Importing Modules

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import joblib
import seaborn as sns
```

In [2]:
```
warnings.filterwarnings('ignore')
```

Read the dataset

In [3]:
```
data = pd.read_csv("C:\\Users\\Dell\\Downloads\\Restaurant_Reviews.tsv",sep='\t')
```

In [4]:
```
data.head()
```

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

In [5]:
```
data.shape
```

Out[5]:
```
(1000, 2)
```

Preprocessing the data

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Review    0
Liked     0
dtype: int64
```

In [7]:
```
data['Liked'].value_counts()
```

Out[7]:
```
1    500
0    500
Name: Liked, dtype: int64
```

In [8]:
```
data.head()
```

Out[8]:

	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 [9]:
```
data['char_count']=data['Review'].apply(len)
```

In [10]:
```
data.head()
```
Out[10]:

	Review	Liked	char_count
0	Wow... Loved this place.	1	24
1	Crust is not good.	0	18
2	Not tasty and the texture was just nasty.	0	41
3	Stopped by during the late May bank holiday of...	1	87
4	The selection on the menu was great and so wer...	1	59

In [11]:
```
data['word_count']=data['Review'].apply(lambda x :len(str(x).split()))
```
In [12]:
```
data.head()
```
Out[12]:

	Review	Liked	char_count	word_count
0	Wow... Loved this place.	1	24	4
1	Crust is not good.	0	18	4
2	Not tasty and the texture was just nasty.	0	41	8
3	Stopped by during the late May bank holiday of...	1	87	15
4	The selection on the menu was great and so wer...	1	59	12

In [13]:
```
import nltk
```
In [14]:
```
nltk.download('punkt')
```
```
[nltk_data] Error loading punkt: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```
Out[14]:
```
False
```
In [15]:
```
data['sent_count']=data['Review'].apply(lambda x : len(nltk.sent_tokenize(str(x))))
```
In [16]:
```
data.head()
```
Out[16]:

	Review	Liked	char_count	word_count	sent_count
0	Wow... Loved this place.	1	24	4	2
1	Crust is not good.	0	18	4	1
2	Not tasty and the texture was just nasty.	0	41	8	1
3	Stopped by during the late May bank holiday of...	1	87	15	1
4	The selection on the menu was great and so wer...	1	59	12	1

In [17]:
```
data[data['Liked']==1]['char_count'].mean()
```
Out[17]:
```
55.88
```
In [18]:
```
data[data['Liked']==0]['char_count'].mean()
```
Out[18]:
```
60.75
```
In [19]:
```
import re
```
In [20]:
```
data['Review'][1]
```
Out[20]:
```
'Crust is not good.'
```
In [21]:
```
review = re.sub('[^a-zA-Z]',' ',data['Review'][1])
```
In [22]:
```
review
```
Out[22]:
```
'Crust is not good '
```

```
In [23]: review = review.lower()
```

```
In [24]: review
```

```
Out[24]: 'crust is not good '
```

```
In [25]: review = review.split()
```

```
In [26]: review
```

```
Out[26]: ['crust', 'is', 'not', 'good']
```

Stopwords Removing

```
In [27]: from nltk.corpus import stopwords
```

```
In [28]: all_stopwords = stopwords.words("english")
         all_stopwords.remove('not')
```

```
In [29]: all_stopwords
```

```
Out[29]: ['i',
          'me',
          'my',
          'myself',
          'we',
          'our',
          'ours',
          'ourselves',
          'you',
          "you're",
          "you've",
          "you'll",
          "you'd",
          'your',
          'yours',
          'yourself',
          'yourselves',
          'he',
          'him',
```

```
In [30]: review = [word for word in review if word not in set(all_stopwords)]
```

```
In [31]: review
```

```
Out[31]: ['crust', 'not', 'good']
```

```
In [32]: from nltk.stem.porter import PorterStemmer
```

```
In [33]: ps = PorterStemmer()
```

```
In [34]: review = [ps.stem(word) for word in review]
```

```
In [35]: review = " ".join(review)
```

```
In [36]: review
```

```
Out[36]: 'crust not good'
```

In [37]:
```
import re
custom_stopwords = {'don', "don't", '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", 'no', 'nor', 'not', 'shouldn', "shouldn't",
                    'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"}

corpus =[]
ps =PorterStemmer()
stop_words = set(stopwords.words("english")) - custom_stopwords

