')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



In [10]:
```
# Distribution of ratings
plt.figure(figsize=(8, 6))
sns.histplot(data['Rating'], bins=5, kde=True)
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



Text Data Preprocessing

In [11]:
```
stopwords = stopwords.words('english')
unnecessary_stopwords = {'no', 'nor', 'not', 'ain', 'aren', "aren't", 'couldn', 'what', 'which', '
                         'whom',
                         'why', 'how', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', '
                         'hasn',
                         "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
                         "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
                         "wasn't",
                         'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", 'don', "don't"}

stopwords = [word for word in stopwords if word not in unnecessary_stopwords]
lemma = WordNetLemmatizer()
nouns = ['NN', 'NNS']  # Using NLTK's part-of-speech tags for nouns
labelencoder = LabelEncoder()
```

In [12]:
```
# Define a function to clean the text
def text_cleaner(text, remove_stopwords=True, lemmatize=True):
    import nltk  # Import nltk here

    tokens = nltk.word_tokenize(text)  # Use nltk.word_tokenize instead of word_tokenize
    if lemmatize:
        lemmatizer = nltk.WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(token) for token in tokens]
    if remove_stopwords:
        tokens = [token for token in tokens if token.lower() not in nltk.corpus.stopwords.words('e
    cleaned_text = " ".join(tokens)
    return cleaned_text

    # Removing unnecessary stopwords
    if remove_stopwords:
        text = word_tokenize(text)
        text = " ".join([word for word in text if word not in stopwords.words('english')])

    # Word Lemmatization
    if lemmatize:
        lemmatizer = WordNetLemmatizer()
        text = word_tokenize(text)
        text = " ".join([lemmatizer.lemmatize(word).lower() for word in text if lemmatizer.lemmati

    return text
```

In [13]:
```
def custom_vectorize(x_train, x_test, use_char=False, max_features=10000, ngram_range=(1, 1)):
    if use_char:
        vectorizer = CountVectorizer(analyzer='char', max_features=max_features, ngram_range=ngram
    else:
        vectorizer = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}', max_features=max_fe

    vectorizer.fit(x_train)

    count_x_train = vectorizer.transform(x_train)
    count_x_test = vectorizer.transform(x_test)

    return count_x_train, count_x_test
```

In [14]:
```
def custom_tfidf_vectorize(x_train, x_test, use_char=False, max_features=10000, ngram_range=(1, 1)
    if use_char:
        vectorizer = TfidfVectorizer(analyzer='char', max_features=max_features, ngram_range=ngram
    else:
        vectorizer = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}', max_features=max_fe

    vectorizer.fit(x_train)

    x_train_tfidf = vectorizer.transform(x_train)
    x_test_tfidf = vectorizer.transform(x_test)

    return x_train_tfidf, x_test_tfidf
```

In [15]:
```
# Define the preprocess_text function
def preprocess_text(text, remove_stopwords=True, lemmatize=True):
    cleaned_text = text_cleaner(str(text), remove_stopwords=remove_stopwords, lemmatize=lemmatize)
    return cleaned_text
```

```
In [16]: def prepare_data(data, test_size=0.3, remove_stopwords=True, lemmatize=True, min_class_count=2):
             data = data.sample(frac=1).reset_index(drop=True)
             descriptions = data['Review'].apply(preprocess_text, remove_stopwords=remove_stopwords, lemmat
             labels = data['Review_Title'].values.tolist()
             encoded_labels = labelencoder.fit_transform(labels)

             # Calculate the class counts
             class_counts = np.bincount(encoded_labels)

             # Identify classes with sufficient instances
             valid_classes = np.where(class_counts >= min_class_count)[0]

             # Filter data for valid classes
             valid_indices = [i for i in range(len(encoded_labels)) if encoded_labels[i] in valid_classes]
             descriptions = descriptions[valid_indices]
             encoded_labels = encoded_labels[valid_indices]

             # Create a dictionary to store descriptions for each class
             class_descriptions_dict = {class_idx: [] for class_idx in valid_classes}
             for i, class_idx in enumerate(encoded_labels):
                 class_descriptions_dict[class_idx].append(descriptions.iloc[i])

             # Split the data into training and test sets
             x_train, y_train, x_test, y_test = [], [], [], []
             for class_idx, class_descriptions in class_descriptions_dict.items():
                 num_instances_in_test = int(len(class_descriptions) * test_size)

