nlp-project-a

June 23, 2025

[2]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Imdb.csv to Imdb (3).csv
```

```
<IPython.core.display.HTML object>
```

[58]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("Imdb.csv")
df
```

[58]:

	review	sentiment	\
0	One of the other reviewers has mentioned that …	positive	
1	A wonderful little production. The…	positive	
2	I thought this was a wonderful way to spend ti…	positive	
3	Basically there's a family where a little boy …	negative	
4	Petter Mattei's "Love in the Time of Money" is…	positive	
…	…	…	
49995	I thought this movie did a down right good job…	positive	
49996	Bad plot, bad dialogue, bad acting, idiotic di…	negative	
49997	I am a Catholic taught in parochial elementary…	negative	
49998	I'm going to have to disagree with the previou…	negative	
49999	No one expects the Star Trek movies to be high…	negative	

	Unnamed: 2	Unnamed: 3
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
…	…	…
49995	NaN	NaN
49996	NaN	NaN
49997	NaN	NaN

```
49998         NaN         NaN
49999         NaN         NaN

[50000 rows x 4 columns]
```

[59]:
```
#drop unnammed column
df.drop(columns=["Unnamed: 2","Unnamed: 3"],inplace=True)
df
```

[59]:
```
                                                  review sentiment
0      One of the other reviewers has mentioned that …  positive
1      A wonderful little production. <br /><br />The…  positive
2      I thought this was a wonderful way to spend ti…  positive
3      Basically there's a family where a little boy …  negative
4      Petter Mattei's "Love in the Time of Money" is…  positive
…                                                    …         …
49995  I thought this movie did a down right good job…  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di…  negative
49997  I am a Catholic taught in parochial elementary…  negative
49998  I'm going to have to disagree with the previou…  negative
49999  No one expects the Star Trek movies to be high…  negative

[50000 rows x 2 columns]
```

[60]:
```
#head
df.head()
```

[60]:
```
                                              review sentiment
0  One of the other reviewers has mentioned that …  positive
1  A wonderful little production. <br /><br />The…  positive
2  I thought this was a wonderful way to spend ti…  positive
3  Basically there's a family where a little boy …  negative
4  Petter Mattei's "Love in the Time of Money" is…  positive
```