for i in range(len(data)):
    review = re.sub('[^a-zA-Z]',' ',data['Review'][i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if word not in stop_words]
    review = " ".join(review)
    corpus.append(review)
```

In [38]:
```
data['processed_text']=corpus
```

In [39]:
```
data.head()
```

Out[39]:

	Review	Liked	char_count	word_count	sent_count	processed_text
0	Wow... Loved this place.	1	24	4	2	wow love place
1	Crust is not good.	0	18	4	1	crust not good
2	Not tasty and the texture was just nasty.	0	41	8	1	not tasti textur nasti
3	Stopped by during the late May bank holiday of...	1	87	15	1	stop late may bank holiday rick steve recommen...
4	The selection on the menu was great and so wer...	1	59	12	1	select menu great price

Exploratory Data Analysis(EDA)

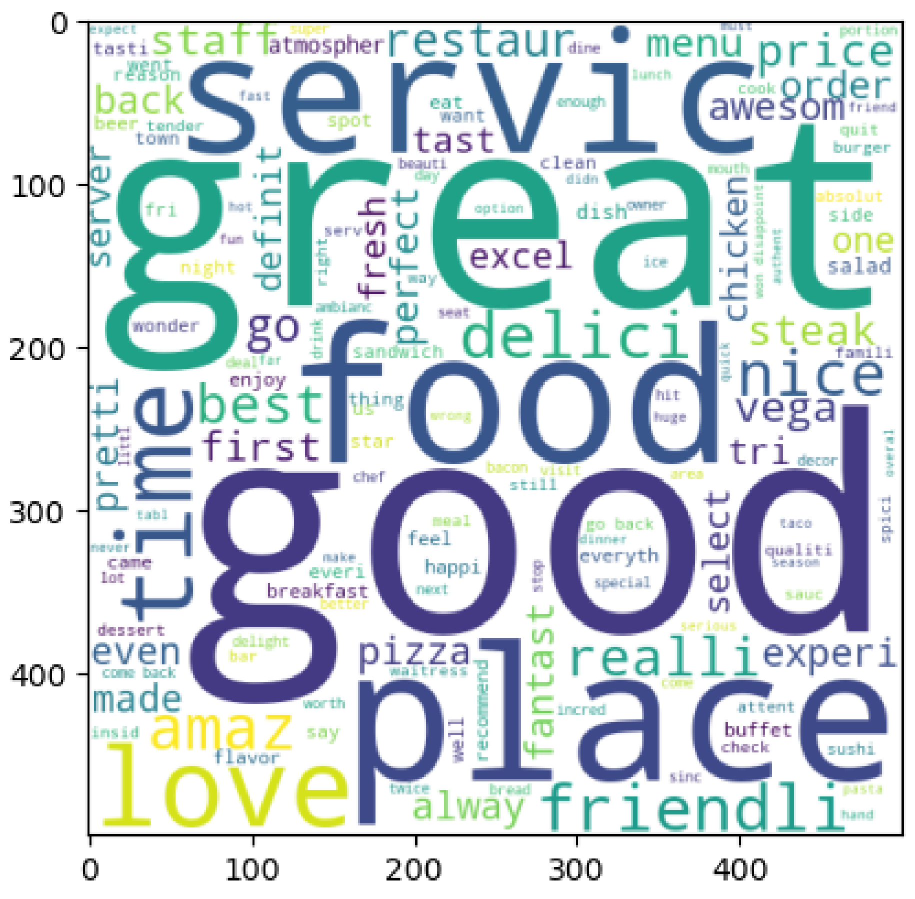
In [40]:
```
from wordcloud import WordCloud
```

In [41]:
```
wc = WordCloud(width=500,height=500,min_font_size=8,background_color="white")
```

In [42]:
```
pos = wc.generate(data[data['Liked']==1]['processed_text'].str.cat(sep=" "))
```

In [43]:
```
plt.imshow(pos)
```

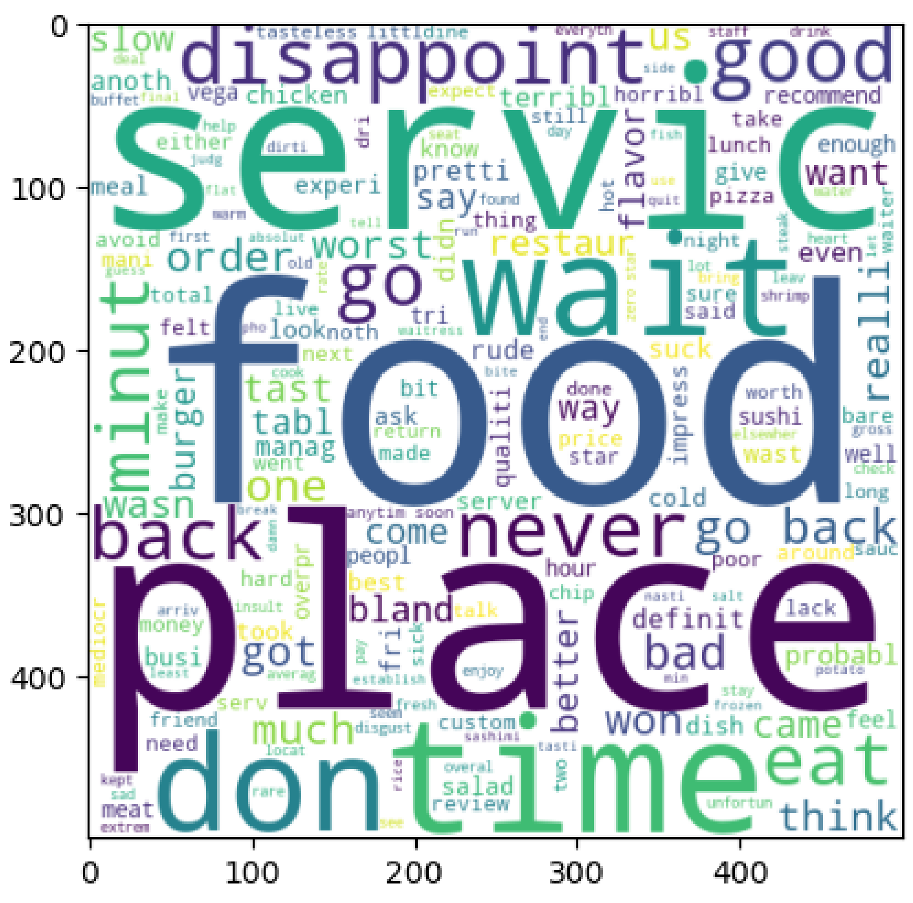
Out[43]:
```
<matplotlib.image.AxesImage at 0x278692cf9d0>
```



```
In [44]: negative = wc.generate(data[data['Liked']==0]['processed_text'].str.cat(sep=" "))
```