                 # Shuffle the instances for this class
                 class_descriptions = shuffle(class_descriptions, random_state=42)

                 # Split into training and test sets
                 x_train.extend(class_descriptions[num_instances_in_test:])
                 y_train.extend([class_idx] * (len(class_descriptions) - num_instances_in_test))
                 x_test.extend(class_descriptions[:num_instances_in_test])
                 y_test.extend([class_idx] * num_instances_in_test)

             return x_train, x_test, y_train, y_test

         # Now you can use the prepare_data function
         x_train, x_test, y_train, y_test = prepare_data(data, test_size=0.3, remove_stopwords=True, lemmat

         print("Total training examples: ", len(x_train))
         print("Total test examples: ", len(x_test))
```

```
Total training examples:  614
Total test examples:  53
```

SVM

```
In [17]: # Vectorize your text data using TF-IDF
         tfidf_vectorizer = TfidfVectorizer(max_features=10000)
         x_train_tfidf = tfidf_vectorizer.fit_transform(x_train)
         x_test_tfidf = tfidf_vectorizer.transform(x_test)

         # Initialize and train the SVM model
         svm_model = SVC(kernel='linear', C=1.0)
         svm_model.fit(x_train_tfidf, y_train)

         # Make predictions on the test set
         y_pred = svm_model.predict(x_test_tfidf)
```

```
In [18]: # Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_rep)
```

```
Accuracy: 0.9811320754716981
Classification Report:
              precision    recall  f1-score   support

          11       0.98      1.00      0.99        46
          92       1.00      1.00      1.00         1
         158       1.00      1.00      1.00         1
         347       1.00      1.00      1.00         2
         359       0.00      0.00      0.00         1
         481       1.00      1.00      1.00         1
         780       1.00      1.00      1.00         1

    accuracy                           0.98        53
   macro avg       0.85      0.86      0.86        53
weighted avg       0.96      0.98      0.97        53
```

Naive Bayes

```
In [19]: # Vectorize your text data using CountVectorizer
count_vectorizer = CountVectorizer(max_features=10000)
x_train_count = count_vectorizer.fit_transform(x_train)
x_test_count = count_vectorizer.transform(x_test)

# Initialize and train the Naive Bayes model
naive_bayes_model = MultinomialNB()
naive_bayes_model.fit(x_train_count, y_train)

# Make predictions on the test set
y_pred = naive_bayes_model.predict(x_test_count)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_rep)
```

```
Accuracy: 0.9622641509433962
Classification Report:
              precision    recall  f1-score   support

          11       0.98      1.00      0.99        46
          92       0.00      0.00      0.00         1
         158       1.00      1.00      1.00         1
         347       1.00      1.00      1.00         2
         359       0.00      0.00      0.00         1
         481       0.50      1.00      0.67         1
         780       1.00      1.00      1.00         1

    accuracy                           0.96        53
   macro avg       0.64      0.71      0.67        53
weighted avg       0.93      0.96      0.95        53
```

RNN

In [20]:
```
# Tokenize the text data
max_words = 10000
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(x_train)

x_train_seq = tokenizer.texts_to_sequences(x_train)
x_test_seq = tokenizer.texts_to_sequences(x_test)

# Pad sequences for uniform input length
max_seq_length = 100  # Adjust as needed
x_train_pad = pad_sequences(x_train_seq, maxlen=max_seq_length, padding='post')
x_test_pad = pad_sequences(x_test_seq, maxlen=max_seq_length, padding='post')

# Convert labels to one-hot encoding
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

num_classes = len(np.unique(y_train_encoded))

# Build the RNN model
model = Sequential()
model.add(Embedding(input_dim=max_words, output_dim=128, input_length=max_seq_length))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='softmax'))

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

# Train the model
batch_size = 32
epochs = 10
model.fit(x_train_pad, y_train_encoded, batch_size=batch_size, epochs=epochs, validation_split=0.2
```

