SENTIMENT ANALYSIS ON MOVIE REVIEWS

SENTIMENT ANALYSIS:

Sentiment analysis is the process of analyzing digital text to determine if the emotional tone of the message is positive, negative, or neutral.

MAIN STEPS

- Dataset Loading
- Data Preprocessing
- Model Building and Training
- Evaluation

Dataset Loading

Dataset loading includes extraction of dataset, importing dependencies, and loading the dataset into collab.

Extracting Dataset

```
from zipfile import ZipFile
movie_reviews='/content/drive/MyDrive/Dataset/Movie_reviews.zip'

with ZipFile(movie_reviews,'r') as zip:
  zip.extractall()
  print("The dataset is extracted.")
```

```
    The dataset is extracted.
```

Importing Dependencies

```
import numpy as np
import pandas as pd
```

Loading dataset

```
Reviews=pd.read_csv("/content/ MR_Dataset.csv")
print(Reviews)
```

```
                                                  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

[50000 rows x 2 columns]
```

```
Reviews.shape
```

```
(50000, 2)
```

```
#naming the columns and reading the dataset again
column_names=['review','sentiment']
Reviews=pd.read_csv("/content/ MR_Dataset.csv",names=column_names,encoding='ISO-8859-1')
Reviews.head()
```

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

Data Preprocessing

Data preprocessing/preparation/cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset, or and refers to identifying incorrect, incomplete, irrelevant parts of the data and then modifying, replacing, or deleting the dirty data.

Steps Involved

- Tokenization
- Lower Casing
- Removing special Characters
- Removing Stopwords
- Stemming/Lemmatization
- Setting Target values

Tokenization

The process of converting a sequence of text into smaller parts, known as tokens.

```
#import libraries
import nltk
from nltk import word_tokenize
from nltk.corpus import words
nltk.download('words')
nltk.download('punkt')
```

```
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
def Tokenizing_reviews(review):
  tokenized_review=word_tokenize(review)
  return " ".join(tokenized_review)
```

Sample input And Sample output

```
print(Tokenizing_reviews(Reviews['review'][1]))
print(Tokenizing_reviews(Reviews['review'][200]))
print(Tokenizing_reviews(Reviews['review'][4550]))
```

```
A wonderful little production . < br / > < br / > The filming technique is very unass
Interesting and short television movie describes some of the machinations surrounding
Rated E < br / > < br / > I never actually owned a Nintendo 64 but I have played one
```

```
Reviews['review']=Reviews['review'].apply(Tokenizing_reviews)
print(Reviews['review'])
```

```
0        One of the other reviewers has mentioned that ...
1        A wonderful little production . < br / > < br ...
2        I thought this was a wonderful way to spend ti...
3        Basically there 's a family where a little boy...
4        Petter Mattei 's `` Love in the Time of Money ...
                              ...
49995    I thought this movie did a down right good job...
49996    Bad plot , bad dialogue , bad acting , idiotic...
49997    I am a Catholic taught in parochial elementary...
49998    I 'm going to have to disagree with the previo...
49999    No one expects the Star Trek movies to be high...
Name: review, Length: 50000, dtype: object
```

Lower Casing

Converting all text to lowercase

Double-click (or enter) to edit

```
import string
def Lower_Casing(review):
  lower_cased_review= review.lower()
  return lower_cased_review
```

Sample input and Sample output

```
print(Lower_Casing(Reviews['review'][10]))
print(Lower_Casing(Reviews['review'][573]))
print(Lower_Casing(Reviews['review'][4595]))
```

```
phil the alien is one of those quirky films where the humour is based around the oddn
1 ) i am not weapon expert , but even i can see difference between u.s. army riffles
it 's very simple to qualify that movie : `` a pure masterpiece '' . this opinion is
```

```
Reviews['review']=Reviews['review'].apply(Lower_Casing)
print(Reviews['review'])
```

```
0        one of the other reviewers has mentioned that ...
1        a wonderful little production . < br / > < br ...
2        i thought this was a wonderful way to spend ti...
3        basically there 's a family where a little boy...
4        petter mattei 's `` love in the time of money ...
                              ...
49995    i thought this movie did a down right good job...
49996    bad plot , bad dialogue , bad acting , idiotic...
49997    i am a catholic taught in parochial elementary...
49998    i 'm going to have to disagree with the previo...
49999    no one expects the star trek movies to be high...
Name: review, Length: 50000, dtype: object
```

Removing Special Characters

Eliminating non_alphanumeric characters or symbols can enhance the clarity of the text.

```
import re
valid_words = set(words.words())
def Removing_spe_char(review):
  clear_review=re.sub('[^a-zA-Z]',' ',review)
  clear_words=word_tokenize(clear_review)
  clear_words=[word for word in clear_words if word in valid_words]
  return " ".join(clear_words)
```