```
In [45]: plt.imshow(negative)
```

```
Out[45]: <matplotlib.image.AxesImage at 0x278693fbe90>
```



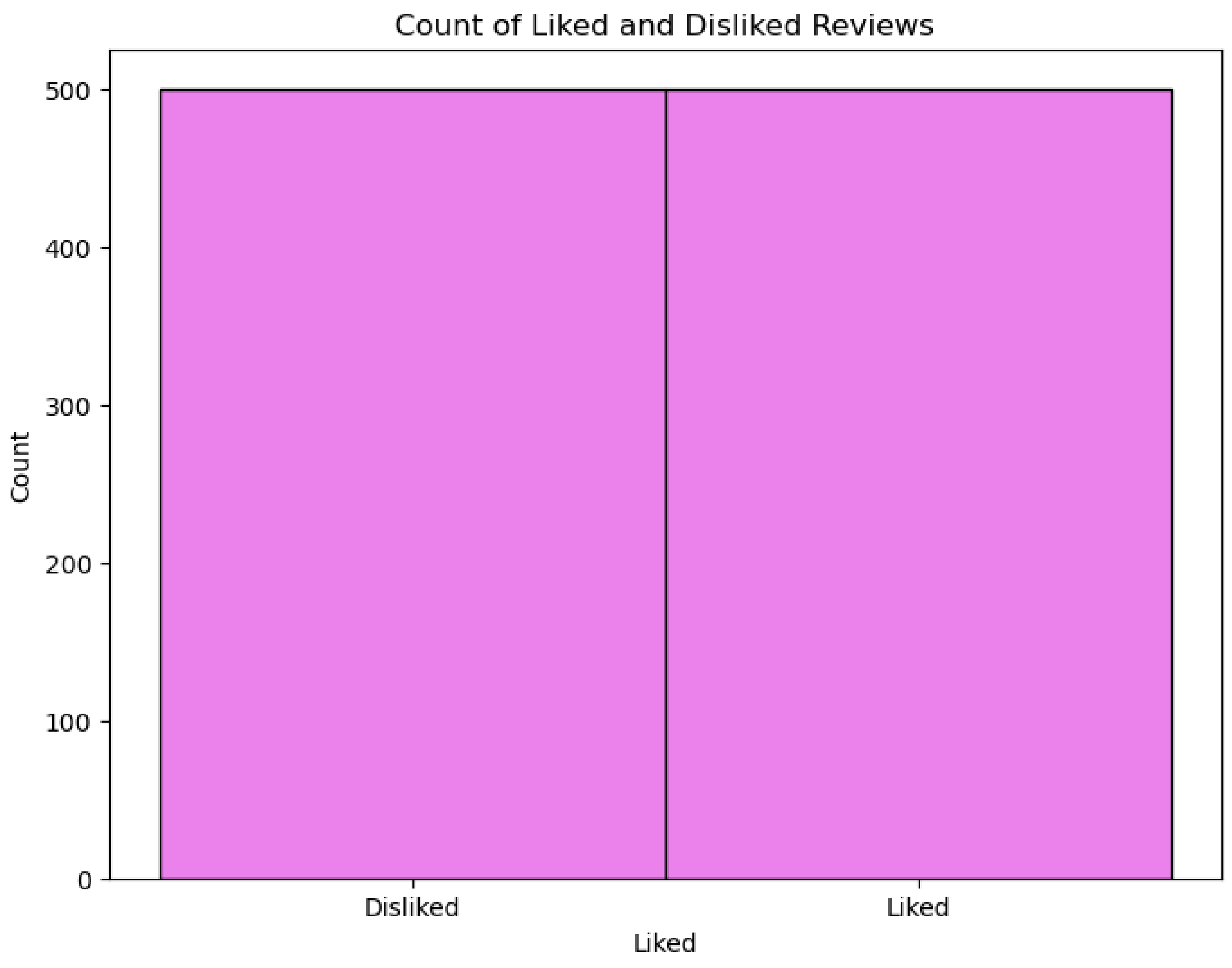
```
In [46]: data.head()
```

Out[46]:

	Review	Liked	char_count	word_count	sent_count	processed_text
0	Wow... Loved this place.	1	24	4	2	wow love place
1	Crust is not good.	0	18	4	1	crust not good
2	Not tasty and the texture was just nasty.	0	41	8	1	not tasti textur nasti
3	Stopped by during the late May bank holiday of...	1	87	15	1	stop late may bank holiday rick steve recommen...
