# Week 10 Learning Activities

We will conduct this lab using the **IMDb movie reviews dataset** for a text classification task. This dataset includes positive and negative sentiment labels, making it suitable for a sentiment classification model. **TF-IDF vectorizer** for feature extraction and a Random Forest Classification model for classification will be used.

## Dataset Load and Visualization

```
from pandas import read_csv
df = read_csv('/home/hailq/.cache/kagglehub/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/versions/1/
df
```

✓ 0.8s

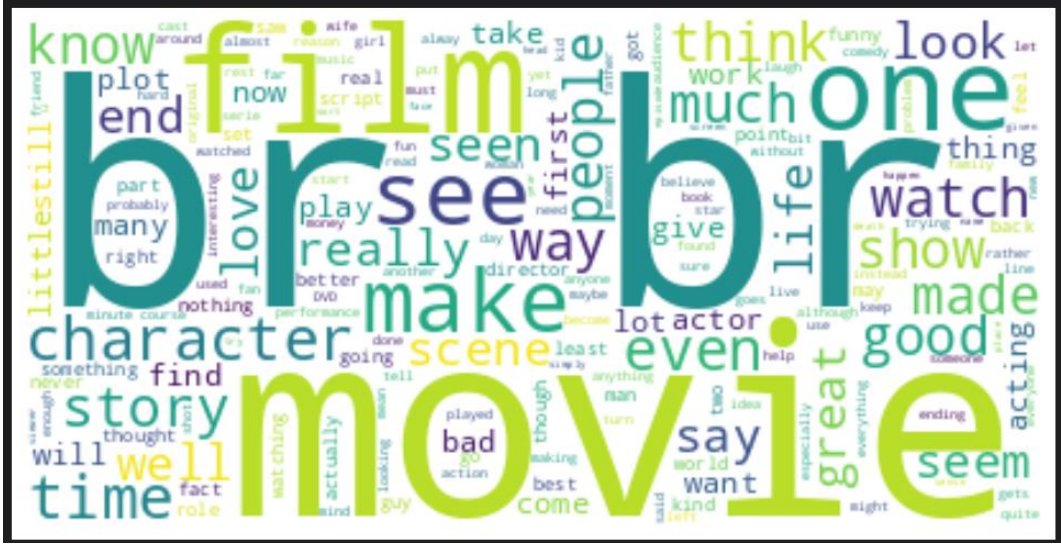
| | review | sentiment |
|---|---|---|
| 0 | One of the other reviewers has mentioned that ... | positive |
| 1 | A wonderful little production. <br /><br />The... | positive |
| 2 | I thought this was a wonderful way to spend ti... | positive |
| 3 | Basically there's a family where a little boy ... | 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 |

## Initial Word Cloud

```
# Generate a word cloud image
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(text)

# Display our Word Cloud
from matplotlib import pyplot as plt
plt.figure(figsize=(12,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

✓ 33.8s



## Word Cloud with Stop Word Eliminated

```
from wordcloud import STOPWORDS
stopwords = set(STOPWORDS)
stopwords.update(["movie", "film", "character", "way", "story", "plot", "really", "know", "look"
                  "scene", "actor", "people", "br", "see", "make", "one", "time", "say", "watch"])

# Step 3: Generate a new word cloud with updated stopwords
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)

# Display the refined word cloud
plt.figure(figsize=(12,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Refined Word Cloud with Custom Stopwords")
plt.show()
```

✓ 29.2s

## Data Preprocess

```python
from re import sub, I
# Split the text and the labels
features = df['review'].values
labels = df['sentiment'].values
processed_features = []

for sentence in range(0, len(features)):
    # Remove all the special characters
    processed_feature = sub(r'\W', ' ', str(features[sentence]))

    # remove all single characters
    processed_feature= sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

    # Remove single characters from the start
    processed_feature = sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)

    # Substituting multiple spaces with single space
    processed_feature = sub(r'\s+', ' ', processed_feature, flags=I)

    # Removing prefixed 'b'
    processed_feature = sub(r'^b\s+', '', processed_feature)

    # Converting to Lowercase
    processed_feature = processed_feature.lower()

    processed_features.append(processed_feature)
```

(variable) processed_feature: str

✓ 11.9s

✓ features ···

```
array(["One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are rig
       'A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-BBC fashio
       'I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned
       ...,
       'I am a Catholic taught in parochial elementary schools by nuns, taught by Jesuit priests in high school & col
       'I\'m going to have to disagree with the previous comment and side with Maltin on this one. This is a second r
       "No one expects the Star Trek movies to be high art, but the fans do expect a movie that is as good as some of
      dtype=object)
```

## Model Build and Training

```
# Create NLP model
from nltk import download
download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer (max_features=2500, min_df=7, max_df=0.8, stop_words=stopwords.words('english'
processed_features = vectorizer.fit_transform(processed_features).toarray()
```

✓ 10.3s

```
[nltk_data] Downloading package stopwords to /home/hailq/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state

from sklearn.ensemble import RandomForestClassifier
text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
```

✓ 4m 5.2s

```
RandomForestClassifier
RandomForestClassifier(n_estimators=200, random_state=0)
```

## Model Evaluation

```
# Let's obtain our predictions on our test dataset
predictions = text_classifier.predict(X_test)
# Let's display the results of our classifier on our test dataset
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print("Accuracy {0:.2f}%".format(100*accuracy_score(y_test, predictions)))
```

✓ 3.4s

```
[[4313  722]
 [ 786 4179]]
              precision    recall  f1-score   support

    negative       0.85      0.86      0.85      5035
    positive       0.85      0.84      0.85      4965

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

Accuracy 84.92%
```

# Classification on a few Samples

```
from numpy.random import randint
for i in range(5):
  sample = randint(0, len(df))
  sample_text = df.iloc[sample]['review']
  sample_label = df.iloc[sample]['sentiment']
  print(sample_text)
  print(sample_label)

  # Process the sample text
  processed_sample = sub(r'\W', ' ', sample_text)
  processed_sample = sub(r'\s+[a-zA-Z]\s+', ' ', processed_sample)
  processed_sample = sub(r'\^[a-zA-Z]\s+', ' ', processed_sample)
  processed_sample = sub(r'\s+', ' ', processed_sample, flags=I)
  processed_sample = sub(r'^b\s+', '', processed_sample)
  processed_sample = processed_sample.lower()

  # Vectorize the processed sample
  vectorized_sample = vectorizer.transform([processed_sample]).toarray()

  # Obtain the prediction of the sample
  prediction = text_classifier.predict(vectorized_sample)
  print(prediction)
```

✓ 0.1s

```
The film Classe tous risques directed by Claude Sautet was not a film, to be honest, I had ever really heard of until the Film Forum in
positive
['positive']
Of course, really experienced reviewers who like stuff like Star Wars and professional crap will definitely drag this movie and say real
positive
['positive']
It was obvious that this movie is designed to appeal to the Chick Flick audience, to which i have sat through quite a few and enjoyed mo
negative
['negative']
I saw this film at Temple University. I cannot imaging that anyone will ever see this film in a theater (projected on film). The acting
negative
['negative']
Some amusing humor, some that falls flat, some decent acting, some that is quite atrocious. This movie is simply hit and miss, guarantee
negative
['negative']
```