Sample input And Sample output

```

```

```
print(Removing_spe_char(Reviews['review'][11]))
print(Removing_spe_char(Reviews['review'][2]))
print(Removing_spe_char(Reviews['review'][3333]))
```

```
arly in the film i sat up and took notice at that point since are making up the story
e but spirited young woman this may not be the crown jewel of his career but it was th
```

```
Reviews['review']=Reviews['review'].apply(Removing_spe_char)
print(Reviews['review'])
```

```
0        one of the other that after watching just epis...
1        a wonderful little production the technique is...
2        i thought this was a wonderful way to spend ti...
3        basically there s a family where a little boy ...
4        petter s love in the time of money is a visual...
                              ...
49995    i thought this movie did a down right good job...
49996    bad plot bad dialogue bad acting idiotic the a...
49997    i am a catholic taught in parochial elementary...
49998    i m going to have to disagree with the previou...
49999    no one the star trek to be high art but the do...
Name: review, Length: 50000, dtype: object
```

Removing Stopwords

stop words are used to eliminate unimportant words, allowing applications to focus on the important words instead.

```
#importing stopwords
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk import TweetTokenizer
tokenizer=TweetTokenizer()
stop_words=set(stopwords.words('english'))
def Remove_stopwords(review):
  words=tokenizer.tokenize(review)
  important_words=[word for word in  words if word not in stop_words]
  return " ".join(important_words)
```

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

Sample input And Sample output

```
print(Remove_stopwords(Reviews['review'][1]))
print(Remove_stopwords(Reviews['review'][222]))
print(Remove_stopwords(Reviews['review'][3333]))
```

```
wonderful little production technique unassuming old time fashion comforting sometime
high school small city finally getting married much delight town abuzz one nominated
```

```
film one fall love shall always remain top time influential sheer simplicity gripping
```

```
Reviews['review']=Reviews['review'].apply(Remove_stopwords)
print(Reviews['review'])
```

```
0        one watching episode hooked right exactly firs...
1        wonderful little production technique unassumi...
2        thought wonderful way spend time hot summer we...
3        basically family little boy jake zombie closet...
4        petter love time money visually stunning film ...
                              ...
49995    thought movie right good job n creative origin...
49996    bad plot bad dialogue bad acting idiotic annoy...
49997    catholic taught parochial elementary taught hi...
49998    going disagree previous comment side one secon...
49999    one star trek high art expect movie good best ...
Name: review, Length: 50000, dtype: object
```

Stemming

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or the roots.

Lemmatization

Lemmatization is the process of grouping together different inflected forms of the same word. It's used in computational linguistics, natural language processing (NLP) and chatbots.

```
# Stemming
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
def Stemming(review):
  words = review.split()
  stemmed_words = [stemmer.stem(word) for word in words]
  return " ".join(stemmed_words)
```

Sample input And Sample output

```
print(Stemming(Reviews['review'][1]))
print(Stemming(Reviews['review'][222]))
print(Stemming(Reviews['review'][333]))
```

```
wonder littl product techniqu unassum old time fashion comfort sometim discomfort sen
high school small citi final get marri much delight town abuzz one nomin act everyon
often laugh loud funni play sex famili class zip code seen sinc day granni jed plot t
```

```
stemmed_reviews = Reviews['review'].apply(Stemming)
print(stemmed_reviews)
```

```
0        one watch episod hook right exactli first thin...
1        wonder littl product techniqu unassum old time...
2        thought wonder way spend time hot summer weeke...
3        basic famili littl boy jake zombi closet fight...
4        petter love time money visual stun film watch ...
                             ...
49995    thought movi right good job n creativ origin f...
49996    bad plot bad dialogu bad act idiot annoy groov...
49997    cathol taught parochi elementari taught high s...
49998    go disagre previou comment side one second rat...
49999    one star trek high art expect movi good best u...
Name: review, Length: 50000, dtype: object
```

Setting Target values

- Converting Positive review value to --> 1
- Converting Negitive review value to --> 0

```
#Before Setting Values
Reviews['sentiment'].value_counts()
```

```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

```
#After Setting Values
Reviews.replace({'sentiment':{'positive':1, 'negative':0}}, inplace=True)
Reviews['sentiment'].value_counts()
```

```
1    25000
0    25000
Name: sentiment, dtype: int64
```

Model Building and Training

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

Steps

- Separating data and label
- Spliting into training and testing data
- Converting into Numerical data
- Training the Model

```

```

```
#importing Dependencies
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Separating data and label

seperating the data and lable includes storing reviews and corresponding sentement in 2 different variables X & Y