4	The selection on the menu was great and so wer...	1	59	12	1	select menu great price

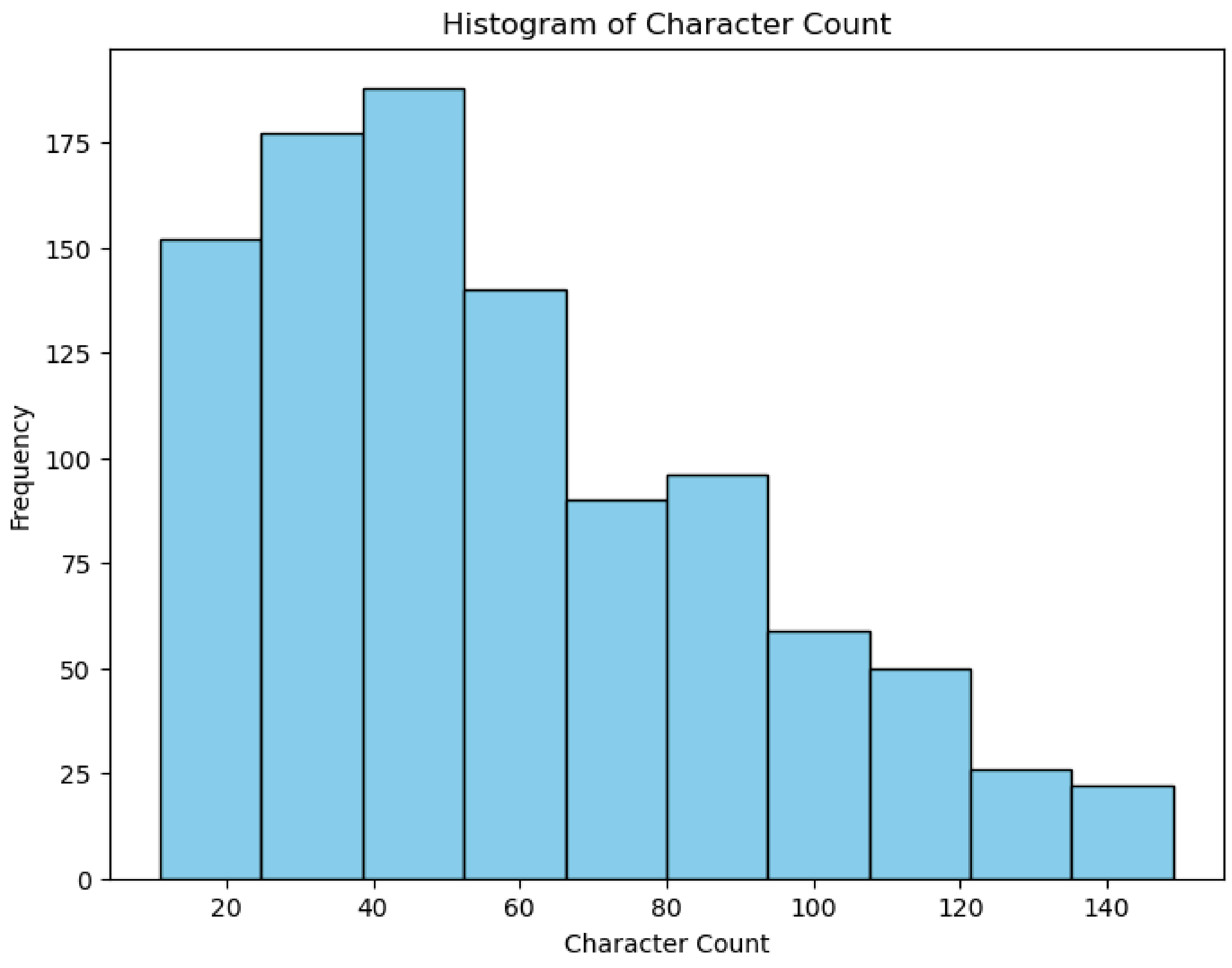
Count Plot

In [47]:
```
# Create a count plot for the 'Liked' column
plt.figure(figsize=(8, 6))
plt.hist(data['Liked'], color='violet', edgecolor='black', bins=[-0.5, 0.5, 1.5], align='mid')
plt.title('Count of Liked and Disliked Reviews')
plt.xlabel('Liked')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Disliked', 'Liked'])
plt.show()
```



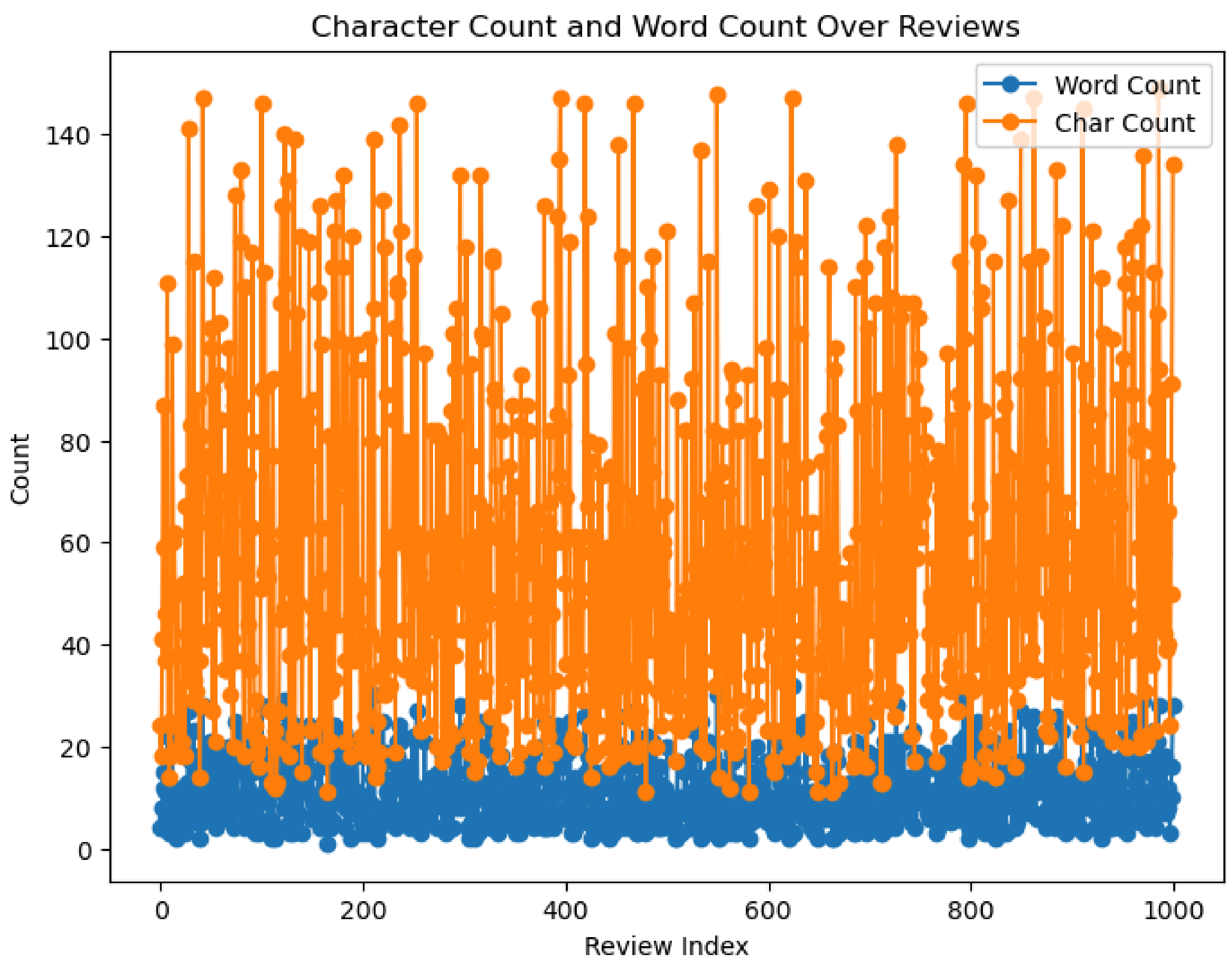
Histogram