```
Epoch 1/10
16/16 [==============================] - 7s 199ms/step - loss: 5.4580 - accuracy: 0.1894 - val_l
oss: 5.6081 - val_accuracy: 0.0000e+00
Epoch 2/10
16/16 [==============================] - 2s 137ms/step - loss: 4.9791 - accuracy: 0.2220 - val_l
oss: 6.6281 - val_accuracy: 0.0000e+00
Epoch 3/10
16/16 [==============================] - 2s 128ms/step - loss: 4.7247 - accuracy: 0.2220 - val_l
oss: 7.4555 - val_accuracy: 0.0000e+00
Epoch 4/10
16/16 [==============================] - 2s 138ms/step - loss: 4.6669 - accuracy: 0.2220 - val_l
oss: 7.7568 - val_accuracy: 0.0000e+00
Epoch 5/10
16/16 [==============================] - 2s 126ms/step - loss: 4.6384 - accuracy: 0.2220 - val_l
oss: 8.1363 - val_accuracy: 0.0000e+00
Epoch 6/10
16/16 [==============================] - 2s 134ms/step - loss: 4.6227 - accuracy: 0.2220 - val_l
oss: 8.3795 - val_accuracy: 0.0000e+00
Epoch 7/10
16/16 [==============================] - 2s 118ms/step - loss: 4.6107 - accuracy: 0.2220 - val_l
oss: 8.5323 - val_accuracy: 0.0000e+00
Epoch 8/10
16/16 [==============================] - 2s 122ms/step - loss: 4.6074 - accuracy: 0.2220 - val_l
oss: 8.5521 - val_accuracy: 0.0000e+00
Epoch 9/10
16/16 [==============================] - 2s 120ms/step - loss: 4.5979 - accuracy: 0.2220 - val_l
oss: 8.6798 - val_accuracy: 0.0000e+00
Epoch 10/10
16/16 [==============================] - 2s 108ms/step - loss: 4.5842 - accuracy: 0.2220 - val_l
oss: 8.6098 - val_accuracy: 0.0000e+00
```

Out[20]:
```
<keras.callbacks.History at 0x1c8ab78d610>
```

In [21]:
```python
# Evaluate the model
y_pred_probs = model.predict(x_test_pad)
y_pred_encoded = np.argmax(y_pred_probs, axis=1)
y_pred = label_encoder.inverse_transform(y_pred_encoded)

accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_rep)
```

```
2/2 [==============================] - 0s 28ms/step
Accuracy: 0.8679245283018868
Classification Report:
              precision    recall  f1-score   support

          11       0.87      1.00      0.93        46
          92       0.00      0.00      0.00         1
         158       0.00      0.00      0.00         1
         347       0.00      0.00      0.00         2
         359       0.00      0.00      0.00         1
         481       0.00      0.00      0.00         1
         780       0.00      0.00      0.00         1

    accuracy                           0.87        53
   macro avg       0.12      0.14      0.13        53
weighted avg       0.75      0.87      0.81        53
```

Transformer Model

In [22]:
```
!pip install transformers
```

```
Requirement already satisfied: transformers in c:\users\tanmayee\anaconda3\lib\site-packages (4.
31.0)
Requirement already satisfied: packaging>=20.0 in c:\users\tanmayee\anaconda3\lib\site-packages
(from transformers) (21.3)
Requirement already satisfied: requests in c:\users\tanmayee\anaconda3\lib\site-packages (from t
ransformers) (2.28.1)
Requirement already satisfied: regex!=2019.12.17 in c:\users\tanmayee\anaconda3\lib\site-package
s (from transformers) (2022.7.9)
Requirement already satisfied: filelock in c:\users\tanmayee\anaconda3\lib\site-packages (from t
ransformers) (3.6.0)
Requirement already satisfied: tqdm>=4.27 in c:\users\tanmayee\anaconda3\lib\site-packages (from
transformers) (4.64.1)
Requirement already satisfied: pyyaml>=5.1 in c:\users\tanmayee\anaconda3\lib\site-packages (fro
m transformers) (6.0)
Requirement already satisfied: numpy>=1.17 in c:\users\tanmayee\anaconda3\lib\site-packages (fro
m transformers) (1.23.5)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\tanmayee\anaconda3\lib\site-packag
es (from transformers) (0.3.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in c:\users\tanmayee\anaconda3\lib\s
ite-packages (from transformers) (0.16.4)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\tanmayee\anaconda3
\lib\site-packages (from transformers) (0.13.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\tanmayee\anaconda3\lib\sit
e-packages (from huggingface-hub<1.0,>=0.14.1->transformers) (4.7.1)
Requirement already satisfied: fsspec in c:\users\tanmayee\anaconda3\lib\site-packages (from hug
gingface-hub<1.0,>=0.14.1->transformers) (2022.7.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\tanmayee\anaconda3\lib\site-
packages (from packaging>=20.0->transformers) (3.0.9)
Requirement already satisfied: colorama in c:\users\tanmayee\anaconda3\lib\site-packages (from t
qdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\tanmayee\anaconda3\lib\site-packag
es (from requests->transformers) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\tanmayee\anaconda3\lib\site-
packages (from requests->transformers) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\tanmayee\anaconda3\lib\site-pac
kages (from requests->transformers) (1.26.11)
Requirement already satisfied: idna<4,>=2.5 in c:\users\tanmayee\anaconda3\lib\site-packages (fr
om requests->transformers) (3.3)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\tanmayee\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\tanmayee\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\tanmayee\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\tanmayee\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\tanmayee\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\tanmayee\anaconda3\lib\site-packages)
```

In [23]:
```
# Convert labels to numerical values (0, 1, ...)
labels = np.unique(y_train, return_inverse=True)[1]

