In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('Restaurant_Reviews.tsv', sep='\t')
```

In [3]:

```
df.head()
```

Out[3]:

	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

In [4]:

```
df.tail()
```

Out[4]:

	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

In [5]:

```
df.shape
```

Out[5]:

```
(1000, 2)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Review', 'Liked'], dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
4
```

In [8]:

```
df = df.drop_duplicates()
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

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

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 996 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  996 non-null    object
 1   Liked   996 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 23.3+ KB
```

In [11]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [12]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [13]:

```
df['Liked'].unique()
```

Out[13]:

```
array([1, 0], dtype=int64)
```

In [14]:

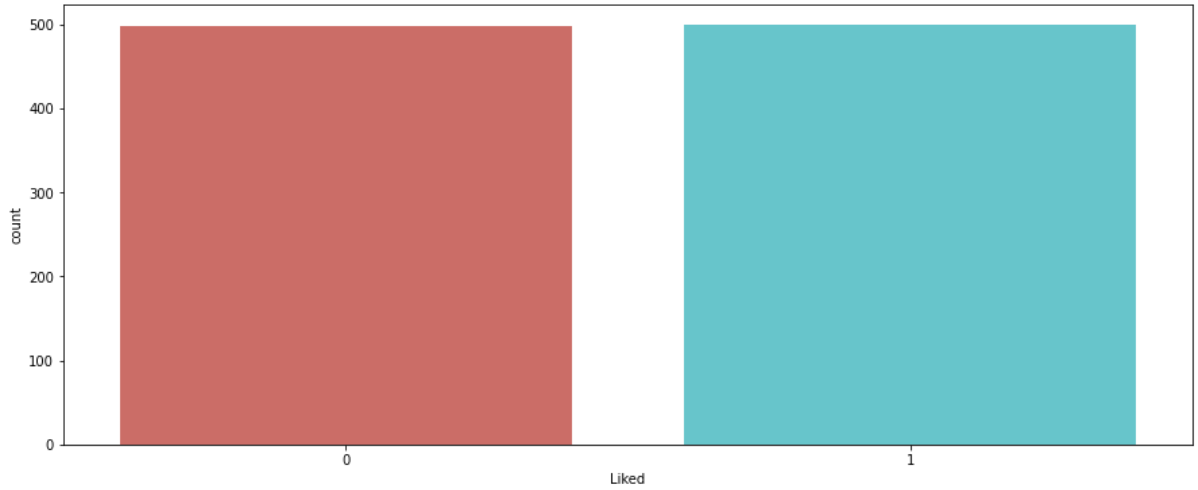
```
df['Liked'].value_counts()
```

Out[14]:

```
1    499
0    497
Name: Liked, dtype: int64
```

In [15]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['Liked'], data = df, palette = 'hls')
plt.show()
```



In [16]:

```
balance_counts = df.groupby('Liked')['Liked'].agg('count').values
balance_counts
```

Out[16]:

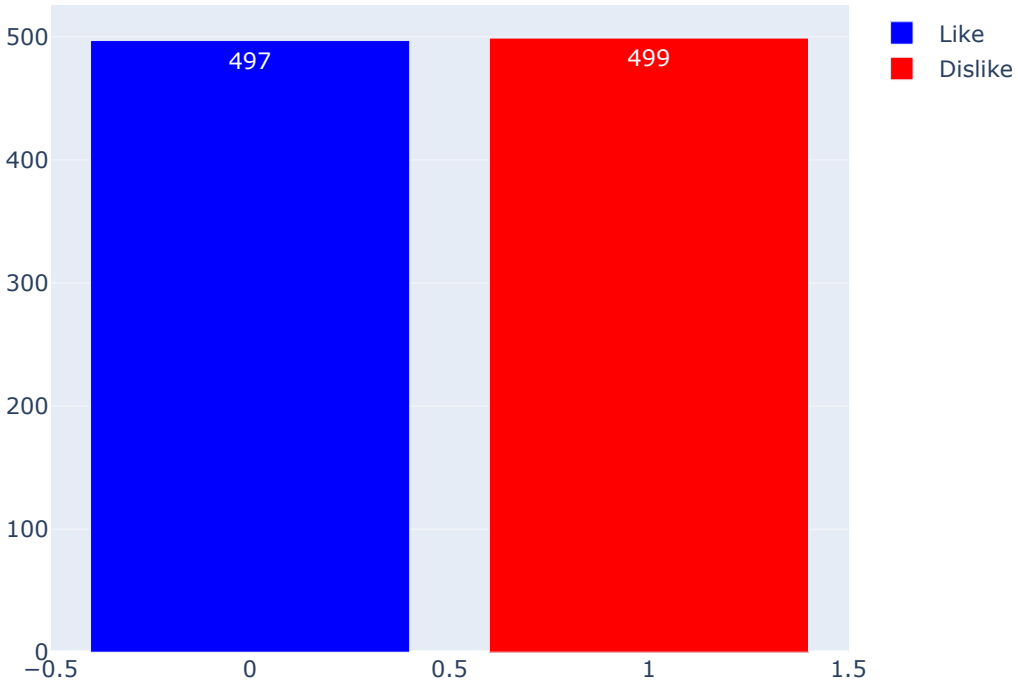
```
array([497, 499], dtype=int64)
```

In [17]:

```
from plotly import graph_objs as go
import plotly.express as px
import plotly.figure_factory as ff
```

In [18]:

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x = [0],
    y=[balance_counts[0]],
    name='Like',
    text=[balance_counts[0]],
    textposition='auto',
    marker_color= 'blue'
))
fig.add_trace(go.Bar(
    x= [1],
    y=[balance_counts[1]],
    name='Dislike',
    text=[balance_counts[1]],
    textposition='auto',
    marker_color= 'red'
))
fig.update_layout(
    title='<span style="font-size:32px; font-family:Times New Roman">Dataset distribution by Likes
)
fig.show()
```

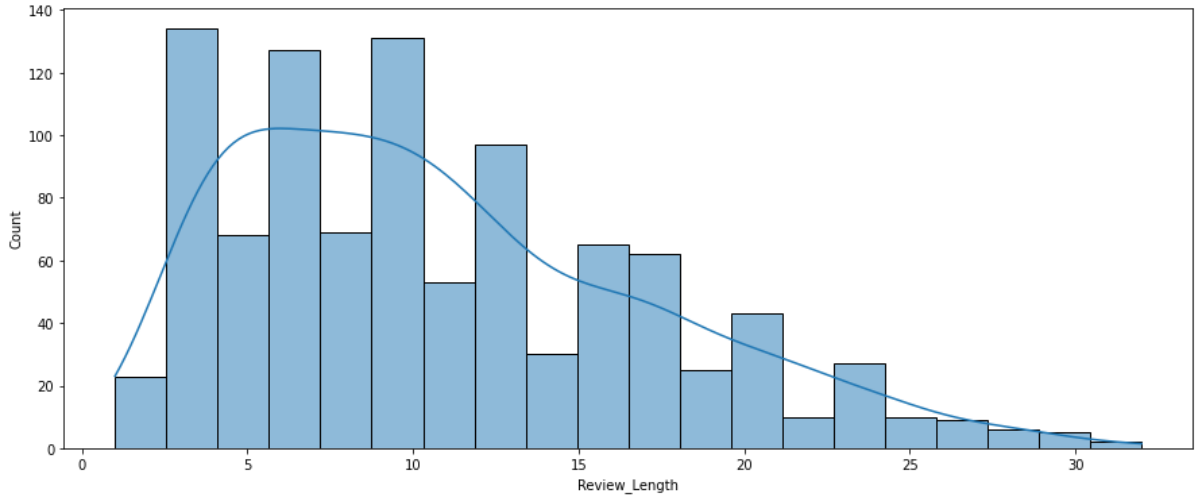


In [19]:

```
df['Review_Length'] = df['Review'].apply(lambda x: len(x.split(' ')))
```

In [20]:

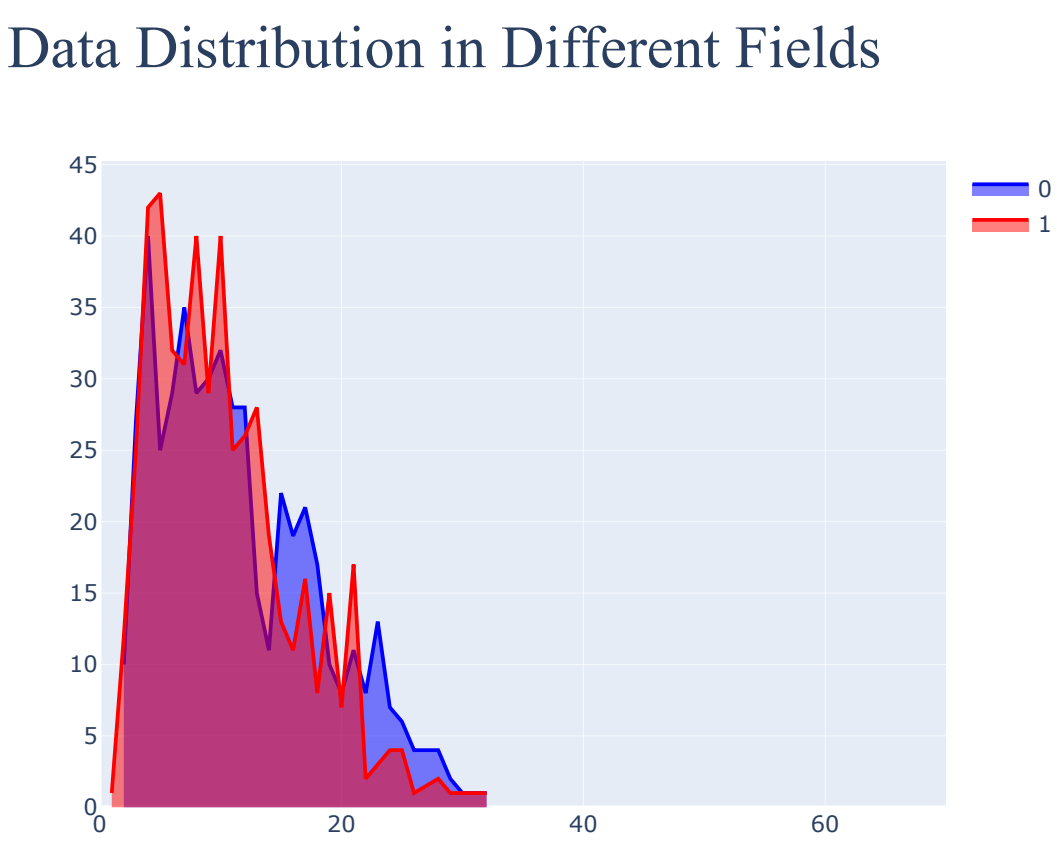
```
plt.figure(figsize=(15,6))
sns.histplot(df['Review_Length'], bins = 20, kde = True, palette = 'hls')
plt.show()
```



In [21]:

```
like_df = df[df['Liked'] == 0]['Review_Length'].value_counts().sort_index()
dislike_df = df[df['Liked'] == 1]['Review_Length'].value_counts().sort_index()

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=like_df.index,
    y=like_df.values,
    name= 0,
    fill='tozeroy',
    marker_color= 'blue',
))
fig.add_trace(go.Scatter(
    x=dislike_df.index,
    y=dislike_df.values,
    name=1,
    fill='tozeroy',
    marker_color= 'red',
))
fig.update_layout(
    title='<span style="font-size:32px; font-family:Times New Roman">Data Distribution in Differe
)
fig.update_xaxes(range=[0, 70])
fig.show()
```



In [22]:

```
df
```

Out[22]:

	Review	Liked	Review_Length
0	Wow... Loved this place.	1	4
1	Crust is not good.	0	4
2	Not tasty and the texture was just nasty.	0	8
3	Stopped by during the late May bank holiday of...	1	15
4	The selection on the menu was great and so wer...	1	12
...	...	...	...
995	I think food should have flavor and texture an...	0	12
996	Appetite instantly gone.	0	3
997	Overall I was not impressed and would not go b...	0	10
998	The whole experience was underwhelming, and I ...	0	16
999	Then, as if I hadn't wasted enough of my life ...	0	28

996 rows × 3 columns

In [23]:

```
df.Review_Length.describe()
```

Out[23]:

```
count    996.000000
mean      10.919679
std        6.256621
min        1.000000
25%        6.000000
50%       10.000000
75%       15.000000
max       32.000000
Name: Review_Length, dtype: float64
```

In [24]:

```
df_new = df.copy()
```

In [25]:

```
def clean_text(text):
    text = text.lower()
    return text.strip()
```

In [26]:

```
df_new.message = df_new.Review.apply(lambda x: clean_text(x))
```

In [27]:

```
import string
string.punctuation
```

Out[27]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [28]:

```
def remove_punctuation(text):
    punctuationfree="".join([i for i in text if i not in string.punctuation])
    return punctuationfree
df_new['Review']= df_new['Review'].apply(lambda x:remove_punctuation(x))
```

In [29]:

```
import re
def tokenization(text):
    tokens = re.split('W+',text)
    return tokens
df_new['Review']= df_new['Review'].apply(lambda x: tokenization(x))
```

In [30]:

```
import nltk
stopwords = nltk.corpus.stopwords.words('english')
```

In [31]:

```
def remove_stopwords(text):
    output= " ".join(i for i in text if i not in stopwords)
    return output
```

In [32]:

```
df_new['Review']= df_new['Review'].apply(lambda x:remove_stopwords(x))
```

In [33]:

```
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()
```

In [34]:

```
def stemming(text):
    stem_text = "".join([porter_stemmer.stem(word) for word in text])
    return stem_text
df_new['Review']=df_new['Review'].apply(lambda x: stemming(x))
```

In [35]:

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
```

In [36]:

```
def lemmatizer(text):
    lemm_text = "".join([wordnet_lemmatizer.lemmatize(word) for word in text])
    return lemm_text
df_new['Review']=df_new['Review'].apply(lambda x:lemmatizer(x))
```

In [37]:

```
def clean_text(text):
    text = re.sub('\[.*\]','', text).strip() # Remove text in square brackets
    text = re.sub('\S*\d\S*\s*','', text).strip()  # Remove words containing numbers
    return text.strip()
```

In [38]:

```
df_new['Review'] = df_new.Review.apply(lambda x: clean_text(x))
```

In [39]:

```
import spacy
nlp = spacy.load('en_core_web_sm')
```

In [40]:

```
stopwords = nlp.Defaults.stop_words
def lemmatizer(text):
    doc = nlp(text)
    sent = [token.lemma_ for token in doc if not token.text in set(stopwords)]
    return ' '.join(sent)
```

In [41]:

```
df_new['Review'] =  df_new.Review.apply(lambda x: lemmatizer(x))
```

In [42]:

```
def remove_urls(vTEXT):
    vTEXT = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', vTEXT, flags=re.MULTILINE)
    return(vTEXT)
```

In [43]:

```
df_new['Review'] = df_new.Review.apply(lambda x: remove_urls(x))
```

In [44]:

```
def remove_digits(text):
    clean_text = re.sub(r"\b[0-9]+\b\s*", "", text)
    return(text)
```

In [45]:

```
df_new['Review'] = df_new.Review.apply(lambda x: remove_digits(x))
```

In [46]:

```
def remove_digits1(sample_text):
    clean_text = " ".join([w for w in sample_text.split() if not w.isdigit()]) # Side effect: rem
    return(clean_text)
```

In [47]:

```
df_new['Review'] = df_new.Review.apply(lambda x: remove_digits1(x))
```

In [48]:

```
def remove_emojis(data):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               "]+", flags=re.UNICODE)
    return re.sub(emoji_pattern, '', data)
```

