```
# -*- coding: utf-8 -*-
"""DecisionTree _IMDB

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1EPnrLQM4B7GGyj5ftKmftcNLMJMVv1_f
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import string
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from google.colab import drive
drive.mount('/content/drive')

path = '/content/drive/MyDrive/IMDB Dataset.csv'

imdb = pd.read_csv(path)

imdb.shape

imdb.head()

imdb['review'][1]

review = imdb['review']
labels = imdb['sentiment']

"""Pre-processing the reviews"""

#start replaceTwoOrMore
def replaceTwoOrMore(s):
    #look for 2 or more repetitions of character and replace with the character itself
    pattern = re.compile(r"(.)\1{1,}", re.DOTALL)
    return pattern.sub(r"\1\1", s)

#start process_review
def processReview(review):
    # Removing numbers
    review = re.sub('[0-9]', '', review)
    #remove HTML tags
    cleanr=re.compile('<.*?>')
    review=re.sub(cleanr,' ',review)
    #Convert to lower case
    review = review.lower()
    review = review.translate(str.maketrans('', '', string.punctuation))
    #Remove additional white spaces
    review = re.sub('[\s]+', ' ', review)
    #Replace #word with word
    review = re.sub(r'#([^\s]+)', r'\1', review)
    #trim
    review = review.strip('\'"')
    review = review.strip('.,')
    review = replaceTwoOrMore(review)
    return review

processedReviews = []
for review in review:
  processedReviews.append(processReview(review))

processedReviews[1]

vectorizer = CountVectorizer(analyzer='word') # Convert a collection of text documents to a matrix of token counts.
featurevector = vectorizer.fit_transform(processedReviews)

featurevector.shape

"""
```

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> vectorizer.get_feature_names_out()
array(['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third',
       'this'], ...)
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
>>> vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2))
>>> X2 = vectorizer2.fit_transform(corpus)
>>> vectorizer2.get_feature_names_out()
array(['and this', 'document is', 'first document', 'is the', 'is this',
       'second document', 'the first', 'the second', 'the third', 'third one',
       'this document', 'this is', 'this the'], ...)
 >>> print(X2.toarray())
 [[0 0 1 1 0 0 1 0 0 0 0 1 0]
 [0 1 0 1 0 1 0 1 0 0 1 0 0]
 [1 0 0 1 0 0 0 0 1 1 0 1 0]
 [0 0 1 0 1 0 1 0 0 0 0 0 1]]
 """

X_train, X_test, y_train, y_test = train_test_split(featurevector, labels, test_size=0.30, random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape

print(X_train)

dt = DecisionTreeClassifier(max_depth = 15)

dt.fit(X_train, y_train)

y_train_pred = dt.predict(X_train)
print("Train Accuracy: ", accuracy_score(y_train, y_train_pred))

y_test_pred = dt.predict(X_test)
print("Test Accuracy: ", accuracy_score(y_test, y_test_pred))

from sklearn.feature_extraction.text import TfidfVectorizer    # tf-idf method
#Convert a collection of raw documents to a matrix of TF-IDF features

tfidf = TfidfVectorizer(ngram_range=(1, 1))
tfidf_feature = tfidf.fit_transform(processedReviews)

tfidf_feature.get_shape()

feature_names = tfidf.get_feature_names_out()
len(feature_names)

feature_names[:10]

X_train, X_test, y_train, y_test = train_test_split(tfidf_feature, labels, test_size=0.30, random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape

dt = DecisionTreeClassifier(max_depth = 15)
dt.fit(X_train, y_train)

y_train_pred = dt.predict(X_train)
print("Train Accuracy: ", accuracy_score(y_train, y_train_pred))

y_test_pred = dt.predict(X_test)
print("Test Accuracy: ", accuracy_score(y_test, y_test_pred))

from sklearn.linear_model import LogisticRegression

logit = LogisticRegression()
```

```
logit.fit(X_train, y_train)

y_train_pred_logit = logit.predict(X_train)
print("Training Accuracy :", accuracy_score(y_train, y_train_pred_logit))

y_test_pred_logit = logit.predict(X_test)
print("Testing Accuracy :", accuracy_score(y_test, y_test_pred_logit))
```