Detailed Project Report (DPR)

This "Restaurant-Review-Sentiment-Analysis" project is going to give the exact meaning of the review. For this we use some python libraries, tools, dataset and Google colab.

I'll explain each and every command in the project code.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

These four lines get import the python libraries for our project.

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

This line mounts the dataset from our colab.

```
dataset.head()
```

This line prints the first five lines of the data which is available in dataset.

```

dataset.isna().sum()
```

isan is a pandas Data Frame method that returns a Data Frame of the same shape as the original Data Frame, where each element is either True if it's a missing or null value (NaN), or False if it's not.

```
dataset.shape
```

It gives the shape of the dataset.

```python
import seaborn as sns

sns.countplot(x='Liked', data=pd.DataFrame({'Liked': [1, 0, 1, 0]}))
```

This line prepares the model output for our “Restaurant-Review-Sentiment-Analysis” by using 0’s and **1’s**.

```python
import re
import nltk
nltk.download('stopwords')
```

import re: This imports the re module, which stands for "regular expressions. "Regular expressions are a powerful tool for pattern matching and manipulation of strings in Python. They allow you to search for specific patterns within text, extract information, or replace parts of the text based on defined patterns.

import nltk: This imports the nltk library, which stands for "Natural Language Toolkit." NLTK is a popular library for natural language processing (NLP) tasks in Python. It provides various tools and resources for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, parsing, and more.

nltk.download('stopwords'): This line downloads the stopwords dataset from NLTK. Stopwords are common words (such as "and", "the", "is", etc.) that are often filtered out from text data during preprocessing, as they typically do not carry significant meaning for many NLP tasks like text classification or sentiment analysis. NLTK provides a list of stopwords for various languages, and downloading them makes them accessible for your NLP tasks in Python.

```python
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

from nltk.corpus import stopwords: This line imports the stopwords module from the NLTK corpus submodule. NLTK provides a collection of corpora, lexical resources, and datasets for various natural language processing tasks. The stopwords module contains lists of common stopwords for different languages, which are often removed from text during preprocessing to improve the performance of NLP tasks.

from nltk.stem.porter import PorterStemmer: This line imports the PorterStemmer class from the porter module in the nltk.stem submodule. Stemming is a technique used in natural language processing and information retrieval to reduce words to their root or base form (known as the stem) by removing affixes. The Porter stemming algorithm, implemented in NLTK as PorterStemmer, is one of the most commonly used stemming algorithms.

```
corpus=[]
```

The line corpus=[] initializes an empty list named corpus in Python.

In natural language processing (NLP) tasks, a "corpus" typically refers to a collection of text documents used for analysis, processing, or training machine learning models.

```
dataset['Review'][0]
```

This command prints the first review of the dataset.

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

This line iterates through each word in the review list, applies stemming using the PorterStemmer, and filters out any words that are in the list of English stopwords. The list of stopwords is obtained from the NLTK stopwords module for the English language. The result is a list of processed words for the current review.

```
review
```

It prints the review.

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

It joints the review without spaces.

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

It gives the spaces for each and every word.

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

It means collect all reviews and give a proper structure to the review.

```
corpus
```

It prints all 1000 reviews.

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus).toarray()
y=dataset.iloc[: , -1].values
```

Running these lines of code, X will contain the numerical representation of the text data, and y will contain the target variable values from the dataset, suitable for use in machine learning models. Typically, X would be used as the input features, and y would be used as the target variable for training and evaluation.

```
X
```

It will print X array.

```
y
```

It will print y array.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random
_state=0)
```

By using the above commands we'll have separate arrays (X_train, X_test, y_train, y_test) containing the training and testing data, which you can then use to train and evaluate machine learning models.Typically, you would use the training data (X_train and y_train) to train the model and the testing data (X_test and y_test) to evaluate its performance.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

GaussianNB()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

It will predict the y reviewsand classify.

```
from sklearn.metrics import confusion_matrix , accuracy_score
cm = confusion_matrix(y_test , y_pred)
sns.heatmap(cm , annot=True)
```

It will print the confusion matrix.

```
cm
```

It will give the value of confusion matrix.

```
accuracy_score(y_test , y_pred)
```

It prints the model accuracy.

```
X_pred = np.array(['The food was delicious'])
X_pred = vectorizer.transform(X_pred).toarray()
classifier.predict(X_pred)
```

Now, we will test the model by giving some example reviews.