

```
In [2]: from sklearn.naive_bayes import MultinomialNB
import pandas as pd
import numpy as np
import seaborn as sns
import re
import string
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
```

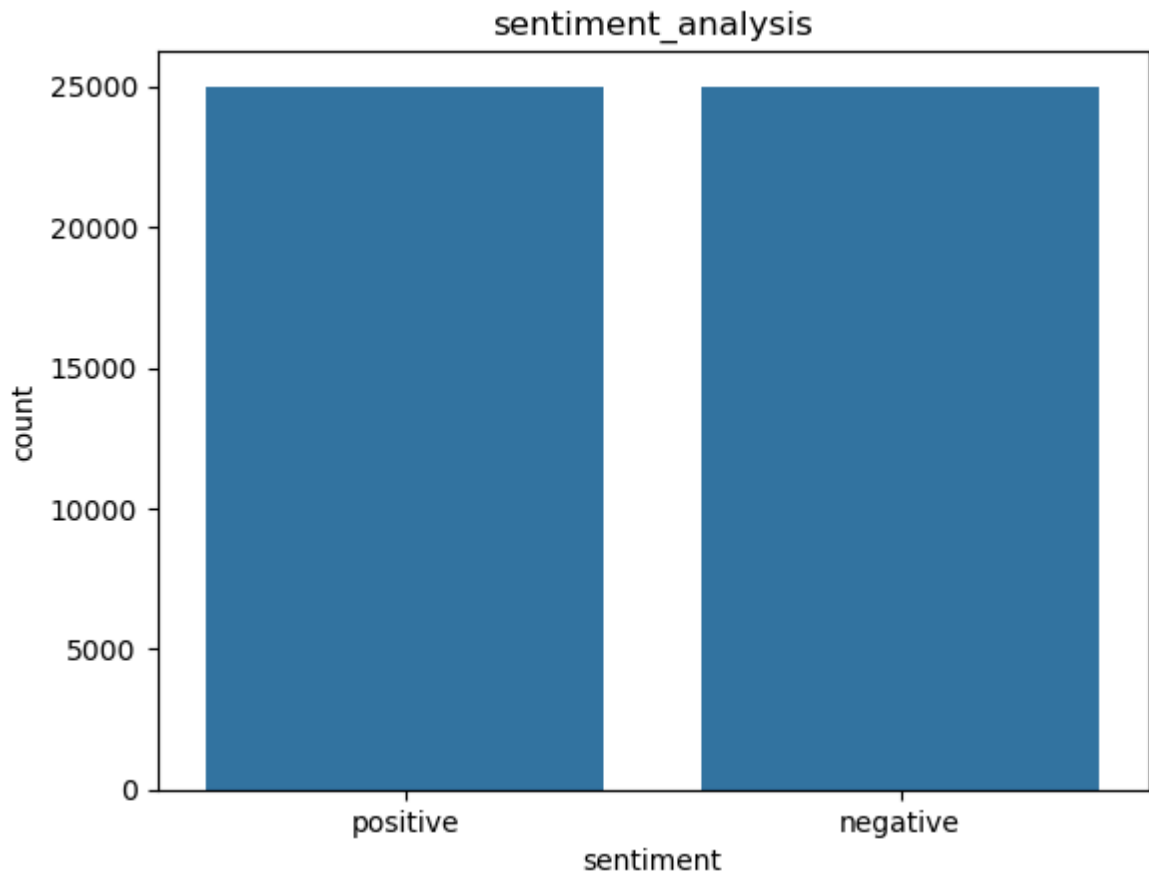
```
In [3]: dm=pd.read_csv(r"C:\Users\SENAPATHI REDDY.K\Downloads\_MConverter.eu_imdb_review
```

```
In [4]: dm.head()
```

```
Out[4]:
```

	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

```
In [5]: sns.countplot(x="sentiment",data=dm)
plt.title("sentiment_analysis")
plt.show()
```



```
In [7]: dm.isna().sum()
```

```
Out[7]: review      0  
sentiment      0  
dtype: int64
```

