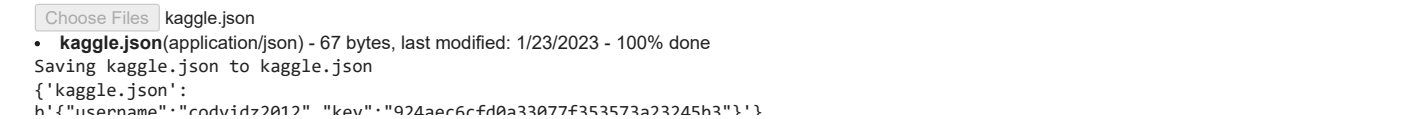
To perform the following lab, I took Kaggle's API and downloaded the dataset IMDB Movie Review to google colab, and extracted it. Below are the following commands I performed to get the dataset and extract it to a new folder.

```
! pip install -q kaggle
```

```
from google.colab import files

files.upload()
```




```
! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/
```

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
! kaggle datasets download lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
```

```
Downloading imdb-dataset-of-50k-movie-reviews.zip to /content
 31% 8.00M/25.7M [00:00<00:00, 83.1MB/s]
100% 25.7M/25.7M [00:00<00:00, 157MB/s]
```

```
! mkdir IMDB-Movie_Review
```

```
! unzip /content/imdb-dataset-of-50k-movie-reviews.zip -d /content/IMDB-Movie_Review
```

```
Archive:  /content/imdb-dataset-of-50k-movie-reviews.zip
  inflating: /content/IMDB-Movie_Review/IMDB Dataset.csv
```

Here I import the neccessary libraries and dataset I will use to perform the sentiment analysis on.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import spacy

plt.rcParams['figure.figsize'] = (8, 8)
```

```
movie_reviews = pd.read_csv('/content/IMDB-Movie_Review/IMDB Dataset.csv')
movie_reviews.head()
```

	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

```
movie_reviews.shape
```

```
(50000, 2)
```

Here I begin to perform Text-Preprocessing

- Remove duplicates
- Lowercase the reviews
- Remove HTML tags

- Remove Punctuation
  *
  *
  *

```
movie_reviews.duplicated().sum() #Check for duplicates
```

```
418
```

```
movie_reviews.drop_duplicates(inplace=True) #Remove duplicates
```

```
movie_reviews.duplicated().sum() #Check again to make sure duplicates were removed
```

```
0
```

```
# Performing data cleaning, remvoing html tags, and more.
import re

def removeHTML(text):
    pattern = re.compile('<.*?>')
    cleaned_text = re.sub(pattern,' ',text)
    return cleaned_text
```

```
movie_reviews['review'] = movie_reviews['review'].str.lower() #Lower case all reviews.
```

```
movie_reviews.head()
```

	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

```
movie_reviews['review'] = movie_reviews['review'].apply(removeHTML) #Apply the removeHTML function to the review column
```

```
movie_reviews.head()
```

	review	sentiment
0	one of the other reviewers has mentioned that ...	positive
1	a wonderful little production. the filming t...	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

```
movie_reviews['review'] = movie_reviews['review'].str.replace('[^\w\s]', '  ', regex=True) #Remove punctuations
```

```
movie_reviews.head()
```

	review	sentiment
0	one of the other reviewers has mentioned that ...	positive
1	a wonderful little production the filming ...	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 ...	positive

BoW Naive Bayes Classifier for sentiment analysis

Here we will generate the bag-of-words for the training and testing of the moview review data. Then we will use it to train Naive Bayes classifier and print its accuracy.

```
X = movie_reviews['review']
y = movie_reviews['sentiment']
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

```
# Create a CounterVectorizer object
vectorizer = CountVectorizer(lowercase=True, stop_words='english')

# fit and transform X_train
X_train_bow = vectorizer.fit_transform(X_train)

# Transform X_test
X_test_bow = vectorizer.transform(X_test)

# Print shape of X_train_bow and X_test_bow
print(X_train_bow.shape)
print(X_test_bow.shape)
```

```
(37186, 90201)
(12396, 90201)
```

```
from sklearn.metrics import classification_report
```

```
#It is incapable of detecting neutral reviews, only positive and negative.
from sklearn.naive_bayes import MultinomialNB

# Create a MultinomialNB object
clf = MultinomialNB()

# Fit the classifier
clf.fit(X_train_bow, y_train)

# Measure the accuracy
accuracy = clf.score(X_test_bow, y_test)
print("The accuracy of the classifier on the test set is %.2f \n" % accuracy)
nb_bow_preds = clf.predict(X_test_bow)

''' Uncomment this for predicting the sentiment of a negative/positive review

#review = 'The movie was terrible. The music was underwhelming and the acting mediocre.'
#prediction = clf.predict(vectorizer.transform([review]))[0]
#print("The sentiment predicted by the classifier is", prediction)
'''

print("Classification report: \n", classification_report(y_test, nb_bow_preds, target_names=["negative","positive"]))
```

```
The accuracy of the classifier on the test set is 0.86

Classification report:
               precision    recall  f1-score   support

     negative       0.84      0.88      0.86      6195
     positive       0.87      0.83      0.85      6201

     accuracy                           0.86     12396
    macro avg       0.86      0.86      0.86     12396
 weighted avg       0.86      0.86      0.86     12396
```

Higher order n-grams for sentiment analysis

Here we will use n-grams up to n=2 for testing using MultinomialNB and print its accuracy.

```

```

```
ng_vectorizer = CountVectorizer(ngram_range=(1, 2))
X_train_ng = ng_vectorizer.fit_transform(X_train)
X_test_ng = ng_vectorizer.transform(X_test)
```

```
# Define an instance of MultinomialNB
clf_ng = MultinomialNB()

# Fit the classifier
clf_ng.fit(X_train_ng, y_train)

# Measure the accuracy
accuracy = clf_ng.score(X_test_ng, y_test)
print("The accuracy of the classifier on the test set is %.2f \n" % accuracy)
ng_preds = clf_ng.predict(X_test_ng)

''' Uncomment this for predicting the sentiment of a negative review OR enter positive text for positive review

#review = 'The movie was not good. The plot had several holes and the acting lacked panache'
#prediction = clf_ng.predict(ng_vectorizer.transform([review]))[0]
#print("The sentiment predicted by the classifier is", prediction)

'''

print("Classification report: \n", classification_report(y_test, ng_preds, target_names=["negative","positive"]))
```

```
The accuracy of the classifier on the test set is 0.88

Classification report:
               precision    recall  f1-score   support

     negative       0.87      0.89      0.88      6256
     positive       0.89      0.86      0.88      6140

     accuracy                           0.88     12396
    macro avg       0.88      0.88      0.88     12396
weighted avg       0.88      0.88      0.88     12396
```

Overall, the n-gram model performs slightly better than BoW NB model.

References Used:

https://colab.research.google.com/github/goodboychan/chans_jupyter/blob/main/_notebooks/2020-07-17-03-N-Gram-models.ipynb#scrollTo=ibIgyPo7pbqh

https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/code?datasetId=134715

https://stackoverflow.com/questions/9662346/python-code-to-remove-html-tags-from-a-string