Data Preparation

In [ ]:
```
# Importing nltk libarary
import nltk
import random
from collections import Counter

# Downloading the movie-review datase
nltk.download('movie_reviews')

# Basic Imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Importing the required functions for pre-processing the dataset

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer

# Download the requirements
nltk.download('stopwords')
nltk.download("punkt")
nltk.download('wordnet')
stopwords = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     /Users/paarthvisharma/nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/paarthvisharma/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/paarthvisharma/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/paarthvisharma/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [ ]:
```
from nltk.corpus import movie_reviews

# Access the movie reviews and labels
documents = [(list(movie_reviews.words(fileid)), category)
for category in movie_reviews.categories()
for fileid in movie_reviews.fileids(category)]
# Shuffle the documents to ensure a balanced distribution of positive and nega
random.shuffle(documents)
```

In [ ]:
```
# Review the results
print(documents[0][0])
print(documents[0][1])
```

```
['--', 'comedy', ',', 'rated', 'pg', ',', 'runs', 'about', '1', ':', '40', '--
', 'starring', ':', 'john', 'goodman', ',', 'kathy', 'moriarty', ',', 'and',
'a', 'bunch', 'of', 'teenagers', '--', 'directed', 'by', 'joe', 'dante', 'an
d', 'written', 'by', 'charles', 'hass', '--', 'summary', ':', 'lawrence', 'woo
lsley', '(', 'john', 'goodman', ')', 'brings', 'his', 'new', 'horror', 'film',
'mant', '!', 'to', 'premiere', 'in', 'key', 'west', 'during', 'the', 'height',
'of', 'the', 'cuban', 'missile', 'crisis', '.', 'he', 'hopes', 'to', 'capitali
ze', 'on', 'the', 'tense', 'moment', 'by', 'providing', 'an', 'escape', 'for',
'the', 'town', '.', 'we', 'see', 'most', 'of', 'the', 'events', 'through', 'th
e', 'stories', 'of', 'four', 'teenagers', 'and', 'how', 'life', 'affects', 'th
em', '.', 'quick', 'and', 'easy', 'review', ':', 'i', 'really', 'enjoyed', 'ma
tinee', '.', 'the', 'mixture', 'of', 'comedy', 'and', 'tension', 'blended', 'n
icely', '.', 'unlike', 'many', 'comedies', 'this', 'film', 'tries', ',', 'an
d', 'succeeds', ',', 'in', 'getting', 'past', 'the', 'stage', 'of', 'doing',
'anything', 'for', 'a', 'laugh', '.', 'the', 'makers', 'of', 'the', 'film', 'a
lso', 'cared', 'about', 'telling', 'an', 'intelligent', 'story', '.', 'the',
'performances', 'of', 'all', 'the', 'principals', 'are', 'right', 'on', 'the',
'mark', ',', 'particularly', 'john', 'goodman', 'as', 'the', 'schlock', 'maste
r', '.', 'so', 'i', 'would', 'definitely', 'recommend', 'this', 'film', 'to',
'anyone', 'looking', 'for', 'a', 'light', 'hearted', ',', 'yet', 'interestin
g', 'way', 'to', 'spend', 'a', 'couple', 'of', 'hours', '.', 'longer', ',', 'm
ore', 'detailed', 'review', ':', '[', 'beware', 'of', 'spoilers', ']', 'the',
'primary', 'reason', 'i', 'enjoyed', 'this', 'film', 'was', ',', 'that', 'whil
e', 'being', 'a', 'comedy', ',', 'the', 'film', 'also', 'had', 'an', 'intellig
ent', 'story', 'to', 'tell', '.', 'too', 'many', 'comedies', 'today', 'subscri
be', 'to', 'the', 'the', 'notion', 'that', 'a', 'comedy', 'need', 'only', 'mak
e', 'you', 'laugh', '.', 'you', 'watch', 'the', 'movie', ',', 'laugh', 'a', 'l
ot', ',', 'leave', 'the', 'theatre', 'and', 'take', 'nothing', 'with', 'you',
'.', 'matinee', 'is', 'not', 'like', 'that', '.', 'i', 'left', 'the', 'pictur
e', 'thinking', 'about', 'what', 'i', 'would', 'do', 'faced', 'with', 'the',
'cuban', 'missile', 'crisis', '.', 'i', 'found', 'myself', 'wondering', 'wha
t', 'would', 'happen', 'to', 'the', 'characters', 'of', 'the', 'film', '.', 'b
ut', 'most', 'importantly', ',', 'i', 'found', 'myself', 'caring', 'about', 'w
hat', 'would', 'happen', 'to', 'the', 'characters', '.', 'the', 'comedy', 'o
f', 'the', 'film', 'centers', 'around', 'goodman', ',', 'his', 'character',
',', 'and', 'the', 'film', 'he', 'brings', 'to', 'key', 'west', '.', 'i', 'bel
ieve', 'that', 'goodman', 'is', 'one', 'of', 'the', 'finest', 'comedic', 'acto
rs', 'in', 'the', 'business', 'today', '.', 'he', 'is', 'highly', 'expressiv
e', 'both', 'physically', 'and', 'vocally', '.', 'i', 'felt', 'he', 'at', 'lea
st', 'deserved', 'an', 'oscar', 'nomination', 'for', 'his', 'work', 'in', 'bar
ton', 'fink', '.', 'the', 'other', 'characters', 'are', 'often', 'lost', 'in',
'a', 'scene', 'with', 'him', 'due', 'to', 'his', 'commanding', 'nature', ',',
'however', ',', 'while', 'the', 'star', ',', 'goodman', 'is', 'actually', 'no
t', 'at', 'the', 'center', 'of', 'the', 'film', '.', 'the', 'movie', 'is', 're
ally', 'the', 'story', 'of', 'the', 'four', 'teenagers', ',', 'discovering',
'who', 'they', 'are', 'and', 'what', 'they', 'want', ',', 'against', 'a', 'bac
kground', 'where', 'at', 'any', 'minute', 'it', 'could', 'all', 'end', '.',
'i', 'thought', 'the', 'kids', 'reaction', 'were', 'highly', 'realistic', '.',
'they', 'tried', 'to', 'block', 'it', 'out', ',', 'they', 'tried', 'to', 'esca
pe', 'from', 'the', 'concerns', 'of', 'their', 'world', '.', 'unfortunately',
'it', 'kept', 'creeping', 'back', 'in', ',', 'particularly', 'with', 'the', 'f
ear', ',', 'and', 'the', 'chaos', 'of', 'the', 'time', '.', 'while', 'the', 'c
omedy', 'centered', 'on', 'goodman', ',', 'and', 'the', 'drama', 'on', 'the',
'teens', ',', 'there', 'was', 'a', 'great', 'deal', 'of', 'overlap', '.', 'sev
eral', 'aspects', 'of', 'the', 'panic', 'are', 'shown', 'in', 'a', 'humorous',
'light', '.', 'one', 'example', 'is', 'a', 'scene', 'where', 'people', 'are',
'fighting', 'each', 'other', 'for', 'the', 'the', 'last', 'cans', 'and', 'boxe
s', 'of', 'food', 'in', 'a', 'grocery', 'store', '.', 'if', 'you', 'think', 'a
bout', 'it', 'the', 'threat', 'of', 'nuclear', 'annihilation', 'seems', 'hardl
y', 'to', 'be', 'the', 'backdrop', 'for', 'a', 'comedy', ',', 'but', 'it', 'wo
```