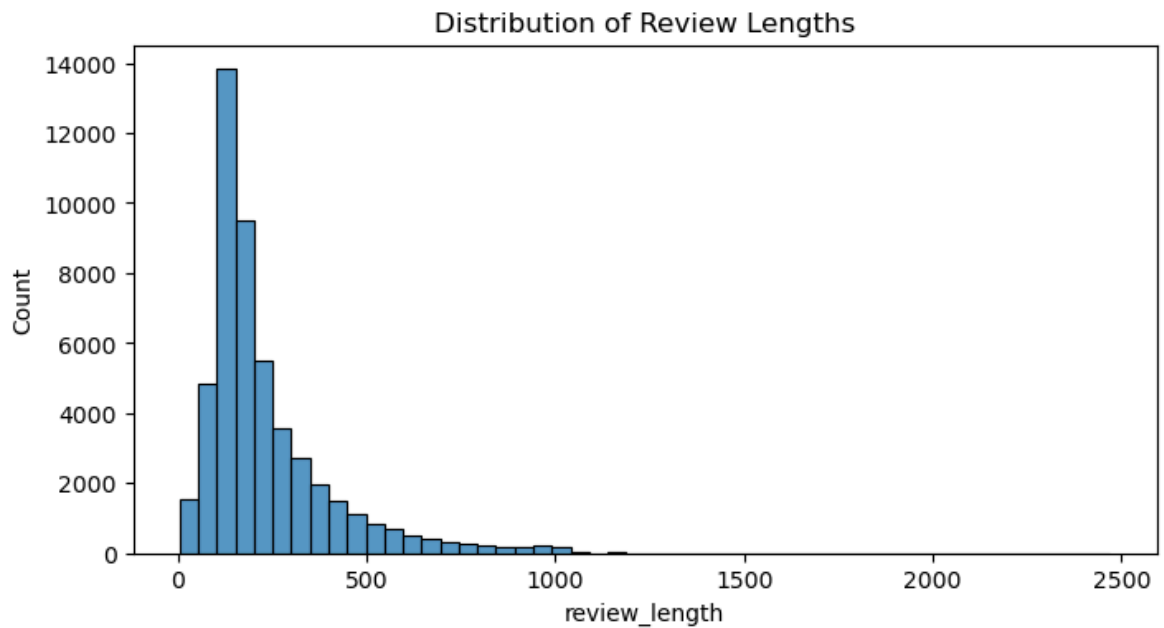
```
In [8]: dm["sentiment"].value_counts
```

```
Out[8]: <bound method IndexOpsMixin.value_counts of 0      positive  
1      positive  
2      positive  
3      negative  
4      positive  
...  
49995   positive  
49996   negative  
49997   negative  
49998   negative  
49999   negative  
Name: sentiment, Length: 50000, dtype: object>
```