```
In [48]: # Histogram for character count
         plt.figure(figsize=(8, 6))
         plt.hist(data['char_count'], bins=10, color='skyblue', edgecolor='black')
         plt.title('Histogram of Character Count')
         plt.xlabel('Character Count')
         plt.ylabel('Frequency')
         plt.show()
```



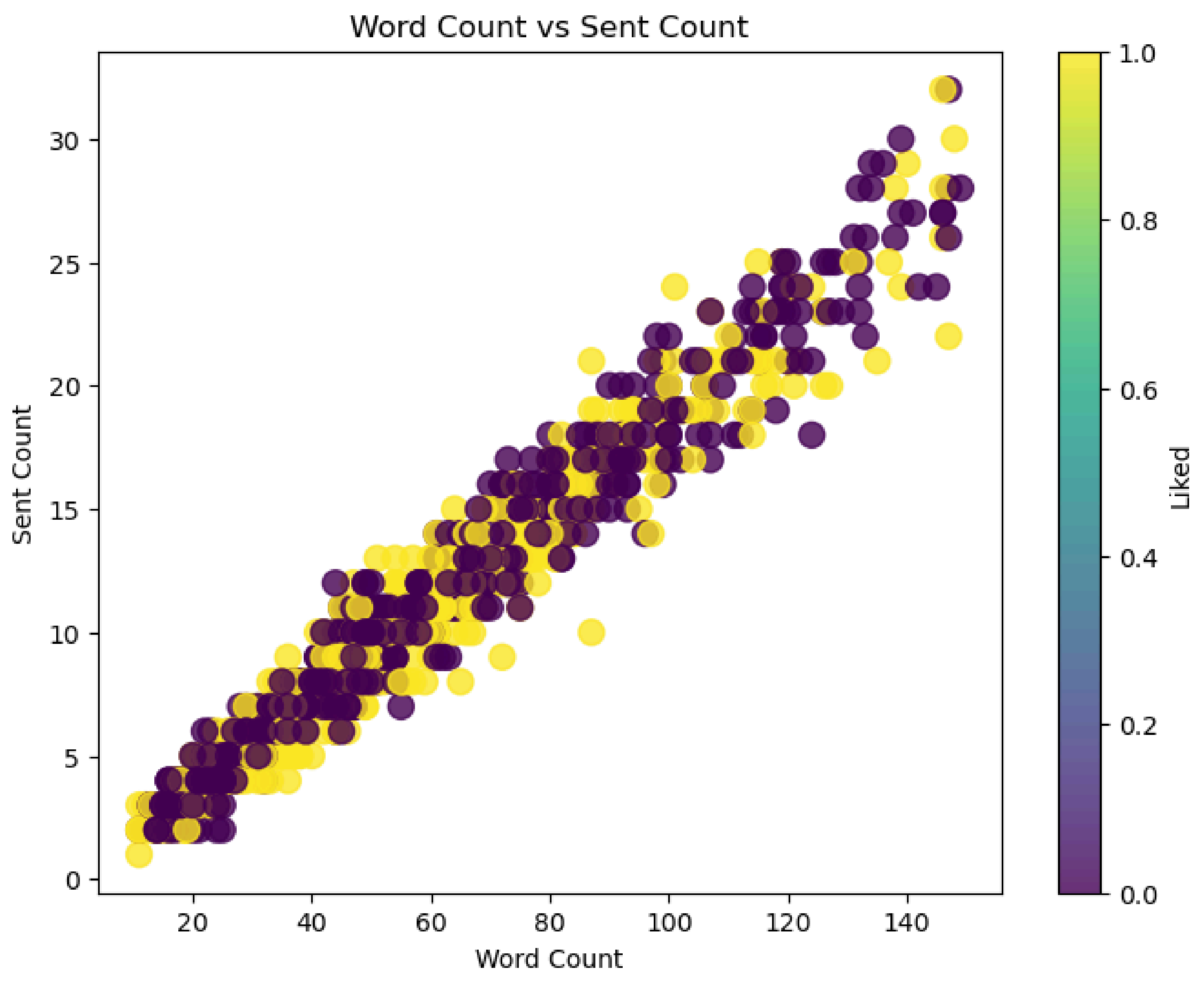
Line Plot

In [49]:
```
# Create a line plot
plt.figure(figsize=(8, 6))
plt.plot(data.index, data['word_count'], marker='o', label='Word Count')
plt.plot(data.index, data['char_count'], marker='o', label='Char Count')
plt.title('Character Count and Word Count Over Reviews')
plt.xlabel('Review Index')
plt.ylabel('Count')
plt.legend()
plt.show()
```



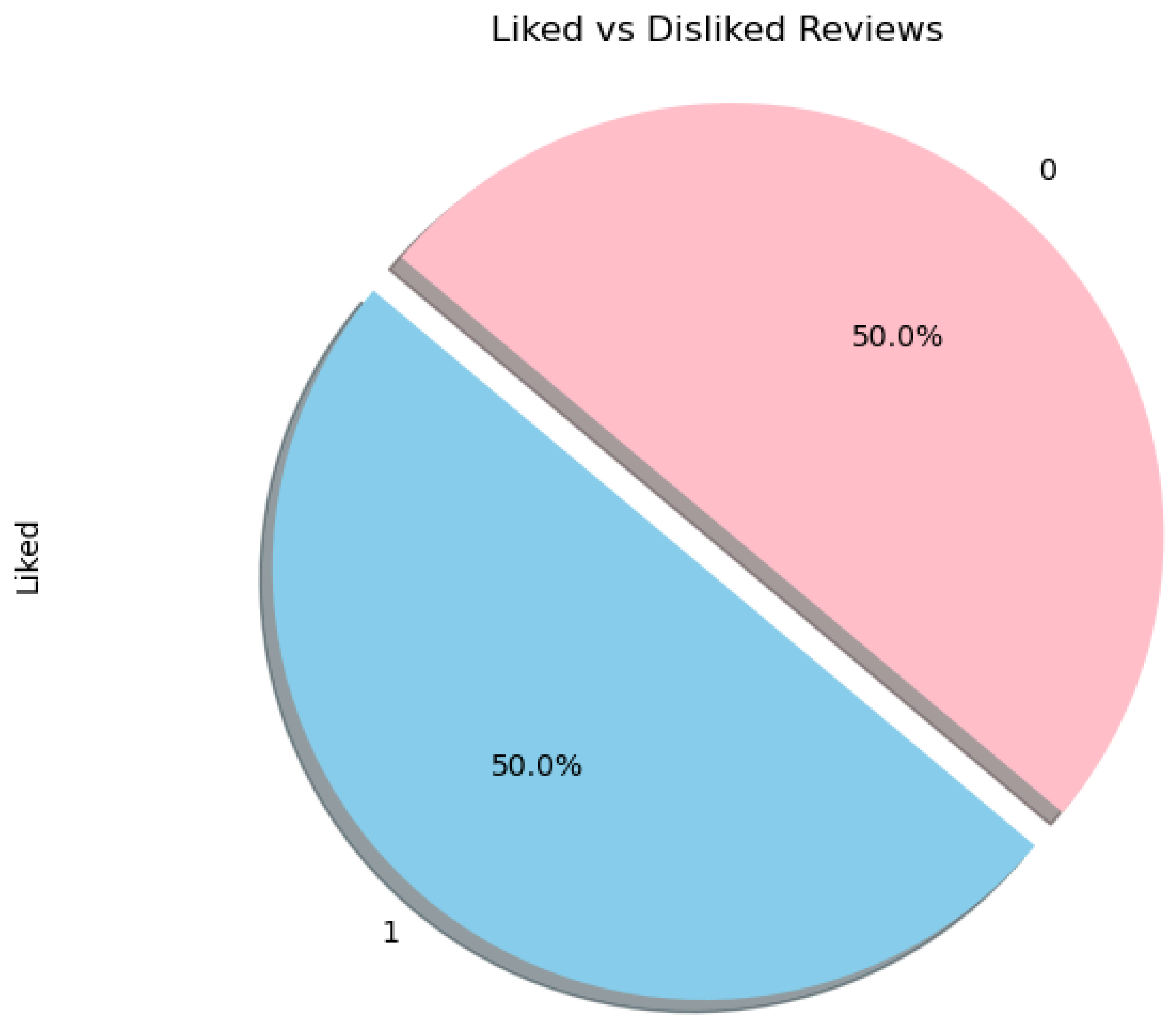
Scatter Plot