```
rks', 'here', '.', 'another', 'reason', 'i', 'like', 'this', 'film', 'is', 'th
at', 'i', 'like', 'b', '-', 'science', 'fiction', 'movies', '.', 'one', 'of',
'my', 'favorite', 'films', 'to', 'go', 'watch', 'is', 'plan', '9', 'from', 'ou
ter', 'space', '.', '(', 'note', 'i', 'did', 'not', 'say', 'it', 'was', 'one',
'of', 'my', 'favorite', 'movies', ',', 'but', 'one', 'of', 'my', 'favorite',
'to', 'see', '.', ')', 'while', 'mant', '!', 'never', 'got', 'made', ',', 'man
y', 'films', 'like', 'it', 'were', ',', 'and', 'mant', '!', 'serves', 'mostl
y', 'as', 'dante', "'", 's', 'homage', 'to', 'the', 'b', '-', 'films', 'he',
'loves', '.', 'so', 'again', 'i', 'would', 'like', 'to', 'recommend', 'matine
e', 'to', 'anybody', 'looking', 'for', 'a', 'good', ',', 'humorous', 'story',
'.', 'this', 'isn', "'", 't', 'a', 'gag', 'film', 'like', 'many', 'other', 'co
medies', 'but', 'an', 'intelligent', ',', 'well', '-', 'thought', 'out', ',',
'film', 'about', 'real', 'people', 'with', 'real', 'problems', 'told', 'in',
'an', 'often', 'hilarious', 'way', '.', 'enjoy', '!']
pos
```

In [ ]:

```python
# Exploring the dataset

# Fequency distribution of most common words
nltk.FreqDist(movie_reviews.words()).most_common(15)
```

Out[ ]:

```
[(',', 77717),
 ('the', 76529),
 ('.', 65876),
 ('a', 38106),
 ('and', 35576),
 ('of', 34123),
 ('to', 31937),
 ("'", 30585),
 ('is', 25195),
 ('in', 21822),
 ('s', 18513),
 ('"', 17612),
 ('it', 16107),
 ('that', 15924),
 ('-', 15595)]
```

In [ ]:

```python
# Printing file IDs of first ten pos reviews
movie_reviews.fileids('pos')[:10]
```

Out[ ]:

```
['pos/cv000_29590.txt',
 'pos/cv001_18431.txt',
 'pos/cv002_15918.txt',
 'pos/cv003_11664.txt',
 'pos/cv004_11636.txt',
 'pos/cv005_29443.txt',
 'pos/cv006_15448.txt',
 'pos/cv007_4968.txt',
 'pos/cv008_29435.txt',
 'pos/cv009_29592.txt']
```

In [ ]:

```python
# Printing file IDs of first ten neg reviews
movie_reviews.fileids('neg')[:10]
```

Out[ ]:
```
['neg/cv000_29416.txt',
 'neg/cv001_19502.txt',
 'neg/cv002_17424.txt',
 'neg/cv003_12683.txt',
 'neg/cv004_12641.txt',
 'neg/cv005_29357.txt',
 'neg/cv006_17022.txt',
 'neg/cv007_4992.txt',
 'neg/cv008_29326.txt',
 'neg/cv009_29417.txt']
```

In [ ]:
```
# Printing words with ID pos/cv995_21821.txt
movie_reviews.words('pos/cv995_21821.txt')
```

Out[ ]:
```
['wow', '!', 'what', 'a', 'movie', '.', 'it', "'", 's', ...]
```

In [ ]:
```
# Print the first review and its label
print("Sample Review:", documents[0][0][:10]) # Displaying the first 10 words
print("Label:", documents[0][1])
```

```
Sample Review: ['--', 'comedy', ',', 'rated', 'pg', ',', 'runs', 'about', '1',
':']
Label: pos
```

In [ ]:
```
# Preprocessing
# Instances for Lemmatization
lemmatizer = WordNetLemmatizer()

processed_data = []

# Iterating through the words and respective categories in doctumenst
for words, category in documents:


    # Using the word tokentizer library from nltk to create tokens
    token_data = word_tokenize(" ".join(words))
    # Removing the stop words
    filtered_token_data = [token.lower() for token in token_data if token.lowe
    # Lemmatization
    lemmatized_token_data = [lemmatizer.lemmatize(token) for token in filtered_

