Importing dependencies

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import re,string,unicodedata
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,f1_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
import re
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

Loading data

```
df=pd.read_csv('/content/drive/MyDrive/Sentiment Analysis/Data/IMDB-Dataset.csv', encoding='latin-1')
```

Data Cleaning and Preprocessing

```
#Customize stopword as per data
nltk.download('stopwords')
stop_words = stopwords.words('english')
new_stopwords = ["would","shall","could","might"]
stop_words.extend(new_stopwords)
stop_words.remove("not")
stop_words=set(stop_words)
print(stop_words)
```

```
{'needn', 'from', 'after', 'now', 'yourselves', "needn't", 'be', 'as', 'again', "should've", 'during', 'been', 'up', 'to', 'will',
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#Removing special character
def remove_special_character(content):
    return re.sub('\W+',' ', content )#re.sub('\[[^&@#!]]*\]', '', content)

# Removing URL's
def remove_url(content):
    return re.sub(r'http\S+', '', content)

#Removing the stopwords from text
def remove_stopwords(content):
    clean_data = []
    for i in content.split():
        if i.strip().lower() not in stop_words and i.strip().lower().isalpha():
            clean_data.append(i.strip().lower())
    return " ".join(clean_data)

# Expansion of english contractions
def contraction_expansion(content):
    content = re.sub(r"won\'t", "would not", content)
    content = re.sub(r"can\'t", "can not", content)
    content = re.sub(r"don\'t", "do not", content)
    content = re.sub(r"shouldn\'t", "should not", content)
    content = re.sub(r"needn\'t", "need not", content)
    content = re.sub(r"hasn\'t", "has not", content)
    content = re.sub(r"haven\'t", "have not", content)
    content = re.sub(r"weren\'t", "were not", content)
    content = re.sub(r"mightn\'t", "might not", content)
    content = re.sub(r"didn\'t", "did not", content)
    content = re.sub(r"n\'t", " not", content)
    '''content = re.sub(r"\'re", " are", content)
    content = re.sub(r"\'s", " is", content)
    content = re.sub(r"\'d", " would", content)
    content = re.sub(r"\'ll", " will", content)
    content = re.sub(r"\'t", " not", content)
    content = re.sub(r"\'ve", " have", content)
    content = re.sub(r"\'m", " am", content)'''
    return content

#Data preprocessing
def data_cleaning(content):
    content = contraction_expansion(content)
    content = remove_special_character(content)
    content = remove_url(content)

    content = remove_stopwords(content)
    return content
```

```
pd.options.display.max_colwidth = 1000
#Data cleaning
df['Reviews_clean']=df['Reviews'].apply(data_cleaning)
df.head(5)
```

| | | | | | |
|---|---|---|---|---|---|
| **2** | 1 | Really, I could write a scathing review of this turd sandwich, but instead, I'm just going to be making a few observations and points I've deduced.There's just no point in watching these movies anymore. Does any reader out there remember Scary Movie? Remember how it was original with a few comedic elements to it? There was slapstick, some funny lines, it was a pretty forgettable comedy, but it was worth the price of admission. Well, That was the last time this premise was funny. STOP MAKING THESE MOVIES. PLEASE.I could call for a boycott of these pieces of monkey sh*t, but we all know there's going to be a line up of pre pubescent annoying little buggers, spouting crappy one liners like, "THIS IS SPARTA!" and, "IM RICK JAMES BITCH" so these movies will continue to make some form of monetary gain, considering the production value of this movie looks like it cost about 10 cents to make.Don't see this movie. Don't spend any money on it. Go home, rent Airplane, laugh your ass off, and ... | Disaster Movie | Realmente, eu poderia escrever uma crÃtica contundente sobre esse sanduÃche de cocÃ´, mas, em vez disso, vou fazer algumas observaÃ§Ãµes e pontos que deduzi. NÃ£o hÃ¡ mais sentido assistir a esses filmes. Algum leitor por aÃ se lembra do filme de terror? Lembra como era original, com alguns elementos cÃ´micos? Havia palhaÃ§ada, algumas frases engraÃ§adas, era uma comÃ©dia bastante esquecÃvel, mas valia o preÃ§o da entrada. Bem, essa foi a Ãºltima vez que essa premissa foi engraÃ§ada. PARE DE FAZER ESTES FILMES. POR FAVOR, eu poderia pedir um boicote a esses pedaÃ§os de macaco, mas todos sabemos que haverÃ¡ uma fila de buggers irritantes e prÃ©-pubescentes, jorrando uns forros ruins como: "ISTO Ã    SPARTA!" e "IM RICK JAMES BITCH", para que esses filmes continuem gerando algum ganho monetÃ¡rio, considerando que o valor de produÃ§Ã£o deste filme parece custar cerca de 10 centavos de dÃ³lar. NÃ£o gaste dinheiro com isso. VÃ¡ para casa, alugue a Airplane, ria e julgue silenciosament... | really write scathing review turd sandwich instead going making observations points deduced point watching movies anymore reader remember scary movie remember original comedic elements slapstick funny lines pretty forgettable comedy worth price admission well last time premise funny stop making movies please call boycott pieces monkey sh know going line pre pubescent annoying little buggers spouting crappy one liners like sparta im rick james bitch movies continue make form monetary gain considering production value movie looks like cost cents make not see movie not spend money go home rent airplane laugh ass silently judge people talking movie monday favor |
| | | If you saw the other previous spoof movies by these two horrible gentlemen, then you | | Se vocÃª viu os outros filmes falsificados anteriores por esses dois senhores horrÃ-veis, deve saber que isso jÃ¡ serÃ¡ ruim. Vou lhe dizer a | saw previous spoof movies two horrible gentlemen know already bad tell truth want watch brainless person ironically meant stereotypical teenagers not laugh bit judge even little |