```
In [50]: # Create the Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(data['char_count'], data['word_count'], c=data['Liked'],s=100,alpha=0.8)
plt.title('Word Count vs Sent Count')
plt.xlabel('Word Count')
plt.ylabel('Sent Count')
plt.colorbar(label='Liked')
plt.show()
```



Pie Chart

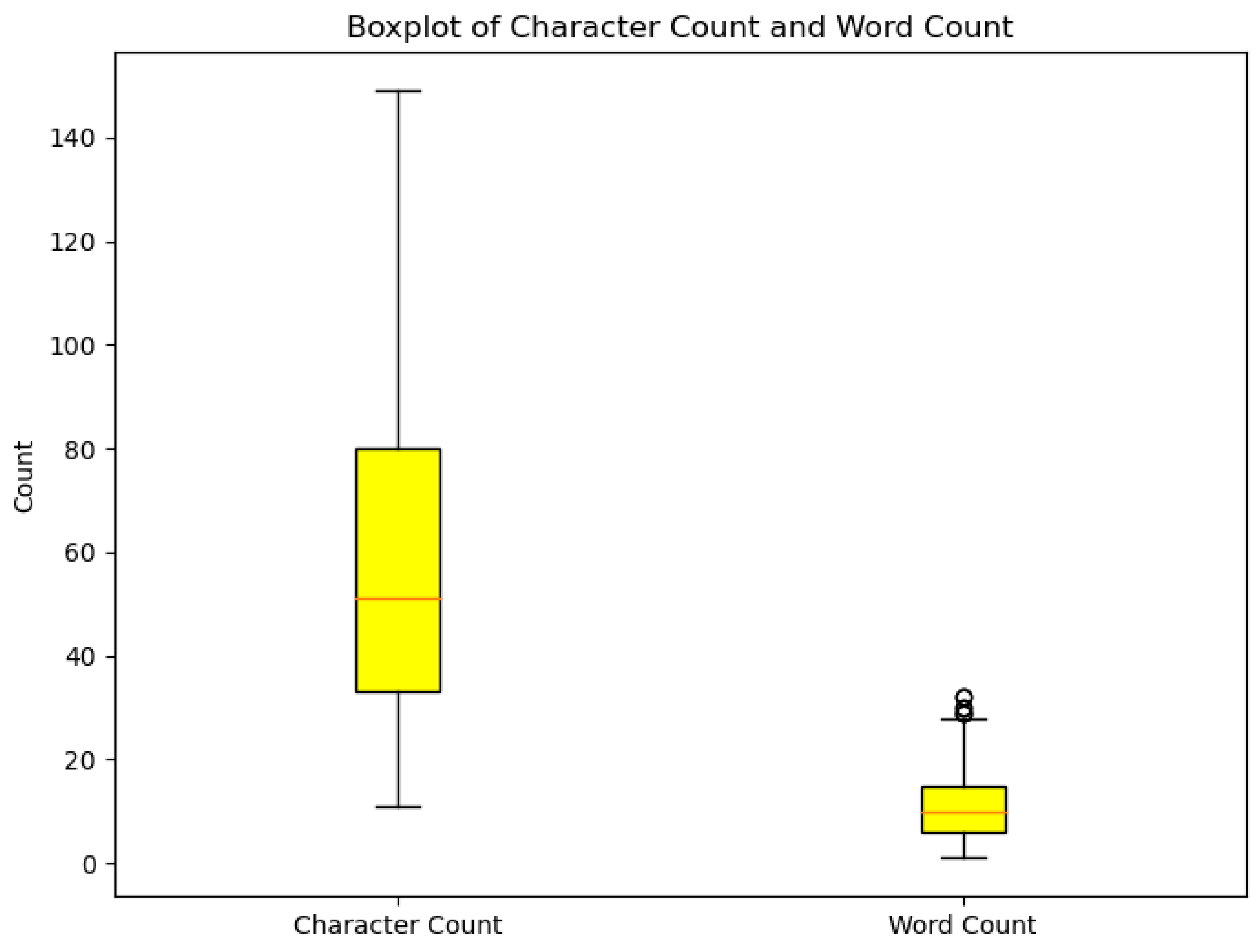
In [51]:
```
# Create a pie chart
plt.figure(figsize=(8, 6))
data['Liked'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['skyblue', 'pink'], explode=(0.1,
plt.title('Liked vs Disliked Reviews')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```