[61]:
```
#information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

[62]:
```
#check the missing value
df.isnull().sum()
```

[62]:
```
review       0
sentiment    0
dtype: int64
```

[63]:
```
#check the class distribution
df["sentiment"].value_counts()
```

[63]:
```
sentiment
positive    25000
negative    25000
Name: count, dtype: int64
```

[64]:
```
#Analyse the review length
df["review_length"]=df["review"].astype(str).apply(len)
df
```

[64]:

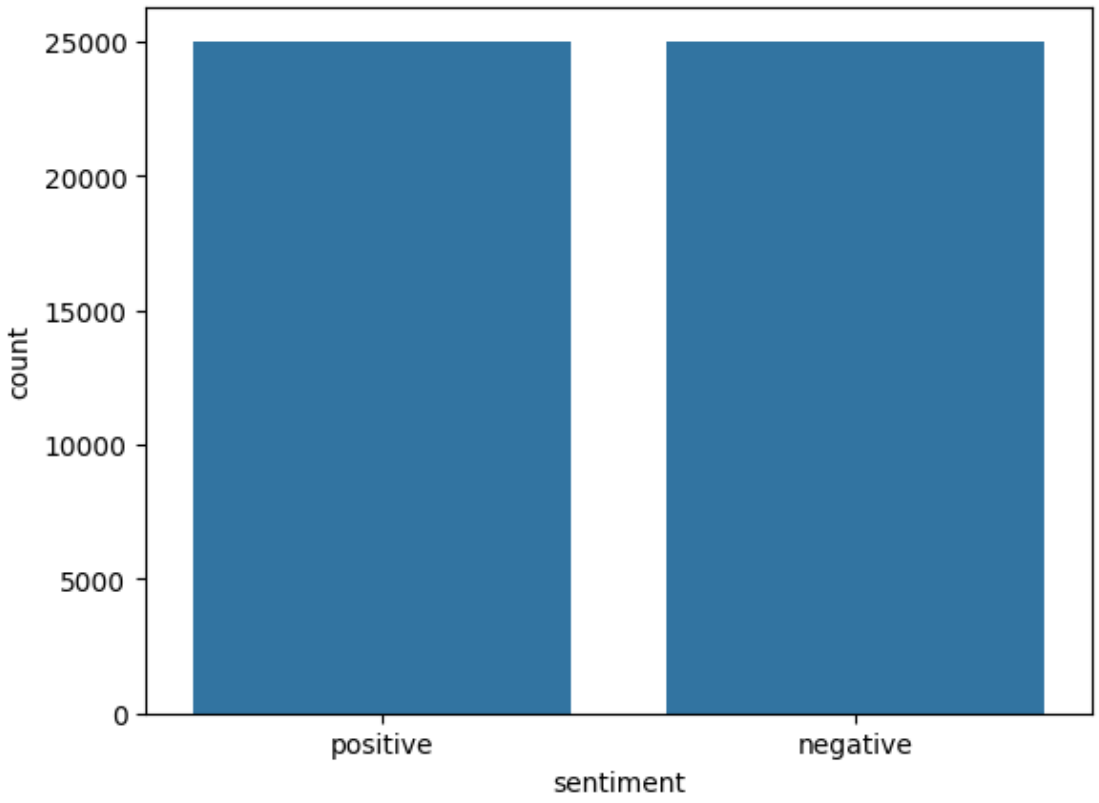
	review	sentiment	review_length
0	One of the other reviewers has mentioned that …	positive	1761
1	A wonderful little production. The…	positive	998
2	I thought this was a wonderful way to spend ti…	positive	926
3	Basically there's a family where a little boy …	negative	748
4	Petter Mattei's "Love in the Time of Money" is…	positive	1317
…	…	…	…
49995	I thought this movie did a down right good job…	positive	1008