Feature Engineering

```
#Mapping rating data to Binary label 1 (+ve) if rating >=7 and 0 (-ve) if rating <=4 and 2 (neutral) if ra
df['Label'] = df['Ratings'].apply(lambda x: '1' if x >= 7 else ('0' if x<=4 else '2'))
#Removing
df=df[df.Label<'2']
data=df[['Reviews_clean','Reviews','Ratings','Label']]
print(data['Label'].value_counts())
```

```
0    60000
1    60000
Name: Label, dtype: int64
```

```
#Importing dependencies for feature engineering
import sys
import os
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from prettytable import PrettyTable
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
```

Lemmatization

```
# lemmatization of word
class LemmaTokenizer(object):
    def __init__(self):
        self.wordnetlemma = WordNetLemmatizer()
    def __call__(self, reviews):
        return [self.wordnetlemma.lemmatize(word) for word in word_tokenize(reviews)]
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
train,test=train_test_split(data,test_size=.3,random_state=42, shuffle=True)
tfidfvect = TfidfVectorizer(analyzer = "word", tokenizer = LemmaTokenizer(), ngram_range=(1,3),min_df=10,m
x_train_tfidf = tfidfvect.fit_transform(train['Reviews_clean']).toarray()
x_test_tfidf = tfidfvect.transform(test['Reviews_clean']).toarray()

y_train = train['Label']
y_test = test['Label']
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

Model Evaluation

```
# Import prerequisite libraries
import sys
import numpy as np
import scipy as sp
import sklearn as sk
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, roc_auc_score, precision_score, recall_score, accuracy_score,precisi
from sklearn.pipeline import make_pipeline, Pipeline
```

Logistic Regression Model

```
from sklearn.pipeline import make_pipeline
model_1=LogisticRegression(penalty='l2',dual=False, tol=0.0001, C=10, solver='lbfgs', max_iter=200, multi_
model_2=Pipeline(
    steps=[
        #best base model("classifier", LogisticRegression(penalty='l2',dual=False, tol=0.0001, C=1.0, solv
    ('vect',TfidfVectorizer(analyzer = "word", tokenizer = LemmaTokenizer(), ngram_range=(1,3),min_df=10,m
)
```

Training of Logistic Regression Model

```
model_1.fit(x_train_tfidf,y_train)
```

```
▾          LogisticRegression
LogisticRegression(C=10, max_iter=200)
```

```
model_2.fit(train['Reviews_clean'],y_train)
```

```
▸        Pipeline
   ▸ TfidfVectorizer
 ▸ LogisticRegression
```

Evaluation on multiple metrics dataset

```
%%time
print("Precision Score for Logistic Regression: %s" % precision_score(y_test,model_1.predict(x_test_tfidf)
print("Recall Score for Logistic Regression: %s" % recall_score(y_test,model_1.predict(x_test_tfidf),avera
print("AUC Score for Logistic Regression: %s" % roc_auc_score(y_test,model_1.predict_proba(x_test_tfidf)[:
```