BoxPlot

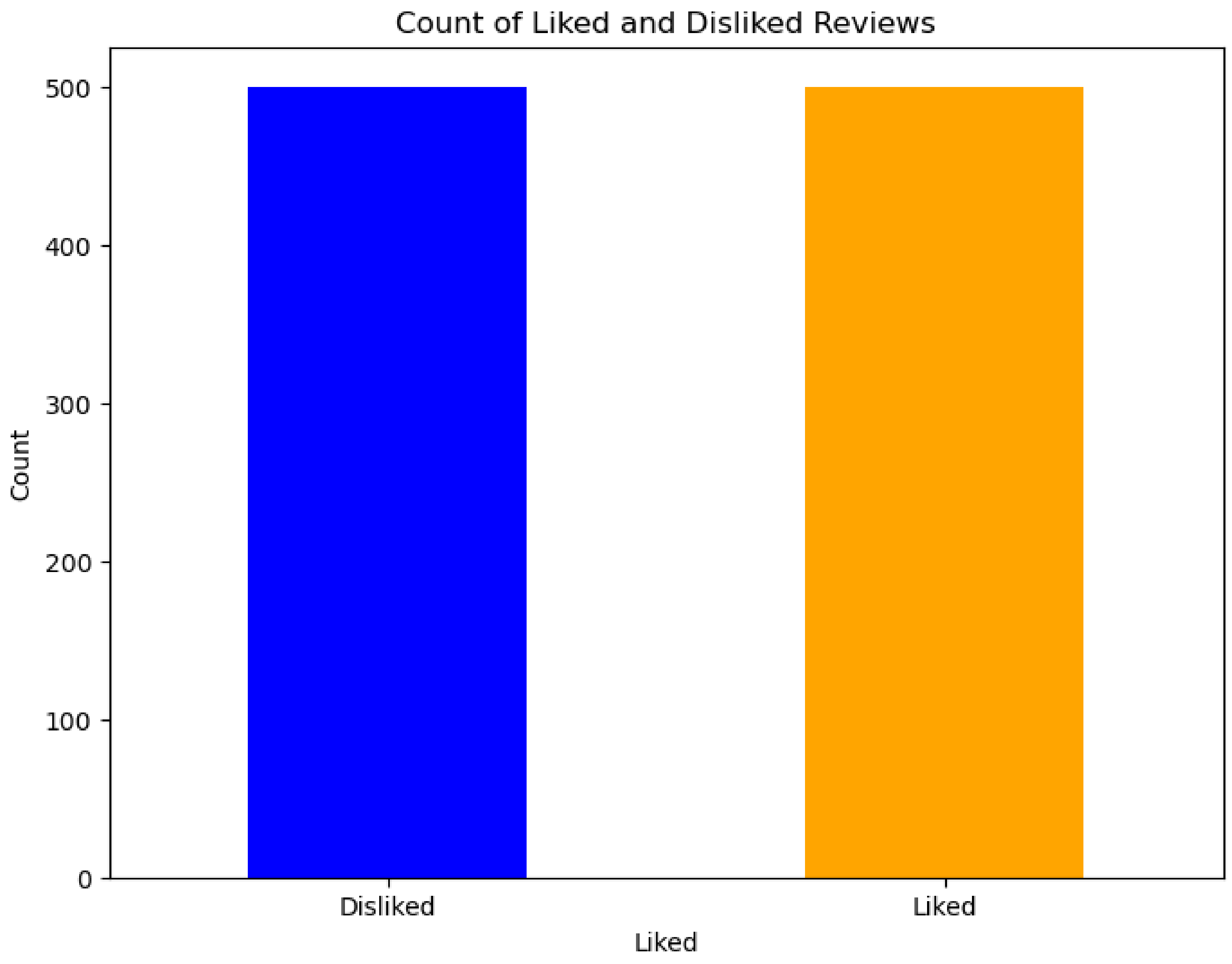
In [52]:
```python
# Boxplot for character count and word count with filled colors
plt.figure(figsize=(8, 6))
boxplot_data = [data['char_count'], data['word_count']]
boxprops = dict(facecolor='yellow', color='black')
plt.boxplot(boxplot_data, labels=['Character Count', 'Word Count'], patch_artist=True, boxprops=boxprops)

plt.title('Boxplot of Character Count and Word Count')
plt.ylabel('Count')
plt.show()
```