49996	Bad plot, bad dialogue, bad acting, idiotic di…	negative	642
49997	I am a Catholic taught in parochial elementary…	negative	1280
49998	I'm going to have to disagree with the previou…	negative	1234
49999	No one expects the Star Trek movies to be high…	negative	678

[50000 rows x 3 columns]

[65]:
```
#Basics statistics of review length
df["review_length"].describe()
```

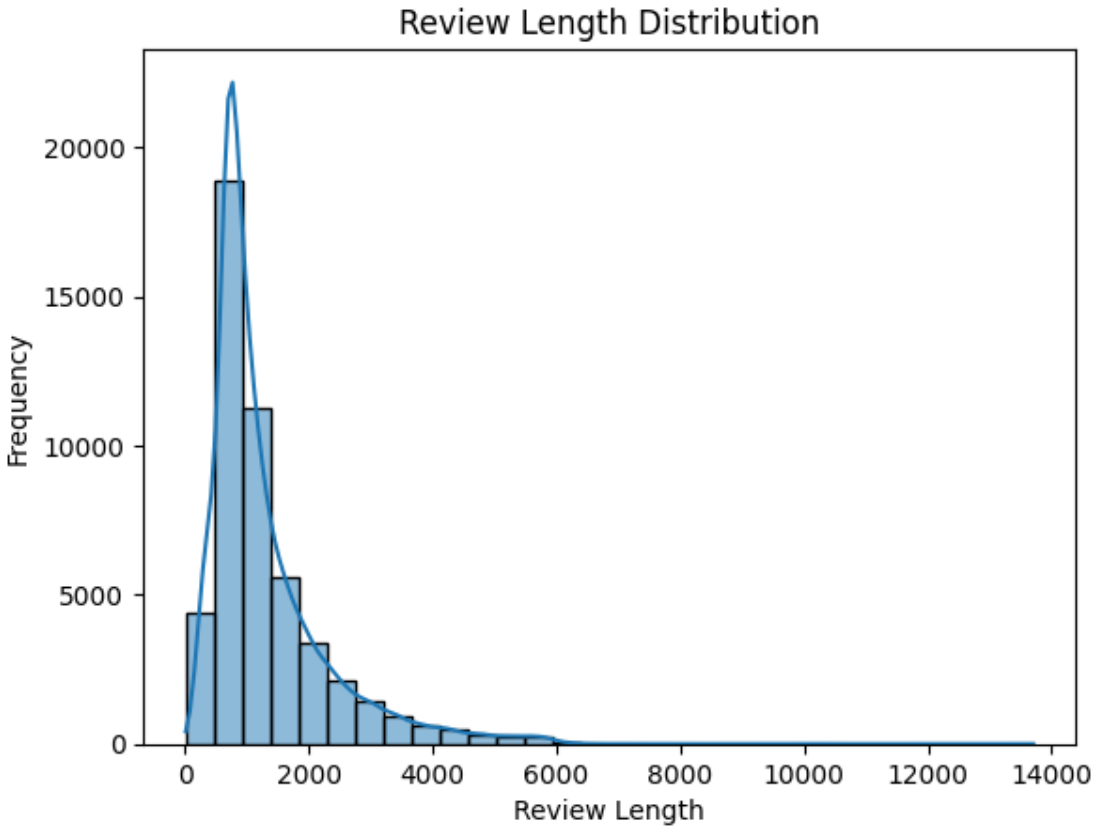
[65]:
```
count    50000.000000
mean      1309.367720
std        989.759532
min          7.000000
25%        699.000000
50%        970.000000
75%       1590.000000
max      13704.000000
Name: review_length, dtype: float64
```

[66]:
```
#Plot sentiment distribution
sns.countplot(x="sentiment",data=df)
plt.show()
```

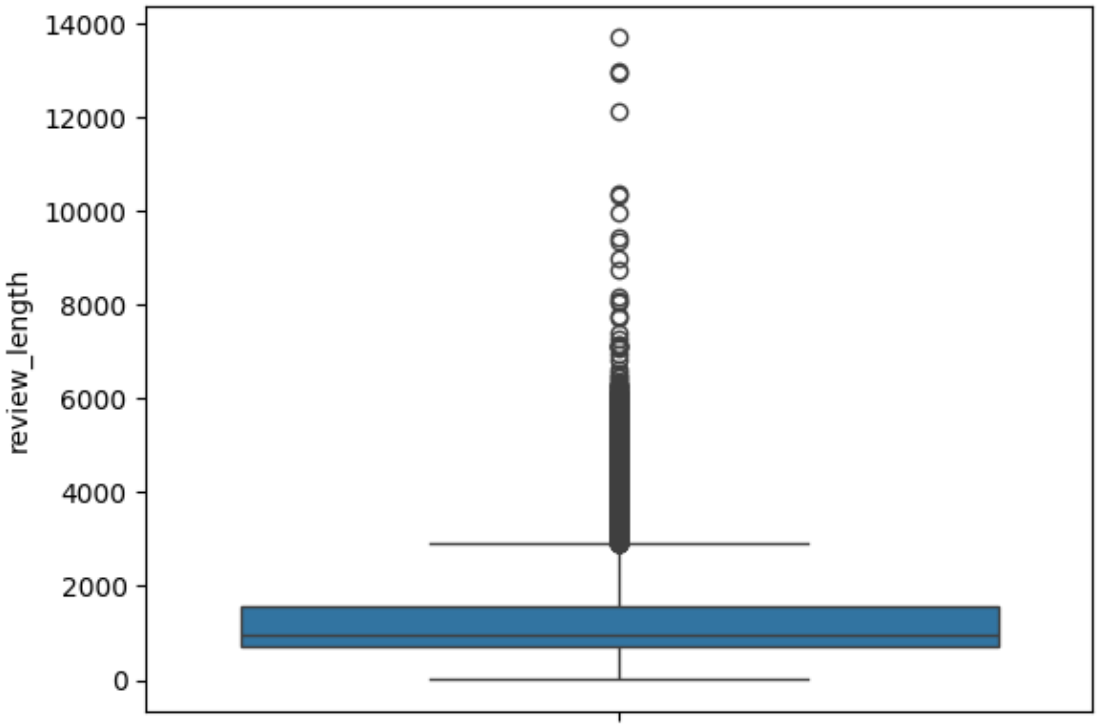


[67]:
```
#plot review lenght distribution histogram
sns.histplot(df["review_length"], bins=30, kde=True)
plt.xlabel("Review Length")
plt.ylabel("Frequency")
```