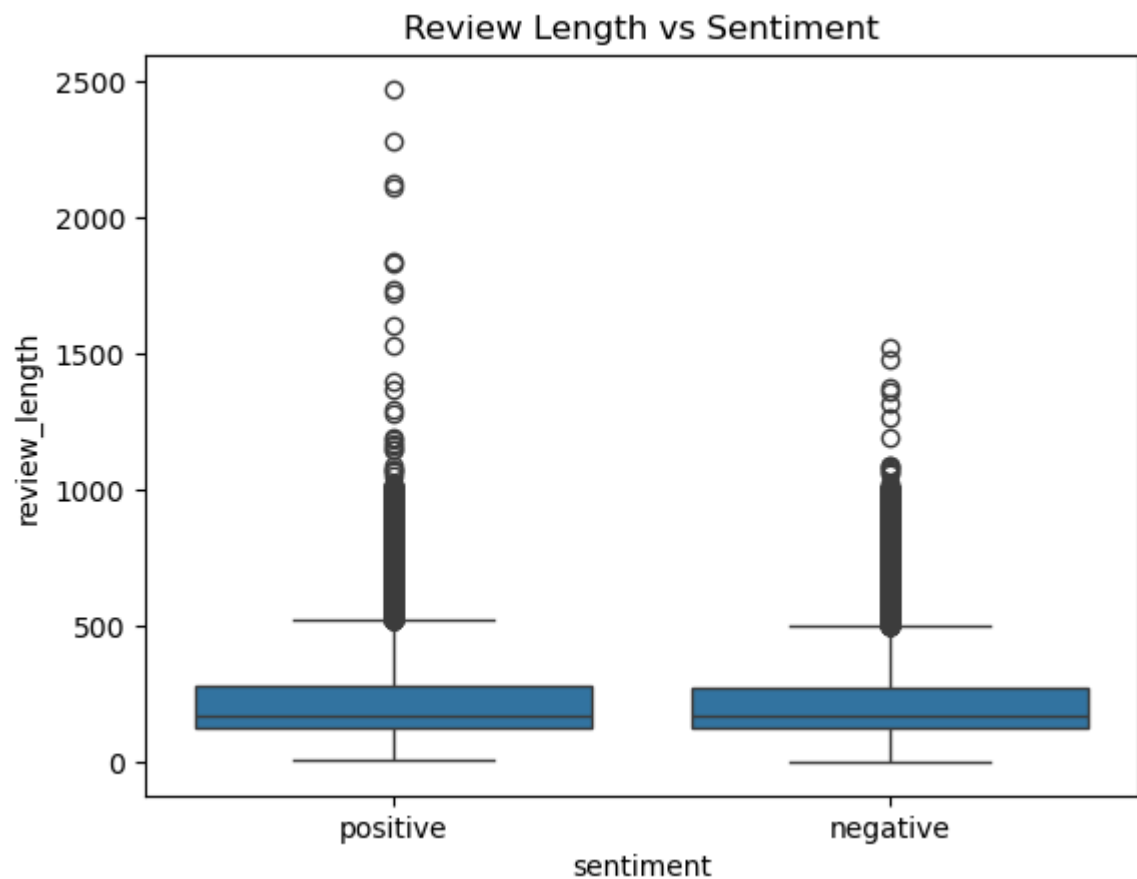
```
In [9]: dm_encoded=pd.get_dummies(  
    dm,  
    columns=["sentiment"],  
    drop_first=True,  
    dtype=int  
)
```

```
In [10]: dm["review_length"]=dm["review"].apply((lambda x: len(x.split())))
```

```
In [12]: plt.figure(figsize=(8,4))  
sns.histplot(dm['review_length'], bins=50)  
plt.title("Distribution of Review Lengths")  
plt.show()
```



```
In [13]: sns.boxplot(x='sentiment', y='review_length', data=dm)  
plt.title("Review Length vs Sentiment")  
plt.show()
```



```

In [14]: def clean_text(text):
          text=text.lower()
          text = re.sub(r"<.*?>", "", text)
          text = re.sub(r"^\w\s]", "", text)
          text = re.sub(r"\d+", "", text)
          return text

In [16]: dm['clean_review'] = dm['review'].apply(clean_text)

In [19]: x_train,x_test,y_train,y_test=train_test_split(dm["review"],dm["sentiment"],test

In [20]: vectorizer=TfidfVectorizer(
          max_features=5000,
          stop_words="english"
        )

In [21]: x_train_vec = vectorizer.fit_transform(x_train)
          x_test_vec = vectorizer.transform(x_test)

In [23]: nb_model = MultinomialNB()
          nb_model.fit(x_train_vec, y_train)

Out[23]: ▼ MultinomialNB ⓘ ?
          MultinomialNB()

In [25]: y_pred_nb = nb_model.predict(x_test_vec)

          print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
          print("\nClassification Report:\n", classification_report(y_test, y_pred_nb))
          print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_nb))

```

Naive Bayes Accuracy: 0.8528

Classification Report:

	precision	recall	f1-score	support
negative	0.85	0.85	0.85	4912
positive	0.85	0.86	0.86	5088
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Confusion Matrix:

```

[[4166 746]
 [ 726 4362]]

```

In []: