

▾ Sentiment Analysis of Restaurant Reviews

A natural language processing (NLP) approach called sentiment analysis of restaurant reviews is used to ascertain the general sentiment or emotional tone represented in customer evaluations of restaurants. To categorise text data from reviews into different sentiment categories, often comprising positive, negative, or neutral feelings, this procedure requires analysing the text data. Here is a basic explanation of how restaurant review sentiment analysis functions:

1. **Gathering Data:** The initial phase entails gathering a dataset of restaurant reviews. Typically, these evaluations come from a variety of internet sources, including Twitter, Facebook, TripAdvisor, Yelp, and Google evaluations.
2. **Text preprocessing:** Unneeded information, punctuation, and special characters are removed from raw text data during preprocessing. Tokenization is the process of dividing the text into tokens, which are single words or phrases.
3. **Sentiment Classification:** To categorise the sentiment of each review, machine learning or deep learning models are used. Pre-trained models like BERT or LSTM-based neural networks are examples of common methodologies. Based on the language and context used in the text, the model labels each review as either favourable, negative, or neutral.
4. **Extraction of characteristics:** Additional characteristics or information, such as the star rating, review length, or the presence of particular terms or phrases referring to the calibre of the cuisine, the level of service, the atmosphere, etc., may occasionally be retrieved from the reviews.
5. **Training and Validation:** A labelled dataset of reviews is used to train the sentiment analysis model, and each review is assigned a specific sentiment label. Utilising validation data, the model's performance is assessed to make sure it can correctly predict sentiments.
6. **Sentiment Visualisation:** To give a fast overview of the distribution of positive, negative, and neutral sentiments, the findings of sentiment analysis are frequently visualised using graphs or charts. The most commonly used positive and negative terms can also be highlighted using word clouds or emotion histograms.
7. **Sentiment Insights:** The outcomes of sentiment analysis give restaurant managers and owners useful information. They may make data-driven decisions to improve their services using this information to determine areas for improvement, assess client contentment, and evaluate consumer satisfaction.
8. **Sentiment Monitoring:** To track client comments in real-time, restaurants may use sentiment analysis as a continuous monitoring technique. This helps them to handle consumer problems and reply quickly to unfavourable feedback.
9. **Reporting and Feedback:** To give a clear picture of consumer attitudes over time, the sentiment analysis results are frequently included into routine reports or dashboards. Marketing initiatives, quality assurance, and corporate strategy may all benefit from this data.

In conclusion, restaurant review sentiment analysis employs NLP and machine learning approaches to assess customer comments and determine the sentiment reflected in reviews. It enables restaurants to maximise customer happiness by enhancing services, managing their online reputation, and making data-driven choices.

▾ Import Libraries

```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import nltk
import string
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

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

pip install catboost
```

```

Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.23.5)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.offline as pyo
import plotly.io as pio
import sklearn
pd.set_option('display.max_columns',None)

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from catboost import CatBoostClassifier
from xgboost import XGBClassifier

from wordcloud import WordCloud,STOPWORDS
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict, cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

import re
import nltk
from nltk.corpus import stopwords
from nltk import word_tokenize
nltk.download('stopwords')

import warnings
warnings.filterwarnings('ignore')

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



```

▼ Import Dataset


```
df = pd.read_csv('/content/Restaurant_Reviews (1) (1).tsv', delimiter = '\t', quoting = 3)
```

▼ Inspecting

```
df.head()
```

	Review	Liked	
0	Wow... Loved this place.	1	
1	Crust is not good.	0	

```
df.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	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    Review   1000 non-null    object
1    Liked    1000 non-null    int64
dtypes: int64(1), object(1)
memory usage: 15.8+ KB
```

```
df.isna().sum()
```

```
Review    0
Liked     0
dtype: int64
```

```
df = df.dropna()
```



```
df.isna().sum()
```

```
Review    0
Liked     0
dtype: int64
```

```
rows = df.shape[0]
cols = df.shape[1]
print("Rows   : " + str(rows))
print("Columns: " + str(cols))
```

```
Rows    : 1000
Columns: 2
```

```
df.describe(exclude=['O'])
```

	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	

```
df.describe(include=['O'])
```

	Review	
count	1000	
unique	996	
top	I would not recommend this place.	
freq	2	

```
df.nunique()
```

```
Review      996
Liked        2
dtype: int64
```

```
stopwords.words('english')
```

```
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
'don't',
'should',
'should've',
'now',
'd',
'll',
'm',
'o',
're',
've',
'y',
'ain',
'aren',
'aren't',
'couldn',
'couldn't',
'didn',
'didn't',
'doesn',
'doesn't',
'hadn',
'hadn't',
'hasn',
'hasn't',
'haven',
'haven't',
'isn',
'isn't',
'ma',
'mightn',
'mightn't',
'mustn',
'mustn't',
'needn',
'needn't',
'shan',
'shan't',
'shouldn',
'shouldn't',
'wasn',
'wasn't',
'weren',
'weren't',
'won',
'won't',
'wouldn',
'wouldn't']
```

```
def text_process(msg):
    nopunc =[char for char in msg if char not in string.punctuation]
    nopunc=''.join(nopunc)
    return ' '.join([word for word in nopunc.split() if word.lower() not in stopwords.words('english')])
```

```

corpus = []
for i in range(0,1000):
    review = re.sub(pattern = '[^a-zA-z]', repl = ' ', string=df['Review'][i])

    review = review.lower()
    review_words = review.split()
    review_words = [word for word in review_words if not word in set(stopwords.words('english'))]

    ps = PorterStemmer()

    review = [ps.stem(word) for word in review_words]
    review = ' '.join(review)

    corpus.append(review)
corpus[:1500]