    # Appending the processed output with categories
    processed_data.append((lemmatized_token_data, category))
```

In [ ]:
```
print("Original Words:", documents[0][0])
print("Preprocessed Words:", processed_data[0][0])
print("Category:", processed_data[0][1])
```

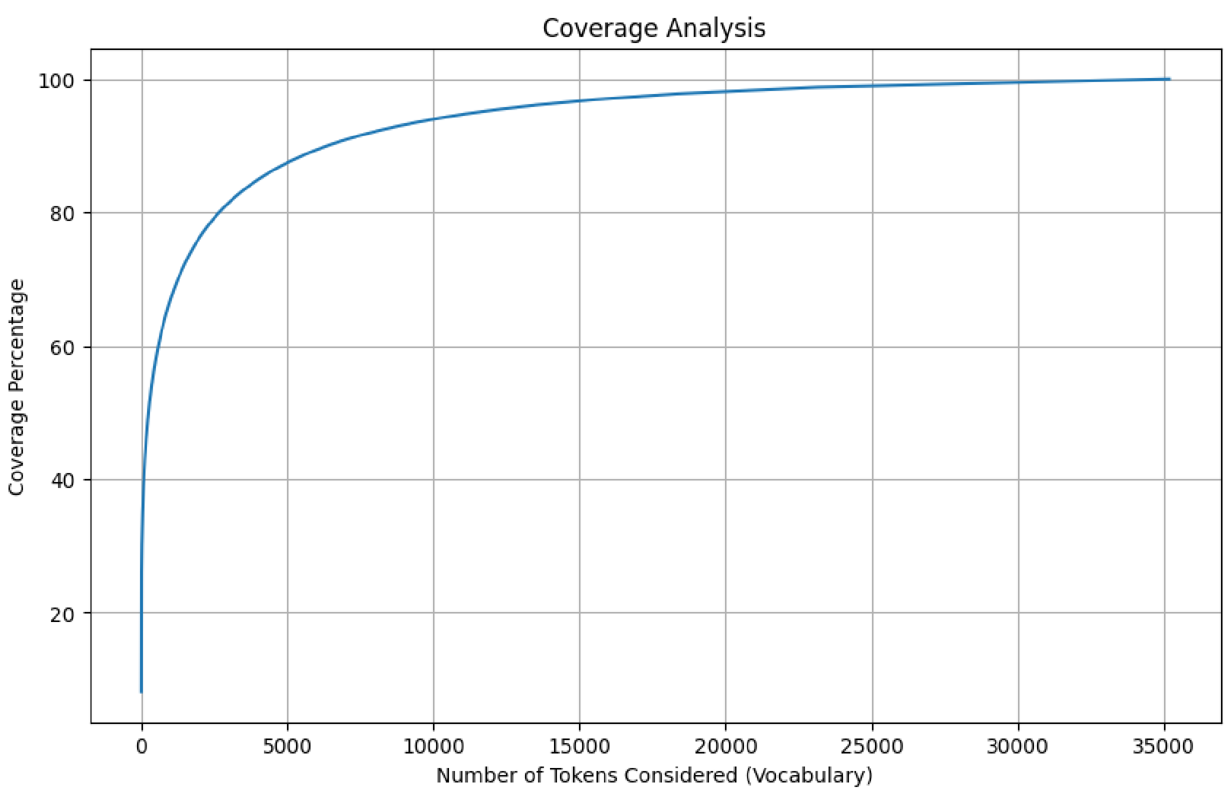
```
Original Words: ['--', 'comedy', ',', 'rated', 'pg', ',', 'runs', 'about',
'1', ':', '40', '--', 'starring', ':', 'john', 'goodman', ',', 'kathy', 'moria
rty', ',', 'and', 'a', 'bunch', 'of', 'teenagers', '--', 'directed', 'by', 'jo
e', 'dante', 'and', 'written', 'by', 'charles', 'hass', '--', 'summary', ':',
'lawrence', 'woolsley', '(', 'john', 'goodman', ')', 'brings', 'his', 'new',
'horror', 'film', 'mant', '!', 'to', 'premiere', 'in', 'key', 'west', 'durin
g', 'the', 'height', 'of', 'the', 'cuban', 'missile', 'crisis', '.', 'he', 'ho
pes', 'to', 'capitalize', 'on', 'the', 'tense', 'moment', 'by', 'providing',
'an', 'escape', 'for', 'the', 'town', '.', 'we', 'see', 'most', 'of', 'the',
'events', 'through', 'the', 'stories', 'of', 'four', 'teenagers', 'and', 'ho
w', 'life', 'affects', 'them', '.', 'quick', 'and', 'easy', 'review', ':',
'i', 'really', 'enjoyed', 'matinee', '.', 'the', 'mixture', 'of', 'comedy', 'a
nd', 'tension', 'blended', 'nicely', '.', 'unlike', 'many', 'comedies', 'thi
s', 'film', 'tries', ',', 'and', 'succeeds', ',', 'in', 'getting', 'past', 'th
e', 'stage', 'of', 'doing', 'anything', 'for', 'a', 'laugh', '.', 'the', 'make
rs', 'of', 'the', 'film', 'also', 'cared', 'about', 'telling', 'an', 'intellig
ent', 'story', '.', 'the', 'performances', 'of', 'all', 'the', 'principals',
'are', 'right', 'on', 'the', 'mark', ',', 'particularly', 'john', 'goodman',
'as', 'the', 'schlock', 'master', '.', 'so', 'i', 'would', 'definitely', 'reco
mmend', 'this', 'film', 'to', 'anyone', 'looking', 'for', 'a', 'light', 'heart
ed', ',', 'yet', 'interesting', 'way', 'to', 'spend', 'a', 'couple', 'of', 'ho
urs', '.', 'longer', ',', 'more', 'detailed', 'review', ':', '[', 'beware', 'o
f', 'spoilers', ']', 'the', 'primary', 'reason', 'i', 'enjoyed', 'this', 'fil
m', 'was', ',', 'that', 'while', 'being', 'a', 'comedy', ',', 'the', 'film',
'also', 'had', 'an', 'intelligent', 'story', 'to', 'tell', '.', 'too', 'many',
'comedies', 'today', 'subscribe', 'to', 'the', 'the', 'notion', 'that', 'a',
'comedy', 'need', 'only', 'make', 'you', 'laugh', '.', 'you', 'watch', 'the',
'movie', ',', 'laugh', 'a', 'lot', ',', 'leave', 'the', 'theatre', 'and', 'tak
e', 'nothing', 'with', 'you', '.', 'matinee', 'is', 'not', 'like', 'that',
'.', 'i', 'left', 'the', 'picture', 'thinking', 'about', 'what', 'i', 'would',
'do', 'faced', 'with', 'the', 'cuban', 'missile', 'crisis', '.', 'i', 'found',
'myself', 'wondering', 'what', 'would', 'happen', 'to', 'the', 'characters',
'of', 'the', 'film', '.', 'but', 'most', 'importantly', ',', 'i', 'found', 'my
self', 'caring', 'about', 'what', 'would', 'happen', 'to', 'the', 'character
s', '.', 'the', 'comedy', 'of', 'the', 'film', 'centers', 'around', 'goodman',
',', 'his', 'character', ',', 'and', 'the', 'film', 'he', 'brings', 'to', 'ke
y', 'west', '.', 'i', 'believe', 'that', 'goodman', 'is', 'one', 'of', 'the',
'finest', 'comedic', 'actors', 'in', 'the', 'business', 'today', '.', 'he', 'i
s', 'highly', 'expressive', 'both', 'physically', 'and', 'vocally', '.', 'i',
'felt', 'he', 'at', 'least', 'deserved', 'an', 'oscar', 'nomination', 'for',
'his', 'work', 'in', 'barton', 'fink', '.', 'the', 'other', 'characters', 'ar
e', 'often', 'lost', 'in', 'a', 'scene', 'with', 'him', 'due', 'to', 'his', 'c
ommanding', 'nature', ',', 'however', ',', 'while', 'the', 'star', ',', 'goodm
an', 'is', 'actually', 'not', 'at', 'the', 'center', 'of', 'the', 'film', '.',
'the', 'movie', 'is', 'really', 'the', 'story', 'of', 'the', 'four', 'teenager
s', ',', 'discovering', 'who', 'they', 'are', 'and', 'what', 'they', 'want',
',', 'against', 'a', 'background', 'where', 'at', 'any', 'minute', 'it', 'coul
d', 'all', 'end', '.', 'i', 'thought', 'the', 'kids', 'reaction', 'were', 'hig
hly', 'realistic', '.', 'they', 'tried', 'to', 'block', 'it', 'out', ',', 'the
y', 'tried', 'to', 'escape', 'from', 'the', 'concerns', 'of', 'their', 'worl
d', '.', 'unfortunately', 'it', 'kept', 'creeping', 'back', 'in', ',', 'partic
ularly', 'with', 'the', 'fear', ',', 'and', 'the', 'chaos', 'of', 'the', 'tim
e', '.', 'while', 'the', 'comedy', 'centered', 'on', 'goodman', ',', 'and', 't
he', 'drama', 'on', 'the', 'teens', ',', 'there', 'was', 'a', 'great', 'deal',
'of', 'overlap', '.', 'several', 'aspects', 'of', 'the', 'panic', 'are', 'show
n', 'in', 'a', 'humorous', 'light', '.', 'one', 'example', 'is', 'a', 'scene',
'where', 'people', 'are', 'fighting', 'each', 'other', 'for', 'the', 'the', 'l
ast', 'cans', 'and', 'boxes', 'of', 'food', 'in', 'a', 'grocery', 'store',
'.', 'if', 'you', 'think', 'about', 'it', 'the', 'threat', 'of', 'nuclear', 'a
nnihilation', 'seems', 'hardly', 'to', 'be', 'the', 'backdrop', 'for', 'a', 'c
```

```
omedy', ',', 'but', 'it', 'works', 'here', '.', 'another', 'reason', 'i', 'lik
e', 'this', 'film', 'is', 'that', 'i', 'like', 'b', '-', 'science', 'fiction',
'movies', '.', 'one', 'of', 'my', 'favorite', 'films', 'to', 'go', 'watch', 'i
s', 'plan', '9', 'from', 'outer', 'space', '.', '(', 'note', 'i', 'did', 'no
t', 'say', 'it', 'was', 'one', 'of', 'my', 'favorite', 'movies', ',', 'but',
'one', 'of', 'my', 'favorite', 'to', 'see', '.', ')', 'while', 'mant', '!', 'n
ever', 'got', 'made', ',', 'many', 'films', 'like', 'it', 'were', ',', 'and',
'mant', '!', 'serves', 'mostly', 'as', 'dante', "'", 's', 'homage', 'to', 'th
e', 'b', '-', 'films', 'he', 'loves', '.', 'so', 'again', 'i', 'would', 'lik
e', 'to', 'recommend', 'matinee', 'to', 'anybody', 'looking', 'for', 'a', 'goo
d', ',', 'humorous', 'story', '.', 'this', 'isn', "'", 't', 'a', 'gag', 'fil
m', 'like', 'many', 'other', 'comedies', 'but', 'an', 'intelligent', ',', 'wel
l', '-', 'thought', 'out', ',', 'film', 'about', 'real', 'people', 'with', 're
al', 'problems', 'told', 'in', 'an', 'often', 'hilarious', 'way', '.', 'enjo
y', '!']
Preprocessed Words: ['--', 'comedy', ',', 'rated', 'pg', ',', 'run', '1', ':',
'40', '--', 'starring', ':', 'john', 'goodman', ',', 'kathy', 'moriarty', ',',
'bunch', 'teenager', '--', 'directed', 'joe', 'dante', 'written', 'charles',
'ha', '--', 'summary', ':', 'lawrence', 'woolsley', '(', 'john', 'goodman',
')', 'brings', 'new', 'horror', 'film', 'mant', '!', 'premiere', 'key', 'wes
t', 'height', 'cuban', 'missile', 'crisis', '.', 'hope', 'capitalize', 'tens
e', 'moment', 'providing', 'escape', 'town', '.', 'see', 'event', 'story', 'fo
ur', 'teenager', 'life', 'affect', '.', 'quick', 'easy', 'review', ':', 'reall
y', 'enjoyed', 'matinee', '.', 'mixture', 'comedy', 'tension', 'blended', 'nic
ely', '.', 'unlike', 'many', 'comedy', 'film', 'try', ',', 'succeeds', ',', 'g
etting', 'past', 'stage', 'anything', 'laugh', '.', 'maker', 'film', 'also',
'cared', 'telling', 'intelligent', 'story', '.', 'performance', 'principal',
'right', 'mark', ',', 'particularly', 'john', 'goodman', 'schlock', 'master',
'.', 'would', 'definitely', 'recommend', 'film', 'anyone', 'looking', 'light',
'hearted', ',', 'yet', 'interesting', 'way', 'spend', 'couple', 'hour', '.',
'longer', ',', 'detailed', 'review', ':', '[', 'beware', 'spoiler', ']', 'prim
ary', 'reason', 'enjoyed', 'film', ',', 'comedy', ',', 'film', 'also', 'intell
igent', 'story', 'tell', '.', 'many', 'comedy', 'today', 'subscribe', 'notio
n', 'comedy', 'need', 'make', 'laugh', '.', 'watch', 'movie', ',', 'laugh', 'l
ot', ',', 'leave', 'theatre', 'take', 'nothing', '.', 'matinee', 'like', '.',
'left', 'picture', 'thinking', 'would', 'faced', 'cuban', 'missile', 'crisis',
'.', 'found', 'wondering', 'would', 'happen', 'character', 'film', '.', 'impor
tantly', ',', 'found', 'caring', 'would', 'happen', 'character', '.', 'comed
y', 'film', 'center', 'around', 'goodman', ',', 'character', ',', 'film', 'bri
ngs', 'key', 'west', '.', 'believe', 'goodman', 'one', 'finest', 'comedic', 'a
ctor', 'business', 'today', '.', 'highly', 'expressive', 'physically', 'vocall
y', '.', 'felt', 'least', 'deserved', 'oscar', 'nomination', 'work', 'barton',
'fink', '.', 'character', 'often', 'lost', 'scene', 'due', 'commanding', 'natu
re', ',', 'however', ',', 'star', ',', 'goodman', 'actually', 'center', 'fil
m', '.', 'movie', 'really', 'story', 'four', 'teenager', ',', 'discovering',
'want', ',', 'background', 'minute', 'could', 'end', '.', 'thought', 'kid', 'r
eaction', 'highly', 'realistic', '.', 'tried', 'block', ',', 'tried', 'escap
e', 'concern', 'world', '.', 'unfortunately', 'kept', 'creeping', 'back', ',',
'particularly', 'fear', ',', 'chaos', 'time', '.', 'comedy', 'centered', 'good
man', ',', 'drama', 'teen', ',', 'great', 'deal', 'overlap', '.', 'several',
'aspect', 'panic', 'shown', 'humorous', 'light', '.', 'one', 'example', 'scen
e', 'people', 'fighting', 'last', 'can', 'box', 'food', 'grocery', 'store',
'.', 'think', 'threat', 'nuclear', 'annihilation', 'seems', 'hardly', 'backdro
p', 'comedy', ',', 'work', '.', 'another', 'reason', 'like', 'film', 'like',
'b', '-', 'science', 'fiction', 'movie', '.', 'one', 'favorite', 'film', 'go',
'watch', 'plan', '9', 'outer', 'space', '.', '(', 'note', 'say', 'one', 'favor
ite', 'movie', ',', 'one', 'favorite', 'see', '.', ')', 'mant', '!', 'never',
'got', 'made', ',', 'many', 'film', 'like', ',', 'mant', '!', 'serf', 'mostl
y', 'dante', "'", 'homage', 'b', '-', 'film', 'love', '.', 'would', 'like', 'r
ecommend', 'matinee', 'anybody', 'looking', 'good', ',', 'humorous', 'story',
```

```
'.', "'", 'gag', 'film', 'like', 'many', 'comedy', 'intelligent', ',', 'well',
'-', 'thought', ',', 'film', 'real', 'people', 'real', 'problem', 'told', 'oft
en', 'hilarious', 'way', '.', 'enjoy', '!']
Category: pos
```

In [ ]:

```
preprocessed_data_words = [word for doc, _ in processed_data for word in doc]

# Counting the frequency of words
frequency = Counter(preprocessed_data_words)

# Sorting words by frequency
words_sorted = sorted(frequency, key=frequency.get, reverse=True)

# Calculate cumulative coverage
coverage = []
word_unique = 0
total = sum(frequency.values())

for word in words_sorted:
    word_unique += frequency[word]
    temp = word_unique / total * 100
    coverage.append(temp)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(coverage)
plt.xlabel('Number of Tokens Considered (Vocabulary)')
plt.ylabel('Coverage Percentage')
plt.title('Coverage Analysis')
plt.grid(True)
plt.show()
```



Coverage Analysis Discussion and Rationalization for Vocabulary Choice:

From the graph, we can observe that as the number of tokens or words in the vocabulary increases, the coverage percentage depicts a logarithmic increase pattern which means that additions to the vocabulary contribute significantly to increasing coverage.

We can also note that the rate of increase in coverage starts to stabilize when the vocabulary token count reaches approximately 30,000.

Depending on the desired coverage, we can stop training early on. After a point, to increase the coverage by a small number we will have to train for much more iterations. This can be seen from the above graph. We achieve 90% coverage when our vocabulary size is 7500. But to achieve a coverage of 95% we need a vocabulary size of 12500.

The decision on vocabulary size is a balancing act. It involves considering the computational resources, the nature of the text data, the specifics of the application, and the characteristics of the chosen algorithms. A smaller vocabulary may lead to faster, more efficient models but at the risk of losing potentially valuable information. Conversely, a larger vocabulary could capture more details but may lead to increased complexity, slower performance, and potential overfitting.

Algorithm Implementation

In [ ]:
```
# Vectors
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer

# Algorithms
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier

# Model Selection
from sklearn.model_selection import train_test_split,cross_validate


# Evaluation metrics
from sklearn.metrics import accuracy_score,roc_auc_score
from sklearn.metrics import classification_report
```

In [ ]:
```
X = [" ".join(tokens) for tokens, category in processed_data]
y = [category for tokens, category in processed_data]
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size= 0.20, ra
```

In [ ]:
```
# TF feature representations
tf = CountVectorizer()
X_train_tf = tf.fit_transform(X_train)
X_test_tf = tf.transform(X_test)
```

```
# TF-IDF feature representations
tfidf = TfidfVectorizer()
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

Naive Bayes

In [ ]:
```
# Creating instanaces of  Multinomial Naive Bayes
naive_bayes_tf = MultinomialNB()
naive_bayes_tfidf = MultinomialNB()

# Training with TF and TFIDF feature represntation
naive_bayes_tf.fit(X_train_tf, y_train)
naive_bayes_tfidf.fit(X_train_tfidf, y_train)

# Predicting
y_pred_nb_tf = naive_bayes_tf.predict(X_test_tf)
y_pred_nb_tfidf = naive_bayes_tfidf.predict(X_test_tfidf)
```

In [ ]:
```
# Evaluating the model with TF features
accuracy_nb_tf = accuracy_score(y_test, y_pred_nb_tf)
report_nb_tf = classification_report(y_test, y_pred_nb_tf)
print("Accuracy with TF feature representations for Naive Bayes", accuracy_nb_
print("Classification Report with TF feature representations for Naive Bayes",
```

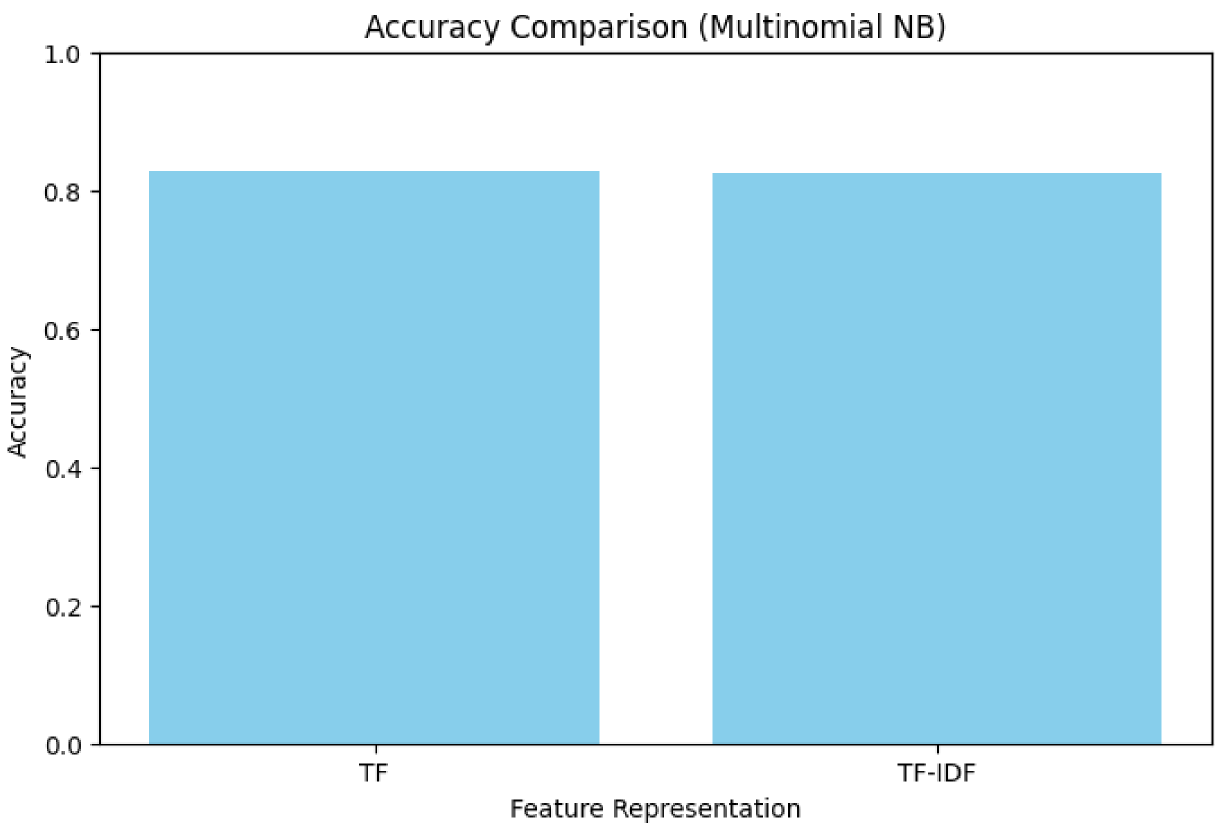
```
Accuracy with TF feature representations for Naive Bayes 0.8275
Classification Report with TF feature representations for Naive Bayes
precision    recall  f1-score   support

         neg       0.82      0.83      0.83       200
         pos       0.83      0.82      0.83       200

    accuracy                           0.83       400
   macro avg       0.83      0.83      0.83       400
weighted avg       0.83      0.83      0.83       400
```

In [ ]:
```
# Evaluating the model with TFIDF features
accuracy_nb_tfidf = accuracy_score(y_test, y_pred_nb_tfidf)
report_nb_tfidf = classification_report(y_test, y_pred_nb_tfidf)
print("Accuracy with TFIDF feature representations for Naive Bayes", accuracy_
print("Classification Report with TFIDF feature representations for Naive Baye
```

```
Accuracy with TFIDF feature representations for Naive Bayes 0.825
Classification Report with TFIDF feature representations for Naive Bayes
precision    recall  f1-score   support

         neg       0.81      0.85      0.83       200
         pos       0.84      0.80      0.82       200

    accuracy                           0.82       400
   macro avg       0.83      0.82      0.82       400
weighted avg       0.83      0.82      0.82       400
```

In [ ]:
```
# Plot the comparison
algorithms = ['TF', 'TF-IDF']
```

```
accuracies = [accuracy_nb_tf, accuracy_nb_tfidf]

plt.figure(figsize=(8, 5))
plt.bar(algorithms, accuracies, color='skyblue')
plt.xlabel('Feature Representation')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison (Multinomial NB)')
plt.ylim(0, 1)
plt.show()
```



Logistic Regression

In [ ]:
```
# Creating instanaces of  Logistic Regression
lr_tf = LogisticRegression()
lr_tfidf = LogisticRegression()

# Training with TF and TFIDF feature represntation
lr_tf.fit(X_train_tf, y_train)
lr_tfidf.fit(X_train_tfidf, y_train)

# Predicting
y_pred_lr_tf = lr_tf.predict(X_test_tf)
y_pred_lr_tfidf = lr_tfidf.predict(X_test_tfidf)
```

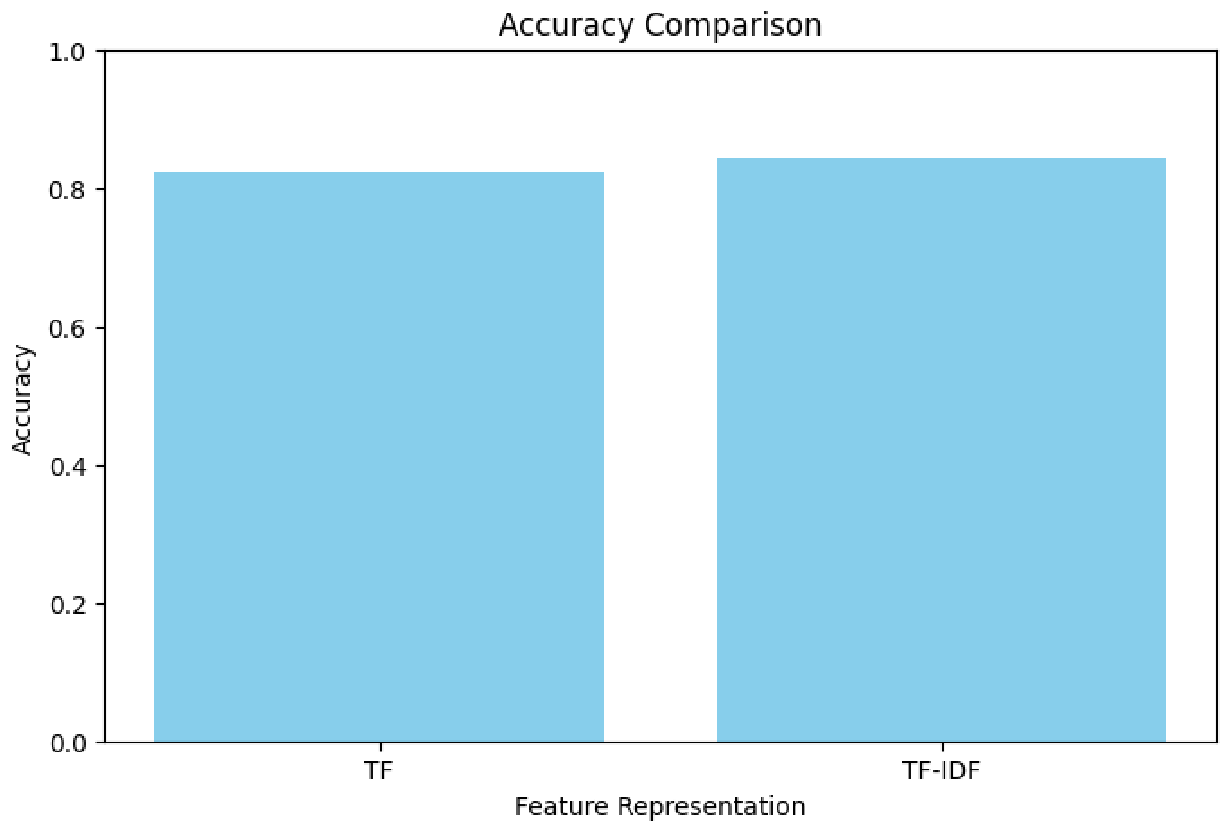
In [ ]:
```
# Evaluating the model with TF features
accuracy_lr_tf = accuracy_score(y_test, y_pred_lr_tf)
report_lr_tf = classification_report(y_test, y_pred_lr_tf)
print("Accuracy with TF feature representations for Logistic Regression", accu
print("Classification Report with TF feature representations for Logistic Regre
```

```
Accuracy with TF feature representations for Logistic Regression 0.8225
Classification Report with TF feature representations for Logistic Regression
precision    recall  f1-score   support

         neg       0.82      0.82      0.82       200
         pos       0.82      0.82      0.82       200

    accuracy                           0.82       400
   macro avg       0.82      0.82      0.82       400
weighted avg       0.82      0.82      0.82       400
```

In [ ]:
```
# Evaluating the model with TFIDF features
accuracy_lr_tfidf = accuracy_score(y_test, y_pred_lr_tfidf)
report_lr_tfidf = classification_report(y_test, y_pred_lr_tfidf)
print("Accuracy with TFIDF feature representations for Logistic Regression", ac
print("Classification Report with TFIDF feature representations for Logistic Re
```

```
Accuracy with TFIDF feature representations for Logistic Regression 0.8425
Classification Report with TFIDF feature representations for Logistic Regressi
on                precision    recall  f1-score   support

         neg       0.86      0.81      0.84       200
         pos       0.82      0.87      0.85       200

    accuracy                           0.84       400
   macro avg       0.84      0.84      0.84       400
weighted avg       0.84      0.84      0.84       400
```

In [ ]:
```
# Plotting the comparison
algorithms = ['TF', 'TF-IDF']
accuracies = [accuracy_lr_tf, accuracy_lr_tfidf]

plt.figure(figsize=(8, 5))
plt.bar(algorithms, accuracies, color='skyblue')
plt.xlabel('Feature Representation')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison')
plt.ylim(0, 1)
plt.show()
```



MLP

In [ ]:
```
# Creating a list with various hidden layer sizes to test the MLP model
mlp = [
    (50,),             # Single hidden layer - 50 neurons
    (100,),            # Single hidden layer - 100 neurons
    (50, 50),          # 2 hidden layers - 50 neurons each
    (50, 100),         # 2 hidden layers - 50 neurons each and 100 neuron each
]


toReturn = []

# Iterating though the various sizes in the list and assigning them to the hidd
for architecture in mlp:

    # Instance of MLP Classififer for TF and TFIDF feature represenations
    mlp_tf = MLPClassifier(hidden_layer_sizes=architecture, max_iter=300, rando
    mlp_tfidf = MLPClassifier(hidden_layer_sizes=architecture, max_iter=300, ra

    # Fitting both the models
    mlp_tf.fit(X_train_tf, y_train)
    mlp_tfidf.fit(X_train_tfidf, y_train)

    # Predicting on the testing set
    y_pred_mlp_tf = mlp_tf.predict(X_test_tf)
    y_pred_mlp_tfidf = mlp_tfidf.predict(X_test_tfidf)

    # Calculating the accuracy and other metrics for both the models
    accuracy_mlp_tf = accuracy_score(y_test, y_pred_mlp_tf)
    report_mlp_tf = classification_report(y_test, y_pred_mlp_tf)
```

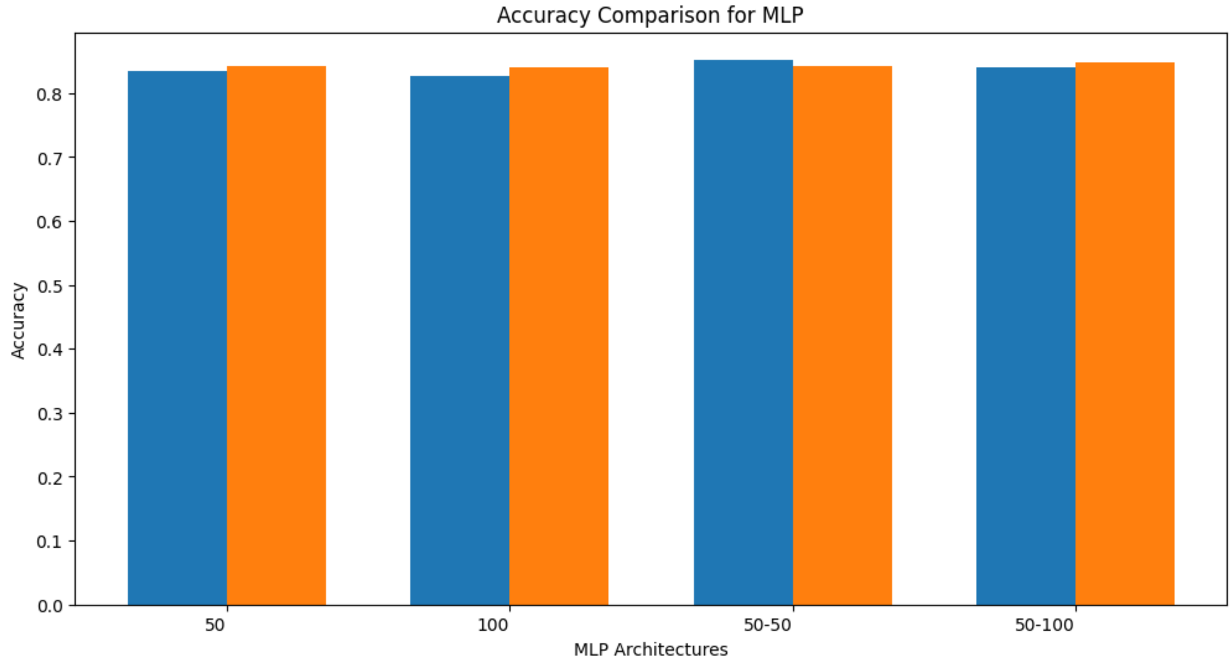
```
    accuracy_mlp_tfidf = accuracy_score(y_test, y_pred_mlp_tfidf)
    report_mlp_tfidf = classification_report(y_test, y_pred_mlp_tfidf)

    # Returning the calculated values
    toReturn.append({
        'architecture': architecture,
        'accuracy_mlp_tf': accuracy_mlp_tf,
        'report_mlp_tf' : report_mlp_tf,
        'accuracy_mlp_tfidf': accuracy_mlp_tfidf,
        'report_mlp_tfidf' : report_mlp_tfidf
    })

# Pring the values
for result in toReturn:
    print("Architecture ", result['architecture'])
    print("Accuracy with TF feature representations for MLP ", result['accuracy
    print("Report with TF feature representations for MLP", result['report_mlp_
    print("Accuracy with TFIDF feature representations for MLP", result['accura
    print("Report with TFIDF feature representations for MLP", result['report_r
    print()
```

```
Architecture  (50,)
Accuracy with TF feature representations for MLP  0.835
Report with TF feature representations for MLP               precision    reca
ll  f1-score   support

         neg       0.83      0.84      0.84       200
         pos       0.84      0.83      0.83       200

    accuracy                           0.83       400
   macro avg       0.84      0.83      0.83       400
weighted avg       0.84      0.83      0.83       400

Accuracy with TFIDF feature representations for MLP 0.8425
Report with TFIDF feature representations for MLP               precision    r
ecall  f1-score   support

         neg       0.85      0.83      0.84       200
         pos       0.83      0.85      0.84       200

    accuracy                           0.84       400
   macro avg       0.84      0.84      0.84       400
weighted avg       0.84      0.84      0.84       400


Architecture  (100,)
Accuracy with TF feature representations for MLP  0.8275
Report with TF feature representations for MLP               precision    reca
ll  f1-score   support

         neg       0.82      0.84      0.83       200
         pos       0.84      0.81      0.83       200

    accuracy                           0.83       400
   macro avg       0.83      0.83      0.83       400
weighted avg       0.83      0.83      0.83       400

Accuracy with TFIDF feature representations for MLP 0.84
Report with TFIDF feature representations for MLP               precision    r
ecall  f1-score   support

         neg       0.85      0.83      0.84       200
         pos       0.83      0.85      0.84       200

    accuracy                           0.84       400
   macro avg       0.84      0.84      0.84       400
weighted avg       0.84      0.84      0.84       400


Architecture  (50, 50)
Accuracy with TF feature representations for MLP  0.8525
Report with TF feature representations for MLP               precision    reca
ll  f1-score   support

         neg       0.85      0.86      0.85       200
         pos       0.86      0.84      0.85       200

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400
```

```
Accuracy with TFIDF feature representations for MLP 0.8425
Report with TFIDF feature representations for MLP               precision    r
ecall  f1-score   support

         neg       0.85      0.83      0.84       200
         pos       0.84      0.85      0.84       200

    accuracy                           0.84       400
   macro avg       0.84      0.84      0.84       400
weighted avg       0.84      0.84      0.84       400


Architecture  (50, 100)
Accuracy with TF feature representations for MLP  0.84
Report with TF feature representations for MLP               precision    reca
ll  f1-score   support

         neg       0.84      0.84      0.84       200
         pos       0.84      0.84      0.84       200

    accuracy                           0.84       400
   macro avg       0.84      0.84      0.84       400
weighted avg       0.84      0.84      0.84       400

Accuracy with TFIDF feature representations for MLP 0.8475
Report with TFIDF feature representations for MLP               precision    r
ecall  f1-score   support

         neg       0.87      0.82      0.84       200
         pos       0.83      0.88      0.85       200

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400
```