```
f1_score_1 =f1_score(y_test,model_1.predict(x_test_tfidf),average="weighted")
print("F1 Score for Logistic Regression: %s" % f1_score_1)
print("Accuracy Score for Logistic Regression: %s" % accuracy_score(y_test,model_1.predict(x_test_tfidf)))
print("Precision Score for Logistic Regression Pipeline: %s" % precision_score(y_test,model_2.predict(test
print("Recall Score for Logistic Regression Pipeline: %s" % recall_score(y_test,model_2.predict(test['Revi
print("AUC Score for Logistic Regression Pipeline: %s" % roc_auc_score(y_test,model_2.predict_proba(test['
f1_score_2 =f1_score(y_test,model_2.predict(test['Reviews_clean']),average="weighted")
print("F1 Score for Logistic Regression Pipeline: %s" % f1_score_2)
print("Accuracy Score for Logistic Regression Pipeline: %s" % accuracy_score(y_test,model_2.predict(test['
```

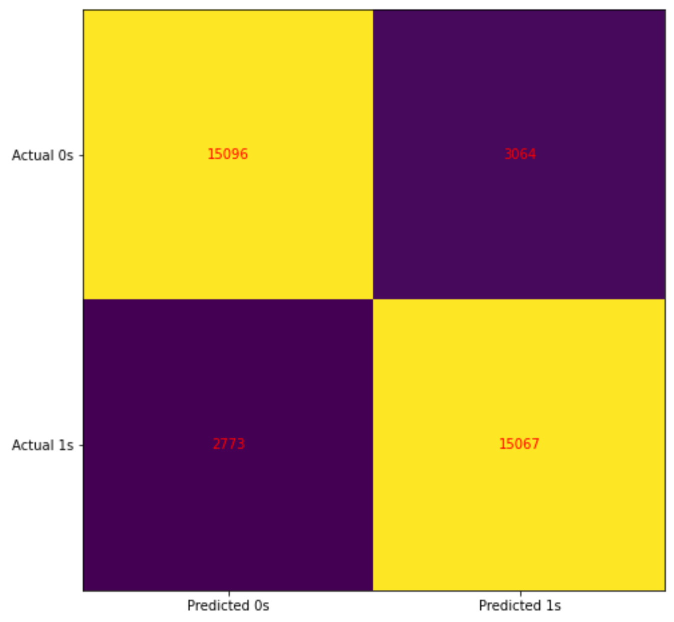
```
Precision Score for Logistic Regression: 0.8378611111111111
Recall Score for Logistic Regression: 0.8378611111111111
AUC Score for Logistic Regression: 0.9173820122824519
F1 Score for Logistic Regression: 0.8378621668911815
Accuracy Score for Logistic Regression: 0.8378611111111111
Precision Score for Logistic Regression Pipeline: 0.8377777777777777
Recall Score for Logistic Regression Pipeline: 0.8377777777777777
AUC Score for Logistic Regression Pipeline: 0.9173821450089883
F1 Score for Logistic Regression Pipeline: 0.8377790575321439
Accuracy Score for Logistic Regression Pipeline: 0.8377777777777777
CPU times: user 4min 25s, sys: 1.19 s, total: 4min 26s
Wall time: 4min 37s
```

```
y_predict=model_1.predict(x_test_tfidf)
y_predict_prob=model_1.predict_proba(x_test_tfidf)[:,1]
y_test_list=y_test.tolist()
y_predict_list=y_predict.tolist()
test_list=test['Reviews_clean'].tolist()
rating_list=test['Ratings'].tolist()
```

Confusion metrics

```
def confusion_matrix_plot(y_test,y_score):
    confmatrix = confusion_matrix(y_test,y_score)
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.imshow(confmatrix)
    ax.grid(False)
    ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
    ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
    ax.set_ylim(1.5, -0.5)
    for i in range(2):
        for j in range(2):
            ax.text(j, i, confmatrix[i, j], ha='center', va='center', color='red')
    plt.show()
```

```
confusion_matrix_plot(y_test,y_predict)
```



Analyzing False Positive and False Negative

```
pip install colorama
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: colorama
Successfully installed colorama-0.4.6
```

```
from colorama import Fore, Back, Style
fn_dict={}
fp_dict={}
for i in range(0, len(y_test_list)):
    if ((y_test_list[i]=='0') & (y_predict_list[i]=='1')):
        fp_dict[i]=[test_list[i],rating_list[i]]
    elif((y_test_list[i]=='1') & (y_predict_list[i]=='0')):
        fn_dict[i]=[test_list[i],rating_list[i]]
    else:
        pass
    i+=1
for k,v in fp_dict.items():
    if v[1]<=2:
        print(Fore.RED +'False Positive: %s %s'%(k,v))
for k,v in fn_dict.items():
    if v[1]>=9:
        print(Fore.GREEN +'False Negative: %s %s'%(k,v))
```