```
plt.title("Review Length Distribution")
plt.show()
```



[68]:
```
#outliers
sns.boxplot(df["review_length"])
plt.show()
```



[69]:
```
#Insights
#1.No missing values were found.
#2.There is a balanced data set with positive:25000 reviews and negative:25000 reviews
#3.The mean length: 1309 character
#4.The min length: 7 character
#5.Median: 970 character
#6.The distribution of review length is right-skewed with some outliers
```

[70]:
```
#Calculate the 95percentile for review length
percentile=df["review_length"].quantile(0.95)
percentile
```

[70]:
```
3391.0
```

[71]:
```
#filter out the reviews longer then 95percentile
df=df[df["review_length"]<percentile]
df
```

[71]:

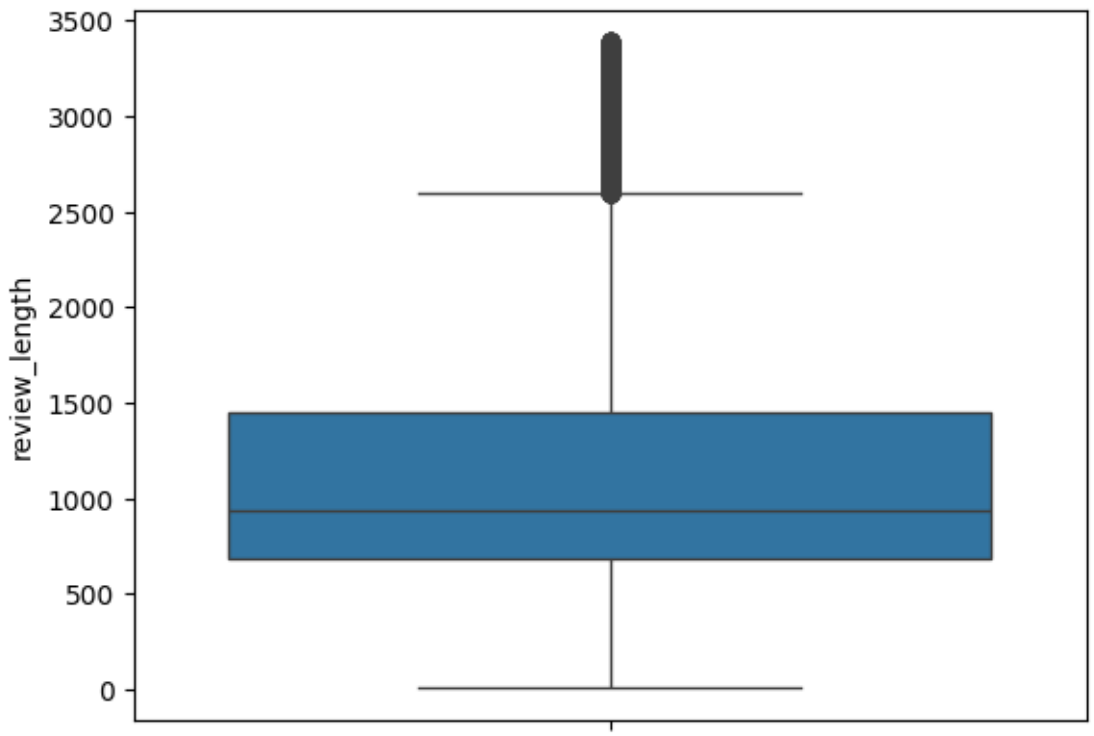
	review	sentiment	\
0	One of the other reviewers has mentioned that …	positive	
1	A wonderful little production. The…	positive	
2	I thought this was a wonderful way to spend ti…	positive	

```
3      Basically there's a family where a little boy …  negative
4      Petter Mattei's "Love in the Time of Money" is…  positive
…                                                    …         …
49995  I thought this movie did a down right good job…  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di…  negative
49997  I am a Catholic taught in parochial elementary…  negative
49998  I'm going to have to disagree with the previou…  negative
49999  No one expects the Star Trek movies to be high…  negative

       review_length
0               1761
1                998
2                926
3                748
4               1317
…                …
49995           1008
49996            642
49997           1280
49998           1234
49999            678

[47499 rows x 3 columns]
```

[73]:
```
#check Nan values for sentiment
print(df["sentiment"].isna().sum())
```

```
0
```

[74]:
```
#outliers
sns.boxplot(df["review_length"])
plt.show()
```



[75]:
```
#New statistics after removing outliers
df["review_length"].describe()
```

[75]:
```
count    47499.000000
mean      1145.277332
std        670.632472
min          7.000000
25%        689.000000
50%        934.000000
75%       1451.000000
max       3390.000000
Name: review_length, dtype: float64
```

[76]:
```
#Value count
df["sentiment"].value_counts()
```

[76]:
```
sentiment
negative    23870
positive    23629
Name: count, dtype: int64
```

[77]:
```
#Insights
#1.reviews kept 47499 out of 50000
```

```
#2.mean is 1145 character
#3.maximum is 3390 character
#4.median is 934 character
#5.The sentiment is still fairly balanced with slight drop in each class
```

[78]:
```
import re
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
```

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

[78]:
```
True
```

[79]:
```
#function to clean a single review
def clean_review(text):
  #remove html tags
  text=BeautifulSoup(text,"html.parser").get_text()
  #lowercase
  text=text.lower()
  #stopwords
  stop_words=set(stopwords.words("english"))
  text=" ".join([word for word in text.split() if word not in stop_words])
  #remove non-alphabetic character(punctuation,number)
  text=re.sub(r'[^a-z\s]','',text)
  #remove extra white space
  text=re.sub(r'\s+',' ',text).strip()
  return text

#Apply cleaning to the reviews
df["cleaned_review"]=df["review"].apply(clean_review)
```

```
<ipython-input-79-348261445>:17: 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
  df["cleaned_review"]=df["review"].apply(clean_review)
```

[80]:
```
#show few clean review
df['cleaned_review'].head(20)
```

[80]:
```
0     one reviewers mentioned watching oz episode ho…
1     wonderful little production filming technique …
2     thought wonderful way spend time hot summer we…
3     basically theres family little boy jake thinks…
4     petter matteis love time money visually stunni…
5     probably alltime favorite movie story selfless…
6     sure would like see resurrection dated seahunt…
7     show amazing fresh innovative idea s first air…
8     encouraged positive comments film looking forw…
9     like original gut wrenching laughter like movi…
10    phil alien one quirky films humour based aroun…
11    saw movie came out recall scariest scene big b…
12    im big fan bolls work many are enjoyed movie p…
13    cast played shakespeareshakespeare losti appre…
14    fantastic movie three prisoners become famous …
15    kind drawn erotic scenes realize one amateuris…
16    films simply remade one them bad film fails ca…
17    movie made one top awful movies horrible conti…
18    remember filmit first film watched cinema pict…
19    awful film must real stinkers nominated golden…
Name: cleaned_review, dtype: object
```

[81]:
```
#lemmatization
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import pos_tag
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger_eng')
lemmatizer = WordNetLemmatizer()
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
```

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

[82]:
```
#Function to convert pos tag
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

#Lemmatization function
def lemmatize_text(text):
   words=text.split()
   tagged_words=pos_tag(words)
   lemmatized_words=[lemmatizer.lemmatize(word,get_wordnet_pos(tag)) for␣
 ↪word,tag in tagged_words]
   return " ".join(lemmatized_words)

#apply to data frame
df["lemmatized_review"]=df["cleaned_review"].apply(lemmatize_text)
```

```
<ipython-input-82-2809464084>:22: 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
  df["lemmatized_review"]=df["cleaned_review"].apply(lemmatize_text)
```

[83]:
```
#textural features
#word count
df["word_count"]=df["lemmatized_review"].apply(lambda x:len(x.split()))
df["word_count"]
```

```
<ipython-input-83-4255314011>:3: 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
```

```
  df["word_count"]=df["lemmatized_review"].apply(lambda x:len(x.split()))
```

[83]:
```
0        167
1         87
2         83
3         66
4        125
        ...
49995     86
49996     57
49997    114
49998    114
49999     68
Name: word_count, Length: 47499, dtype: int64
```

[84]:
```
#Character count
df["character_count"]=df["lemmatized_review"].apply(lambda x:len(x))
df["character_count"]
```

```
<ipython-input-84-2959121045>:2: 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
  df["character_count"]=df["lemmatized_review"].apply(lambda x:len(x))
```

[84]:
```
0        1089
1         631
2         543
3         427
4         838
        ...
49995     518
49996     381
49997     781
49998     808
49999     408
Name: character_count, Length: 47499, dtype: int64
```

[85]:
```
#Average word length
df["average_word_length"]=df["character_count"]/df["word_count"]
df["average_word_length"]
```

```
<ipython-input-85-3416321984>:2: 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
  df["average_word_length"]=df["character_count"]/df["word_count"]
```

[85]:
```
0        6.520958
1        7.252874
2        6.542169
3        6.469697
4        6.704000
           …
49995    6.023256
49996    6.684211
49997    6.850877
49998    7.087719
49999    6.000000
Name: average_word_length, Length: 47499, dtype: float64
```

[86]:
```
#TF-IDF Vectorization
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer=TfidfVectorizer()
X_tfidf=tfidf_vectorizer.fit_transform(df["lemmatized_review"])
```

[87]:
```
#Binary encode labels
df["sentiment"] = df["sentiment"].map({"positive":1,"negative":0})
```

```
<ipython-input-87-258216660>:2: 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
  df["sentiment"] = df["sentiment"].map({"positive":1,"negative":0})
```

[88]:
```
#Check Nan values
print(df["sentiment"].isna().sum())
```

```
0
```

[89]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_tfidf,df["sentiment"],test_size=0.
↪2,random_state=42)
```

[90]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report
lr=LogisticRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
```

```
print("Classification Report:\n",classification_report(y_test,y_pred))
```

```
Accuracy: 0.886
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      4759
           1       0.88      0.89      0.89      4741

    accuracy                           0.89      9500
   macro avg       0.89      0.89      0.89      9500
weighted avg       0.89      0.89      0.89      9500
```

[91]:
```
from sklearn.naive_bayes import MultinomialNB
nb=MultinomialNB()
nb.fit(X_train,y_train)
y_pred=nb.predict(X_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Classification Report:\n",classification_report(y_test,y_pred))
```

```
Accuracy: 0.8557894736842105
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      4759
           1       0.88      0.82      0.85      4741

    accuracy                           0.86      9500
   macro avg       0.86      0.86      0.86      9500
weighted avg       0.86      0.86      0.86      9500
```

[92]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
y_pred=rf.predict(X_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Classification Report:\n",classification_report(y_test,y_pred))
```

```
Accuracy: 0.8494736842105263
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.86      0.85      4759
           1       0.86      0.84      0.85      4741

    accuracy                           0.85      9500
```

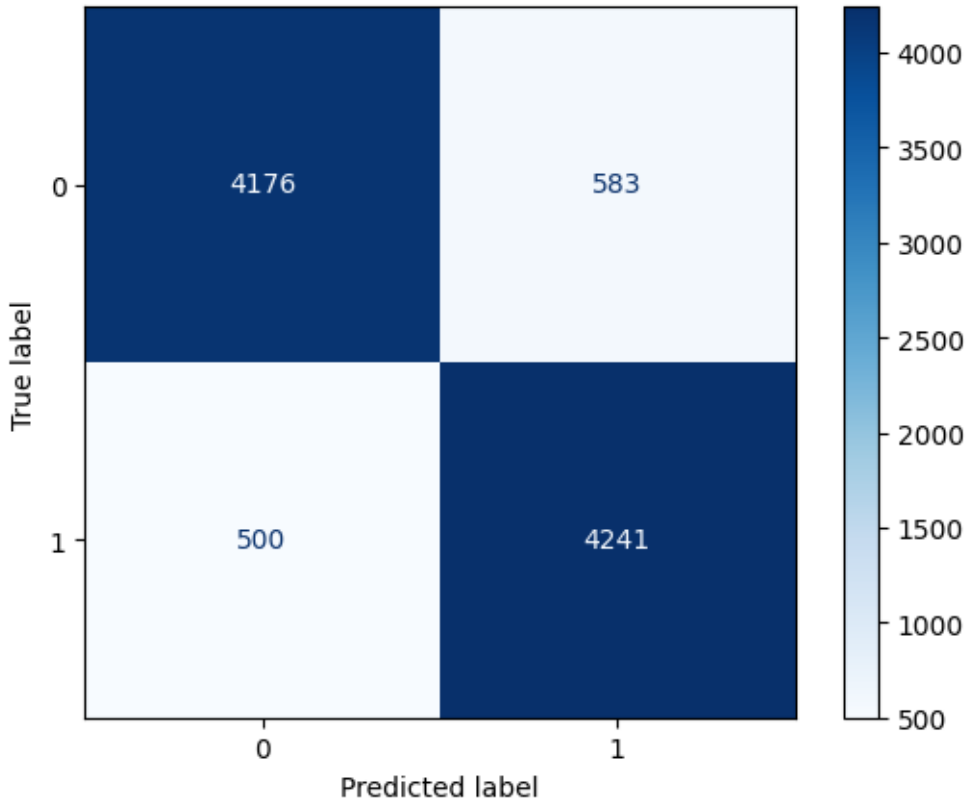
```
   macro avg       0.85      0.85      0.85      9500
weighted avg       0.85      0.85      0.85      9500
```

[93]:
```
#Insights
#1.accuracy of logistic regression is 88 percent
#2.accuracy of Random forest classifier and multinomial is 84 percent
#3. So we will go with logistic regression for the further process
```

[94]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,roc_auc_score
lr=LogisticRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Classification Report:\n",classification_report(y_test,y_pred))
```

```
Accuracy: 0.886
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      4759
           1       0.88      0.89      0.89      4741

    accuracy                           0.89      9500
   macro avg       0.89      0.89      0.89      9500
weighted avg       0.89      0.89      0.89      9500
```

[95]:
```
#Visualizations
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
import matplotlib.pyplot as plt
cm=confusion_matrix(y_test,y_pred)
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=lr.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.show()
```



[96]:
```
#Insights
#1.True positive is 4241
#2.True negative is 4176
#3.False positive is 583
#4. False negative is 500
#5. Indicates that model is balanced no major bias is shown
#6. Most reviews are correctly predicted
```

[114]:
```
#predict sentiment for new input
def predict_sentiment(new_reviews):
  #preprocess the new reviews
  cleaned_reviews=[clean_review(review) for review in new_reviews]
  lemmatized_reviews=[lemmatize_text(review) for review in cleaned_reviews]
  new_reviews_tfidf=tfidf_vectorizer.transform(lemmatized_reviews)
  predictions=lr.predict(new_reviews_tfidf)
  return ["positive" if pred==1 else "negative" for pred in predictions]
```

[115]:
```
#example
print(predict_sentiment(["This movie is great!"]))
```

```
['positive']
```

[116]:
```
#example2
print(predict_sentiment(["This movie is bad!"]))
```

```
['negative']
```