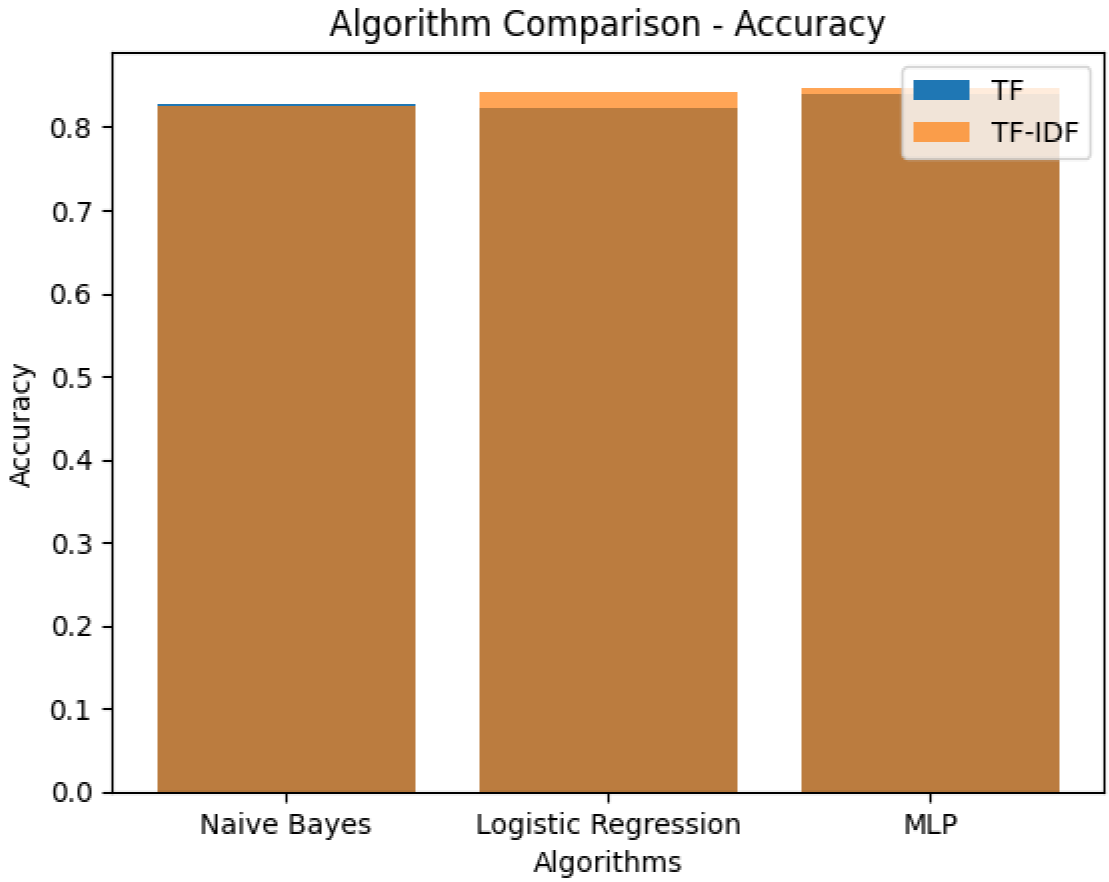
In [ ]:

```python
# Plotting the comparison

architectures = ['-'.join(str(layer) for layer in result['architecture']) for
accuracies_tf = [result['accuracy_mlp_tf'] for result in toReturn]
accuracies_tfidf = [result['accuracy_mlp_tfidf'] for result in toReturn]

# Create an array of x-axis positions for bars
x = np.arange(len(architectures))
bar_width = 0.35

# Creating the bar chart
plt.figure(figsize=(12, 6))
plt.bar(x - bar_width/2, accuracies_tf, bar_width, label='TF')
plt.bar(x + bar_width/2, accuracies_tfidf, bar_width, label='TF-IDF')
plt.xlabel('MLP Architectures')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison for MLP')
plt.xticks(x, architectures, ha='right')
plt.show()
```



Discussion and Analysis

In [ ]:
```
# Comparisons of all the algorithms
algo = ['Naive Bayes', 'Logistic Regression', 'MLP']
accuracy_tf = [accuracy_nb_tf, accuracy_lr_tf, accuracy_mlp_tf]
accuracy_tfidf = [accuracy_nb_tfidf, accuracy_lr_tfidf, accuracy_mlp_tfidf]

plt.bar(algo, accuracy_tf, label='TF')
plt.bar(algo, accuracy_tfidf, label='TF-IDF', alpha=0.7)
plt.xlabel('Algorithm')
plt.ylabel('Accuracy Percentage')
plt.title('Algorithm Comparison - Accuracy')
plt.legend()
plt.show()
```



Comparing the results by all the algorithms:

The performance of each algorithm can vary based on several factors, such as pre-processing steps, the number of iterations, and the choice of parameters. In our analysis, we applied pre-processing steps such astokenization using the NLTK Punkt tokenizer, removal of stop-words, and lemmatization. Furthermore, we trained all three algorithms — Multinomial Naive Bayes (NB), Logistic Regression, and Multilayer Perceptron (MLP) — using both Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) feature representations.

In case of Naive Bayes, the performance was relatively consistent across both TF and TF-IDF, with accuracies of 82.75% for TF and 82.5% for TF-IDF. This slight preference for TF could be due to Naive Bayes' probabilistic nature, which may not fully leverage the additional information provided by TFIDF.

Both MLP and Logistic Regression algorithms showed improved results with TF-IDF over TF. This outcome suggests that the weighting of terms provided by TF-IDF is beneficial in these models.

The MLP algorithm was trained also using various architectures. MLP performed better with TF in the two-layer, 50-neuron each setup. However, in all other configurations — whether a single layer with 50 or 100 neurons, or a two-layer setup with 50 and 100 neurons, the algorithm showed better performance with TF-IDF. This suggests that the complexity of the network architecture can influence the effectiveness of the feature representation.

Discussion Report:

Term Frequency (TF): TF simply counts the number of times word appear in a document. TF is easy to implement which can be helpful in case of simpler tasks.

Term Frequency-Inverse Document Frequency (TF-IDF): TFIDF changes the term frequency by scaling it with the inverse document frequency. This reduces the weight of terms that appear frequently in the document and vice versa. This leads to a more balanced representation where the importance of rare but significant terms are still highlighted.

We can decide the type of feature representations based on the task we have. While TF is simple and easy to implement, TFIDF balances the TF with the importance of terms which can often lead to better results.

Comparsion of Algorithms:

Multinomial Naive Bayes : Naive Bayes is often considered as the baseline classifier for text classification tasks such as sentiment analysis. It is a simple classifier and is known for it's speed. However, it assumes that the tokens are independent, which is not the case in most situations where context and word order matter. Due to it's simplicity, it may not perform as well with complex features.

Logistic Regression : Logistic Regression can handle various relationships in data and does not assume it's independent in comparison to Naive Bayes. Even though it provides good results with feature engineering, it can struggle in case where there are non-linear relationships in data.

Multilayer Perceptron : MLP is a neural network architecture hence is capable of capturing xomplex relationships within data. It is also capable of modelling non-linear data but does require more computational resources to train the models depending on the size.