X <-- reviews

```
print(Reviews['review'])
```

```
0        one watching episode hooked right exactly firs...
1        wonderful little production technique unassumi...
2        thought wonderful way spend time hot summer we...
3        basically family little boy jake zombie closet...
4        petter love time money visually stunning film ...
                              ...
49995    thought movie right good job n creative origin...
49996    bad plot bad dialogue bad acting idiotic annoy...
49997    catholic taught parochial elementary taught hi...
49998    going disagree previous comment side one secon...
49999    one star trek high art expect movie good best ...
Name: review, Length: 50000, dtype: object
```

```
X = Reviews['review'].values #Reviews as X
print(X)
```

```
['one watching episode hooked right exactly first thing struck brutality unflinching
 'wonderful little production technique unassuming old time fashion comforting someti
 'thought wonderful way spend time hot summer weekend sitting air conditioned theater
 ...
 'catholic taught parochial elementary taught high school college still catholic woul
 'going disagree previous comment side one second rate excessively vicious western tr
 'one star trek high art expect movie good best unfortunately movie implausible plot
```

Y <-- sentiment

```
print(Reviews['sentiment'])
```

```
0        1
1        1
2        1
3        0
4        1
        ..
49995    1
49996    0
```

```
49997    0
49998    0
49999    0
Name: sentiment, Length: 50000, dtype: int64
```

```
Y = Reviews['sentiment'].values #Sentiments as Y
print(Y)
```

```
[1 1 1 ... 0 0 0]
```

Spliting into training and testing data

Splitting the data into training set and testing set to train and test model

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=
```

```
print("X -->",X.shape,X_train.shape,X_test.shape)
print("Y -->",Y.shape,Y_train.shape,Y_test.shape)
```

```
X --> (50000,) (40000,) (10000,)
Y --> (50000,) (40000,) (10000,)
```

Converting into Numerical data

The process of converting textual data into numerical data is called vectorization.

Here we convert reviews into numerical values using vectorization.

```
vectorizer=TfidfVectorizer()
X_train=vectorizer.fit_transform(X_train)
X_test=vectorizer.transform(X_test)
print("Vectorization Completed")
```

```
Vectorization Completed
```

```
print("Vectorized Training set")
print(X_train)
```

```
Vectorized Training set
  (0, 1675)     0.04192347497135277
  (0, 16769)    0.0380612091559975
  (0, 11357)    0.05344278894898061
  (0, 12118)    0.06053394909166146
  (0, 21846)    0.06269226662791344
  (0, 9296)     0.05835001448081001
  (0, 8257)     0.03956205680987219
  (0, 26153)    0.08257977152394415
  (0, 12359)    0.05335907245491889
  (0, 10593)    0.06751998522419207
  (0, 10384)    0.07709075295948802
  (0, 9025)     0.05653523466765315
  (0, 14916)    0.0847937739256378
```

```
  (0, 459)      0.07890043806278875
  (0, 12573)    0.13404312782513414
  (0, 24147)    0.13314188935520108
  (0, 11166)    0.08273562180279838
  (0, 11316)    0.09984243026305199
  (0, 8655)     0.04409613601538111
  (0, 20684)    0.06514456569085394
  (0, 9956)     0.06709706182881392
  (0, 6823)     0.05743162991600034
  (0, 20405)    0.11894533900340538
  (0, 666)      0.04723490969760232
  (0, 1626)     0.04874281788327301
  :     :
  (39999, 2813) 0.21124812699621237
  (39999, 25691)        0.07092437850418785
  (39999, 11210)        0.11247286654184543
  (39999, 28619)        0.08033977719040394
  (39999, 5509) 0.06460833373336043
  (39999, 3474) 0.10670789220186185
  (39999, 25753)        0.12682108928517688
  (39999, 1708) 0.06834751379021131
  (39999, 28615)        0.09846218226384074
  (39999, 22377)        0.12025910702318465
  (39999, 11954)        0.1058853907012102
  (39999, 25877)        0.08952589266758329
  (39999, 16360)        0.08326457732949426
  (39999, 17949)        0.08582587322177661
  (39999, 24258)        0.0769452775484964
  (39999, 16769)        0.0728720645899802
  (39999, 10593)        0.06463694708443746
  (39999, 8660) 0.09698829917694086
  (39999, 11285)        0.09437602261369016
  (39999, 14628)        0.0494013564656557
  (39999, 22402)        0.09733062636090178
  (39999, 23573)        0.11111853883049272
  (39999, 4704) 0.09391139797322359
  (39999, 8639) 0.1753466899170269
  (39999, 9437) 0.17737748604487516
```

