In [1]:
```
# Importing necessary libraries for Data Analysis
import pandas as pd
import numpy as np
```

In [2]:
```
from nltk.corpus import stopwords
import string
import nltk
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Hastee\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [3]:
```
# Load the dataset
try:
    df = pd.read_csv('Restaurant_Reviews.tsv', delimiter='\t', on_bad_lines='warn')
except pd.errors.ParserError as e:
    print(f"Error parsing file: {e}")
```

In [4]:
```
df
```

Out[4]:

	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
...	...	...
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

1000 rows × 2 columns

In [5]:
```
#preprocessing the data to remove stopwords
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_words])
    return text
```

In [6]:
```
#Reviwing the data without stopwords
df['Review'] = df['Review'].apply(preprocess_text)
df1 = df.copy()
df1.head()
```

Out[6]:

	Review	Liked
0	wow loved place	1
1	crust good	0
2	tasty texture nasty	0
3	stopped late may bank holiday rick steve recom...	1
4	selection menu great prices	1

In [7]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [8]:
```
vectorizer1 = CountVectorizer(binary = True)
vectorizer2 = CountVectorizer(binary = False)
```

In [9]:
```
x =df1['Review'].str.lower()
y = df1['Liked']
```

In [10]:
```
x1 = vectorizer1.fit_transform(x)
x2 = vectorizer2.fit_transform(x)
```

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
xtrain1,xtest1,ytrain,ytest = train_test_split(x1,y,test_size=0.25,random_state=42) #Bernoulli with counter vectorize
xtrain2,xtest2,ytrain,ytest = train_test_split(x2,y,test_size=0.25,random_state=42) #multinomial with counter vectori
xtrain3,xtest3,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=42) #MultinomialNB with TfidfVectorize
```

In [13]:
```
from sklearn.naive_bayes import BernoulliNB,MultinomialNB
```

In [14]:
```
bnb = BernoulliNB()
```

In [15]:
```
#Bernoulli with counter vectorizer
bnb.fit(xtrain1,ytrain)
```

Out[15]:
```
BernoulliNB
BernoulliNB()
```

In [16]:
```
# Make predictions on the testing set for Bernoulli with counter vectorizer
predictions1 = bnb.predict(xtest1)
```

In [17]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [18]:
```
mnb = MultinomialNB()
```

In [19]:
```
#multinomial with counter vectorizer
mnb.fit(xtrain2,ytrain)
```

Out[19]:
```
MultinomialNB
MultinomialNB()
```

In [20]:
```
# Make predictions on the testing set for multinomial with counter vectorizer
predictions2 = mnb.predict(xtest2)
```

In [21]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [22]:
```
# Create a pipeline with TfidfVectorizer and MultinomialNB
from sklearn.pipeline import make_pipeline
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

In [23]:
```
# Multinomial with Tfidf vectorizer
model.fit(xtrain3,ytrain)
```

Out[23]:
```
Pipeline
TfidfVectorizer
MultinomialNB
```

In [24]:
```
# Make predictions on the test set Multinomial with Tfidf vectorizer
predictions3 = model.predict(xtest3)
```

In [25]:
```
# Import confusion_matrix and classification_report from the sklearn.metrics module
from sklearn.metrics import accuracy_score
```

In [26]:
```
# Evaluate the model of Bernoulli with counter vectorizer
accuracy_score(ytest,predictions1)
```

Out[26]: 0.776

In [27]:
```
# Evaluate the model of Multinomial with counter vectorizer
accuracy_score(ytest,predictions2)
```

Out[27]: 0.788

In [28]:
```
# Evaluate the model of Multinomial with Tfidf vectorizer
accuracy_score(ytest, predictions3)
```

Out[28]: 0.78

Conclusion

1. Bernoulli Naive Bayes with Count Vectorizer: Accuracy 0.776.
2. Multinomial Naive Bayes with Count Vectorizer: Accuracy 0.788.
3. Multinomial Naive Bayes with TF-IDF Vectorizer: Accuracy 0.78.

Multinomial Naive Bayes with Count Vectorizer provides the highest accuracy at 0.788, making it the best choice for this text classification task. The TF-IDF variant is also strong, but slightly less effective at 0.78. The Bernoulli model with binary features is the least effective at 0.776.

In [29]:
```
import joblib
# Save the Multinomial Naive Bayes model
model = 'Multinomial.joblib'
joblib.dump(mnb, model)
```

Out[29]: ['Multinomial.joblib']

In [30]:
```
#predicting the type of review with a dynamic input
def predict_rating(review):
    # Preprocess the review
    preprocessed_review = preprocess_text(review)

    # Transform the preprocessed review using the fitted CountVectorizer
    review_vectorized = vectorizer2.transform([preprocessed_review])

    # Predict the rating using the trained Multinomial Naive Bayes model
    predicted_rating = mnb.predict(review_vectorized)[0]

    return predicted_rating

# Get user input for the review
user_review = input("Enter your review: ")

# Predict the rating
predicted_rating = predict_rating(user_review)

# Print the numerical rating
print("Predicted Rating:", predicted_rating)

# Print the result based on the predicted rating
if predicted_rating == 1:
    print("This is a Positive Review.")
else:
    print("This is a Negative Review.")
```

```
Enter your review: had a good time, great service.
Predicted Rating: 1
This is a Positive Review.
```