```

```

[ 'wow love place',
  'crust good',
  'tasti textur nasti',
  'stop late may bank holiday rick steve recommend love',
  'select menu great price',
  'get angri want damn pho',
  'honeslti tast fresh',
  'potato like rubber could tell made ahead time kept warmer',
  'fri great',
  'great touch',
  'servic prompt',
  'would go back',
  'cashier care ever say still end wayyy overpr',
  'tri cape cod ravoli chicken cranberri mmmm',
  'disgust pretti sure human hair',
  'shock sign indic cash',
  'highli recommend',
  'waitress littl slow servic',
  'place worth time let alon vega',
  'like',
  'burritto blah',
  'food amaz',
  'servic also cute',
  'could care less interior beauti',
  'perform',
  'right red velvet cake ohhh stuff good',
  'never brought salad ask',
  'hole wall great mexican street taco friendli staff',
  'took hour get food tabl restaur food luke warm sever run around like total overwhelm',
  'worst salmon sashimi',
  'also combo like burger fri beer decent deal',
  'like final blow',
  'found place accid could happier',
  'seem like good quick place grab bite familiar pub food favor look elsewher',
  'overall like place lot',
  'redeem qualiti restaur inexpens',
  'ampl portion good price',
  'poor servic waiter made feel like stupid everi time came tabl',
  'first visit hiro delight',
  'servic suck',
  'shrimp tender moist',
  'deal good enough would drag establish',
  'hard judg whether side good gross melt styrofoam want eat fear get sick',
  'posit note server attent provid great servic',
  'frozen puck disgust worst peopl behind regist',
  'thing like prime rib dessert section',
  'bad food damn gener',
  'burger good beef cook right',
  'want sandwich go firehous',
  'side greek salad greek dress tasti pita hummu refresh',
  'order duck rare pink tender insid nice char outsid',
  'came run us realiz husband left sunglass tabl',
  'chow mein good',
  'horribl attitud toward custom talk one custom enjoy food',
  'portion huge',
  'love friendli server great food wonder imagin menu',
  'heart attack grill downtown vega absolut flat line excus restaur',
  'much seafood like string pasta bottom',

```

Vectorization

```

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=1500)

```

```
X = cv.fit_transform(corpus).toarray()
y = df.iloc[:,1].values
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)
X_train.shape,X_test.shape, y_train.shape,y_test.shape

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

▾ Classsification Algorithm

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
gradient_boosting_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1)
gradient_boosting_classifier.fit(X_train, y_train)
y_pred_gradient_boosting = gradient_boosting_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix_gradient_boosting = confusion_matrix(y_test, y_pred_gradient_boosting)
print("Confusion Matrix:\n{}".format(confusion_matrix_gradient_boosting))
from sklearn.metrics import accuracy_score
accuracy_gradient_boosting = accuracy_score(y_test, y_pred_gradient_boosting)
print("Accuracy Score: {:.2f}%".format(accuracy_gradient_boosting * 100))
from sklearn.metrics import precision_score, recall_score
precision_gradient_boosting = precision_score(y_test, y_pred_gradient_boosting)
recall_gradient_boosting = recall_score(y_test, y_pred_gradient_boosting)
print("Precision: {:.2f}".format(precision_gradient_boosting))
print("Recall: {:.2f}".format(recall_gradient_boosting))
```

```
Confusion Matrix:
[[138  14]
 [ 62  86]]
Accuracy Score: 74.67%
Precision: 0.86
Recall: 0.58
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
decision_tree_classifier = DecisionTreeClassifier()
decision_tree_classifier.fit(X_train, y_train)
y_pred_decision_tree = decision_tree_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix_decision_tree = confusion_matrix(y_test, y_pred_decision_tree)
print("Confusion Matrix:\n{}".format(confusion_matrix_decision_tree))
from sklearn.metrics import accuracy_score
accuracy_decision_tree = accuracy_score(y_test, y_pred_decision_tree)
print("Accuracy Score: {:.2f}%".format(accuracy_decision_tree * 100))
from sklearn.metrics import precision_score, recall_score
precision_decision_tree = precision_score(y_test, y_pred_decision_tree)
recall_decision_tree = recall_score(y_test, y_pred_decision_tree)
print("Precision: {:.2f}".format(precision_decision_tree))
print("Recall: {:.2f}".format(recall_decision_tree))
```

```
Confusion Matrix:
[[119  33]
 [ 52  96]]
Accuracy Score: 71.67%
Precision: 0.74
Recall: 0.65
```

Multilayer Perceptron

```
from sklearn.neural_network import MLPClassifier
mlp_classifier = MLPClassifier(hidden_layer_sizes=(100, 100), max_iter=1000)
mlp_classifier.fit(X_train, y_train)
y_pred_mlp = mlp_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix_mlp = confusion_matrix(y_test, y_pred_mlp)
```

```
print("Confusion Matrix:\n{}".format(confusion_matrix_mlp))
from sklearn.metrics import accuracy_score
accuracy_mlp = accuracy_score(y_test, y_pred_mlp)
print("Accuracy Score: {:.2f}%".format(accuracy_mlp * 100))
from sklearn.metrics import precision_score, recall_score
precision_mlp = precision_score(y_test, y_pred_mlp)
recall_mlp = recall_score(y_test, y_pred_mlp)
print("Precision: {:.2f}".format(precision_mlp))
print("Recall: {:.2f}".format(recall_mlp))
```

```
Confusion Matrix:
[[118  34]
 [ 42 106]]
Accuracy Score: 74.67%
Precision: 0.76
Recall: 0.72
```

▼ Analysis & Conclusion

Gradient Boosting: Accuracy - 74.67

Decision Tree: Accuracy - 71.67

Multilayer Perception: Accuracy - 74.67