```
False Negative: 35938 ['episode depicts evil rot modern society gaming amazingly accurate portrayal episode combines finest poin
False Negative: 35943 ['peeps giving bad reviews film jumps gore storyline', 10]
False Negative: 35950 ['believe perfect movie many aspects factor distracted mind made scenario fictitious point view towards cr
```

Explainable AI by Shap

```
pip install shap
```

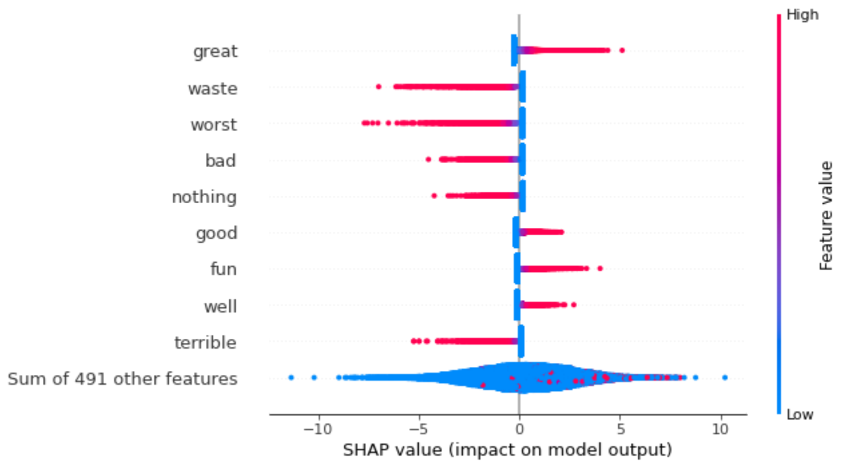
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting shap
  Downloading shap-0.41.0-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (572 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 572.4/572.4 KB 8.8 MB/s eta 0:00:00
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.9/dist-packages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.9/dist-packages (from shap) (23.0)
Requirement already satisfied: numba in /usr/local/lib/python3.9/dist-packages (from shap) (0.56.4)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.9/dist-packages (from shap) (2.2.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from shap) (1.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from shap) (1.22.4)
Collecting slicer==0.0.7
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from shap) (1.4.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from numba->shap) (67.6.0)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.9/dist-packages (from numba->shap) (0.39.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->shap) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn->shap) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn->shap) (1.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas->shap) (1.16
Installing collected packages: slicer, shap
Successfully installed shap-0.41.0 slicer-0.0.7
```

```
import shap

shap.initjs()
```

```
explainer = shap.Explainer(model_1, x_train_tfidf, feature_names=tfidfvect.get_feature_names_out())
shap_values = explainer(x_test_tfidf)
```

```
shap.plots.beeswarm(shap_values)
```



```
shap.initjs()
ind = 4443
print('Probability Score %s' %y_predict_prob[ind])
shap.plots.force(shap_values[ind])
```

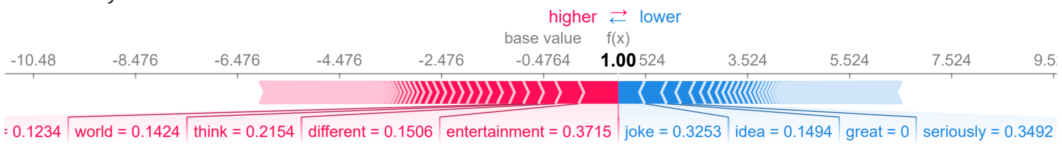
```
y_test_list=y_test.tolist()
print("Positive" if y_test_list[ind] else "Negative", "Review:")
print(test_list[ind])
```

```
Positive Review:
sorry not find movie funny many scenes believe cause offense others justified basis rented dvd skip lot content view see overall vo
```

Visualizing Marginal Contribution of Features for False Positive

```
shap.initjs()
ind = 111
print('Probability Score %s' %y_predict_prob[ind])
shap.plots.force(shap_values[ind])
```

```
Probability Score 0.7316140214275122
```



```
y_test_list=y_test.tolist()
print("Positive" if y_test_list[ind] else "Negative", "Review:")
print(test_list[ind])
```

```
Positive Review:
movie joke seriously anyone right mind think intended applied real life fictional joke something used part religion movie taken con
```

✓ 0s completed at 12:45 PM