RESTAURANT REVIEW ANALYSIS USING ML & NLP METHODS

SOFTWARE DESIGN AND DEVELOPMENT PROJECT

GUIDED BY - PROF. MARIA ANU (52310)

SUBMITTED BY - HEMA CHANDRIKA S (19MIS1199)

IMPLEMENTATION OF ML MODELS

IMPORTING THE DEPENDENCIES

```
import os
import re
import nltk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
```

READING THE CSV FILE

```
dataset = pd.read_csv('/content/Restaurant_Reviews.tsv', delimiter = '\t', quoting = 3)
```

DATA EXPLORATION

DISPLAYING THE FIRST FIVE CONTENTS OF THE FILE

```
dataset.head()
```

	Review	Liked
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1

DISPLAYING THE LAST FIVE CONTENTS OF THE FILE

```
dataset.tail()
```

	Review	Liked
995	I think food should have flavor and texture an...	0
996	Appetite instantly gone.	0
997	Overall I was not impressed and would not go b...	0
998	The whole experience was underwhelming, and I ...	0
999	Then, as if I hadn't wasted enough of my life ...	0

SUMMARY OF THE DATASET

```
dataset.describe()
```

	Liked
count	1000.00000
mean	0.50000
std	0.50025
min	0.00000
25%	0.00000
50%	0.50000
75%	1.00000
max	1.00000

PAIRWISE CORRELATION

```
dataset.corr()
```

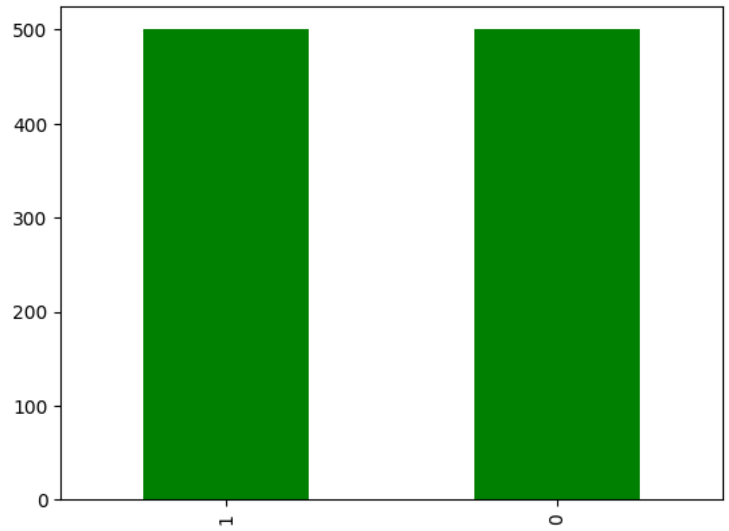
```
<ipython-input-6-c187c74d1e71>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version,
  dataset.corr()
```

	Liked
Liked	1.0

PLOTING THE RATIO OF POSITIVE AND NEGATIVE REVIEWS

```
dataset['Liked'].value_counts().plot.bar(color = 'green')
```

```
<Axes: >
```



TEXT MINING - REMOVAL OF STOPWORDS

```
nltk.download('stopwords')
corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    review = [ps.stem(word) for word in review if not word in set(all_stopwords)]
    review = ' '.join(review)
    corpus.append(review)
print(corpus)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
['wow love place', 'crust not good', 'not tasti textur nasti', 'stop late may bank holiday rick steve recommend love', 'select menu grea
```

CREATING BAG OF WORDS MODEL

```
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, -1].values
```

TRAIN TEST SPLIT

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(800, 1500)
(800,)
(200, 1500)
(200,)
```

MINIMUM AND MAXIMUM SCALING

```
from sklearn.preprocessing import MinMaxScaler
```

CREATING MINIMUM AND MAXIMUM SCALER

```
mm = MinMaxScaler()
```

FEEDING THE INDEPENDENT VARIABLES INTO THE MODEL

```
X_train = mm.fit_transform(X_train)
X_test = mm.transform(X_test)
```

1) NAIVE BAYES CLASSIFIER ON TRAIN SET

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
```