# Split data for validation
x_train, x_val, y_train, y_val = train_test_split(x_train, labels, test_size=0.2, random_state=42)

# Load BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(np.uniqu
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at
bert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions
and inference.
```

In [24]:
```
# Tokenize the input data
train_encodings = tokenizer(x_train, padding=True, truncation=True, max_length=128, return_tensors
val_encodings = tokenizer(x_val, padding=True, truncation=True, max_length=128, return_tensors='pt
test_encodings = tokenizer(x_test, padding=True, truncation=True, max_length=128, return_tensors='

# Prepare PyTorch DataLoader
train_dataset = torch.utils.data.TensorDataset(train_encodings['input_ids'], train_encodings['atte
val_dataset = torch.utils.data.TensorDataset(val_encodings['input_ids'], val_encodings['attention_
test_dataset = torch.utils.data.TensorDataset(test_encodings['input_ids'], test_encodings['attenti

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=32)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=32)
```

In [25]:
```
# Define training loop
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
optimizer = AdamW(model.parameters(), lr=1e-5)
```

In [26]:
```
num_epochs = 5

# Move the model to the device
model.to(device)

for epoch in range(num_epochs):
    model.train()
    for batch in train_loader:
        input_ids, attention_mask, labels = [item.to(device) for item in batch]

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()

    model.eval()
    val_preds = []
    with torch.no_grad():
        for batch in val_loader:
            input_ids, attention_mask, labels = [item.to(device) for item in batch]

            outputs = model(input_ids, attention_mask=attention_mask)
            logits = outputs.logits
            preds = torch.argmax(logits, dim=1)
            val_preds.extend(preds.cpu().numpy())

    val_accuracy = accuracy_score(y_val, val_preds)
    print(f'Epoch {epoch+1}: Validation Accuracy = {val_accuracy:.4f}')
```

```
Epoch 1: Validation Accuracy = 0.1951
Epoch 2: Validation Accuracy = 0.1951
Epoch 3: Validation Accuracy = 0.1951
Epoch 4: Validation Accuracy = 0.1951
Epoch 5: Validation Accuracy = 0.1951
```

In [27]:
```python
# Make predictions on the test set
test_preds = []
model.eval()
with torch.no_grad():
    for batch in test_loader:
        input_ids, attention_mask = batch
        input_ids, attention_mask = input_ids.to(device), attention_mask.to(device)

        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        preds = torch.argmax(logits, dim=1)
        test_preds.extend(preds.cpu().numpy())

# Convert numerical labels back to original labels
test_preds = np.array(test_preds)
test_labels = np.unique(y_train)[test_preds]
```

In [28]:
```python
# Evaluate the model
test_accuracy = accuracy_score(y_test, test_labels)
classification_rep = classification_report(y_test, test_labels)

print("Test Accuracy:", test_accuracy)
print("Classification Report:")
print(classification_rep)
```

```
Test Accuracy: 0.0
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00       0.0
          11       0.00      0.00      0.00      46.0
          82       0.00      0.00      0.00       0.0
          92       0.00      0.00      0.00       1.0
         158       0.00      0.00      0.00       1.0
         347       0.00      0.00      0.00       2.0
         359       0.00      0.00      0.00       1.0
         481       0.00      0.00      0.00       1.0
         780       0.00      0.00      0.00       1.0

    accuracy                           0.00      53.0
   macro avg       0.00      0.00      0.00      53.0
weighted avg       0.00      0.00      0.00      53.0
```