```
print("Vectorized Testing set")
print(X_test)
```

```
Vectorized Testing set
  (0, 29182)    0.10546513636575149
  (0, 29031)    0.1430322214580646
  (0, 26458)    0.21723100226264083
  (0, 26043)    0.12262255788413343
  (0, 25366)    0.2076687672875543
  (0, 24956)    0.273141061531683
  (0, 24500)    0.1750291568207712
  (0, 23536)    0.1388617148152434
  (0, 22373)    0.2357298408817069
  (0, 22208)    0.14026685781394765
  (0, 21951)    0.13058960053749064
  (0, 19378)    0.31395720523520393
  (0, 19080)    0.16587178209012185
  (0, 18775)    0.14298619067236834
  (0, 17278)    0.05168849650104275
  (0, 16967)    0.15447668528933625
```

```
  (0, 16360)    0.09863691853056637
  (0, 16246)    0.22977317174880083
  (0, 15517)    0.10731165124685284
  (0, 14918)    0.1317640522738754
  (0, 14762)    0.13033766907435002
  (0, 14628)    0.05852185562318697
  (0, 13763)    0.11940599674587125
  (0, 11828)    0.11875881005340135
  (0, 11457)    0.14655280493035994
  :     :
(9999, 12359) 0.09161352691374605
(9999, 11716) 0.09378311876781464
(9999, 11217) 0.19033487724448736
(9999, 11050) 0.2691051402534418
(9999, 10852) 0.04922226912353863
(9999, 10593) 0.057963376226023815
(9999, 10006) 0.1174613034966068
(9999, 9462)  0.07494822867756634
(9999, 9115)  0.09215684457018063
(9999, 8650)  0.06622237836432096
(9999, 8455)  0.10645937238368454
(9999, 8379)  0.09803197887727402
(9999, 7972)  0.09033874174625225
(9999, 7469)  0.176201939044844
(9999, 5230)  0.14301060237852958
(9999, 3703)  0.07690980360596984
(9999, 3528)  0.14543011508853815
(9999, 3365)  0.11387941491603144
(9999, 3146)  0.1922603497437669
(9999, 2262)  0.06712816493481649
(9999, 1021)  0.20857301303606696
(9999, 1018)  0.08412883815444862
(9999, 793)   0.12378487035952845
(9999, 338)   0.2498682537996516
(9999, 240)   0.0636104220968769
```

Training the Model

Model training in machine language is the process of feeding an ML algorithm with data

```
#Logistic Regression
NLP_Model=LogisticRegression(max_iter=1000)
NLP_Model.fit(X_train,Y_train)
```

```
▾       LogisticRegression
LogisticRegression(max_iter=1000)
```

Model Evaluation

Model evaluation aims to define how well the model performs its task.

- Checking Accuracy
- Saving the model

Checking Accuracy

We evaluate the model by checking the accuracy score of the model built.

```
#Accuracy Score for Training Data
X_train_prediction=NLP_Model.predict(X_train)
training_data_accuracy=accuracy_score(Y_train,X_train_prediction)
print('Accuracy Score:',training_data_accuracy)
```

```
Accuracy Score: 0.920125
```

```
#Accuracy Score for Testing Data
X_test_prediction=NLP_Model.predict(X_test)
test_data_accuracy=accuracy_score(Y_test,X_test_prediction)
print('Accuracy Score:',test_data_accuracy)
```

```
Accuracy Score: 0.882
```

Saving The Model

Saving the model that has been trained and tested in a file.

```
import pickle
filename="My_NLP_Model.sav"
pickle.dump(NLP_Model,open(filename,'wb'))
print("Model successfully Saved")
```

```
Model successfully Saved
```

Testing Model

```
# using the saved model for future predictions
NLP=pickle.load(open('/content/My_NLP_Model.sav','rb'))
```

```
X_new=X_test[2000]
print("Original:",Y_test[2000])
prediction=NLP_Model.predict(X_new)
print("Predicted:",prediction)

if(prediction[0]==0):
  print("Negative Review")
else:
  print("Positive Review")
```

```
Original: 0
Predicted: [0]
Negative Review
```

```
X_new=X_test[573]
print("Original:",Y_test[573])
prediction=NLP_Model.predict(X_new)
print("Predicted:",prediction)

if(prediction[0]==0):
  print("Negative Review")
else:
  print("Positive Review")
```

```
Original: 1
Predicted: [1]
Positive Review
```

```
prediction=NLP_Model.predict(X_test)
print("Original:",Y_test)
print("Predicted:",prediction)
Correct_prediction=0
for i in range(len(prediction)):
  if prediction[i]==Y_test[i]:
    Correct_prediction+=1
print("Correctly predicted percentage: ",Correct_prediction*100/len(prediction))
```

```
Original: [1 0 1 ... 0 1 1]
Predicted: [1 1 1 ... 0 1 1]
Correctly predicted percentage:  88.2 8820
```