CREATING THE MODEL

```
model = GaussianNB()
```

FITTING THE TRAINING DATA TO THE MODEL

```
model.fit(X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

PREDICTING THE TEST SET RESULTS

```
y_pred = model.predict(X_test)
```

CHECKING THE ACCURACIES

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.92125
Testing Accuracy : 0.725
```

CONFUSION MATRIX

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[55 42]
 [13 90]]
```

2) RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
```

CREATING THE MODEL

```
model = RandomForestClassifier()
```

FITTING THE TRAINING DATA TO THE MODEL

```
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

PREDICTING THE TEST SET RESULTS

```
y_pred = model.predict(X_test)
```

CHECKING THE ACCURACIES

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.99625
Testing Accuracy : 0.765
```

CONFUSION MATRIX

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[88  9]
 [38 65]]
```

3) DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
```

CREATING THE MODEL

```
model = DecisionTreeClassifier()
```

FITTING THE TRAINING DATA TO THE MODEL

```
model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

PREDICTING THE TEST SET RESULTS

```
y_pred = model.predict(X_test)
```

CHECKING THE ACCURACIES

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.99625
Testing Accuracy : 0.73
```

CONFUSION MATRIX

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[73 24]
 [30 73]]
```

4) LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

CREATING THE MODEL

```
model = LogisticRegression()
```

FITTING THE TRAINING DATA TO THE MODEL

```
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

PREDICTING THE TEST SET RESULTS

```
y_pred = model.predict(X_test)
```

CHECKING THE ACCURACIES

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.96875
Testing Accuracy : 0.785
```

CONFUSION MATRIX

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[80 17]
 [26 77]]
```

5) SUPPORT VECTOR MACHINES

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

CREATING THE MODEL

```
model = SVC()
```

FITTING THE TRAINING DATA TO THE MODEL

```
model.fit(X_train, y_train)
```

```
SVC()
```

PREDICTING THE TEST SET RESULTS

```
y_pred = model.predict(X_test)
```

CHECKING THE ACCURACIES

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.98
Testing Accuracy : 0.775
```

CONFUSION MATRIX

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[87 10]
 [35 68]]
```

6) MULTI LAYER PERCEPTRON

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
```

CREATING THE MODEL

```
model = MLPClassifier()
```

FITTING THE TRAINING DATA TO THE MODEL

```
model.fit(X_train, y_train)
```

```
MLPClassifier()
```

PREDICTING THE TEST SET RESULTS

```
y_pred = model.predict(X_test)
```

CHECKING THE ACCURACIES

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.99375
Testing Accuracy : 0.77
```

CONFUSION MATRIX

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[76 21]
 [25 78]]
```

INFERENCE: LOGISTIC REGRESSION GIVES THE HIGHEST ACCURACY OF 78.5% OUT OF ALL MODELS

CREATING WORDCLOUD

```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from wordcloud) (1.23.5)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1
```

```
from wordcloud import WordCloud
from wordcloud import STOPWORDS
# Import matplotlib
import matplotlib.pyplot as plt

def word_cloud(text):

    # Create stopword list
    stopword_list = set(STOPWORDS)

    # Create WordCloud
    word_cloud = WordCloud(width = 550, height = 550,
                           background_color ='white',
                           stopwords = stopword_list,
                           min_font_size = 12).generate(text)

    # Set wordcloud figure size
    plt.figure(figsize = (8, 6))

    # Show image
    plt.imshow(word_cloud)

    # Remove Axis
    plt.axis("off")

    # show plot
    plt.show()

paragraph=' '.join(dataset.Review.tolist())
word_cloud(paragraph)
```


```
paragraph=' '.join(dataset[dataset.Liked==1].Review.tolist())
word_cloud(paragraph)
```


```
paragraph=' '.join(dataset[dataset.Liked==0].Review.tolist())
word_cloud(paragraph)
```


place
food
service
time
good
minutes
worst
got
really
much
go
will
back
ordered
definitely
will never
think
another
meal
us
pretty
probably
cold
slow
bland
server
experience
even
table
coming
go back
way
disappointed
know
better
took
salad
best
quality
say
terrible
waited
wait
burger
rude
came