In [49]:

```
df_new['Review'] = df_new.Review.apply(lambda x: remove_emojis(x))
```

In [50]:

```
df_new
```

Out[50]:

	Review	Liked	Review_Length
0	ow love place	1	4
1	crust good	0	4
2	tasty texture nasty	0	8
3	stop late bank holiday rick steve recommendati...	1	15
4	selection menu great price	1	12
...	...	...	...
995	think food flavor texture lack	0	12
996	appetite instantly go	0	3
997	overall impressed	0	10
998	experience underwhelme think ninja sushi time	0	16
999	not waste life pour salt wound draw time take ...	0	28

996 rows × 3 columns

In [51]:

```
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [52]:

```
import numpy as np
```

In [53]:

```
twitter_mask = np.array(Image.open('twitter_mask.png'))

wc = WordCloud(
    background_color='white',
    max_words=200,
    mask=twitter_mask,
)
wc.generate(' '.join(text for text in df_new.loc[df_new['Liked'] == 0, 'Review']))
plt.figure(figsize=(18,10))
plt.title('Top words for 0 Reviews',
          fontdict={'size': 22,  'verticalalignment': 'bottom'})
plt.imshow(wc)
plt.axis("off")
plt.show()
```

In [54]:

```
twitter_mask = np.array(Image.open('twitter_mask.png'))

wc = WordCloud(
    background_color='white',
    max_words=200,
    mask=twitter_mask,
)
wc.generate(' '.join(text for text in df_new.loc[df_new['Liked'] == 1, 'Review']))
plt.figure(figsize=(18,10))
plt.title('Top words for 1 Reviews',
          fontdict={'size': 22,  'verticalalignment': 'bottom'})
plt.imshow(wc)
plt.axis("off")
plt.show()
```

In [55]:

```
x = df_new['Review']
y = df_new['Liked']

print(len(x), len(y))
```

```
996 996
```

In [56]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42)
print(len(x_train), len(y_train))
print(len(x_test), len(y_test))
```

```
747 747
249 249
```

In [57]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline
```

In [58]:

```
from sklearn.naive_bayes import MultinomialNB
```

In [59]:

```
pipe = Pipeline([('bow', CountVectorizer()),
                 ('tfid', TfidfTransformer()),
                 ('model', MultinomialNB())])
```

In [60]:

```
pipe.fit(x_train, y_train)
y_pred_class = pipe.predict(x_test)
```

In [61]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [62]:

```
print(accuracy_score(y_test, y_pred_class))
```

```
0.7751004016064257
```

In [63]:

```
print(confusion_matrix(y_test, y_pred_class))
```

```
[[99 33]
 [23 94]]
```

In [64]:

```
print(classification_report(y_test, y_pred_class))
```

```
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       132
           1       0.74      0.80      0.77       117

    accuracy                           0.78       249
   macro avg       0.78      0.78      0.78       249
weighted avg       0.78      0.78      0.78       249
```

In [65]:

```
import xgboost as xgb

pipe = Pipeline([
    ('bow', CountVectorizer()),
    ('tfid', TfidfTransformer()),
    ('model', xgb.XGBClassifier(
        learning_rate=0.1,
        max_depth=7,
        n_estimators=80,
        use_label_encoder=False,
        eval_metric='auc',
        # colsample_bytree=0.8,
        # subsample=0.7,
        # min_child_weight=5,
    ))
])
```

In [66]:

```
pipe.fit(x_train, y_train)
y_pred_class = pipe.predict(x_test)
```

In [67]:

```
print(accuracy_score(y_test, y_pred_class))
```

```
0.7108433734939759
```

In [68]:

```
print(confusion_matrix(y_test, y_pred_class))
```

```
[[112  20]
 [ 52  65]]
```

In [69]:

```
print(classification_report(y_test, y_pred_class))
```

```
              precision    recall  f1-score   support

           0       0.68      0.85      0.76       132
           1       0.76      0.56      0.64       117

    accuracy                           0.71       249
   macro avg       0.72      0.70      0.70       249
weighted avg       0.72      0.71      0.70       249
```