BarPlot

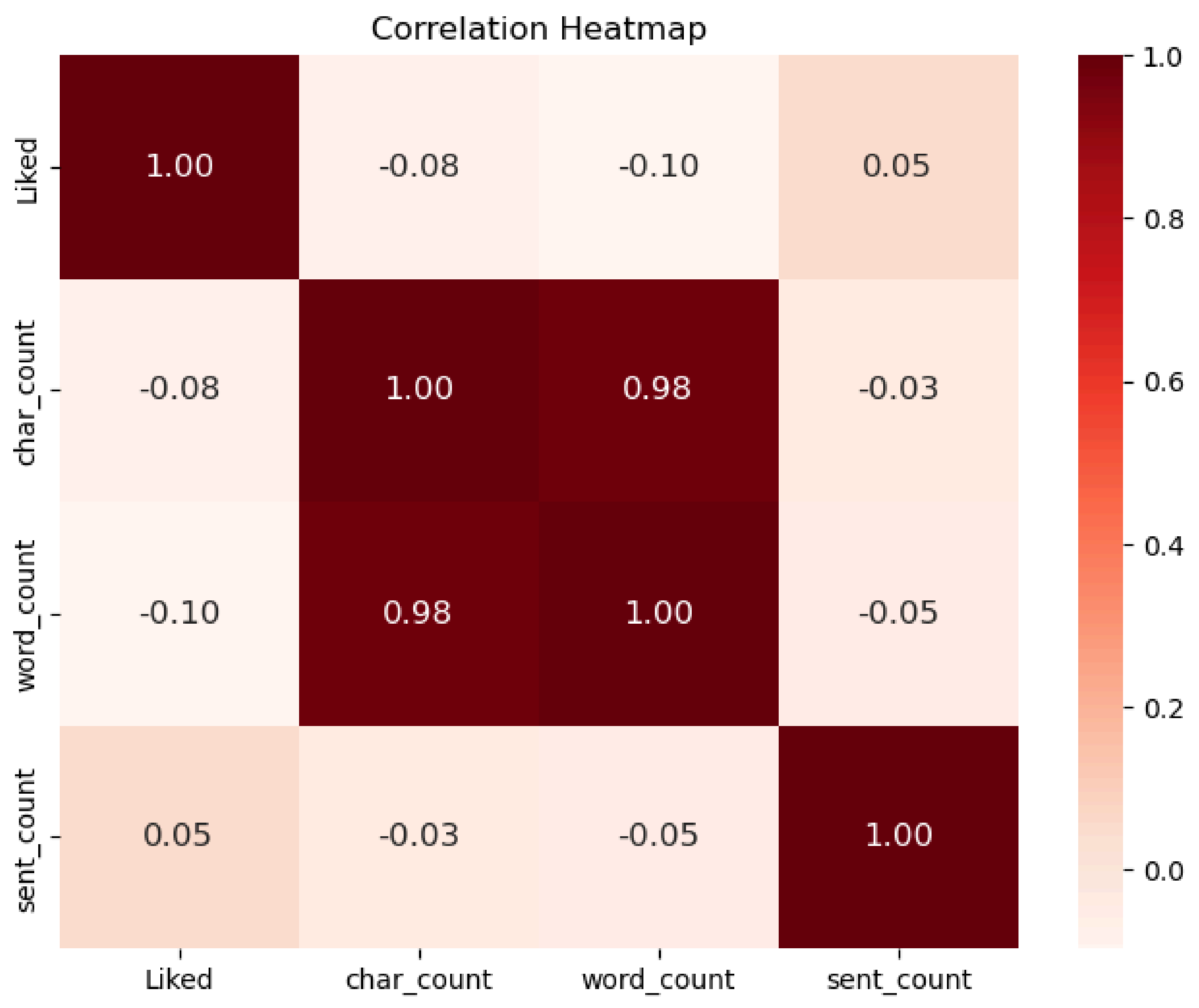
In [53]:
```python
# Create a bar plot for the count of liked and disliked reviews
plt.figure(figsize=(8, 6))
data['Liked'].value_counts().plot(kind='bar', color=['blue', 'orange'])
plt.title('Count of Liked and Disliked Reviews')
plt.xlabel('Liked')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Disliked', 'Liked'], rotation=0)
plt.show()
```



Heatmap

```
In [54]: # Compute the correlation matrix
         corr = data[['Liked', 'char_count', 'word_count', 'sent_count']].corr()

         # Create a heatmap
         plt.figure(figsize=(8, 6))
         sns.heatmap(corr, annot=True, cmap='Reds', fmt=".2f", annot_kws={"size": 12})
         plt.title('Correlation Heatmap')
         plt.show()
```



Vectorization

```
In [55]: from sklearn.feature_extraction.text import CountVectorizer
```

```
In [56]: cv = CountVectorizer(max_features=1500)
```

```
In [57]: X = cv.fit_transform(corpus).toarray()
```

```
In [58]: X
```

```
Out[58]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
In [59]: X.shape
```

```
Out[59]: (1000, 1500)
```

```
In [60]: joblib.dump(cv,"count_v_res")
```

```
Out[60]: ['count_v_res']
```

```
In [61]: y = data['Liked']
In [62]: y
```

```
Out[62]: 0      1
         1      0
         2      0
         3      1
         4      1
               ..
         995    0
         996    0
         997    0
         998    0
         999    0
         Name: Liked, Length: 1000, dtype: int64
```

Spliting the dataset for training and testing purpose

```
In [63]: from sklearn.model_selection import train_test_split
```

```
In [64]: X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.20,random_state=42)
```

Classification using various models

1.Naive Bayes

```
In [65]: from sklearn.naive_bayes import GaussianNB
```

```
In [66]: nb =GaussianNB()
         nb.fit(X_train,y_train)
         y_pred = nb.predict(X_test)
```

```
In [67]: from sklearn.metrics import accuracy_score, confusion_matrix
```

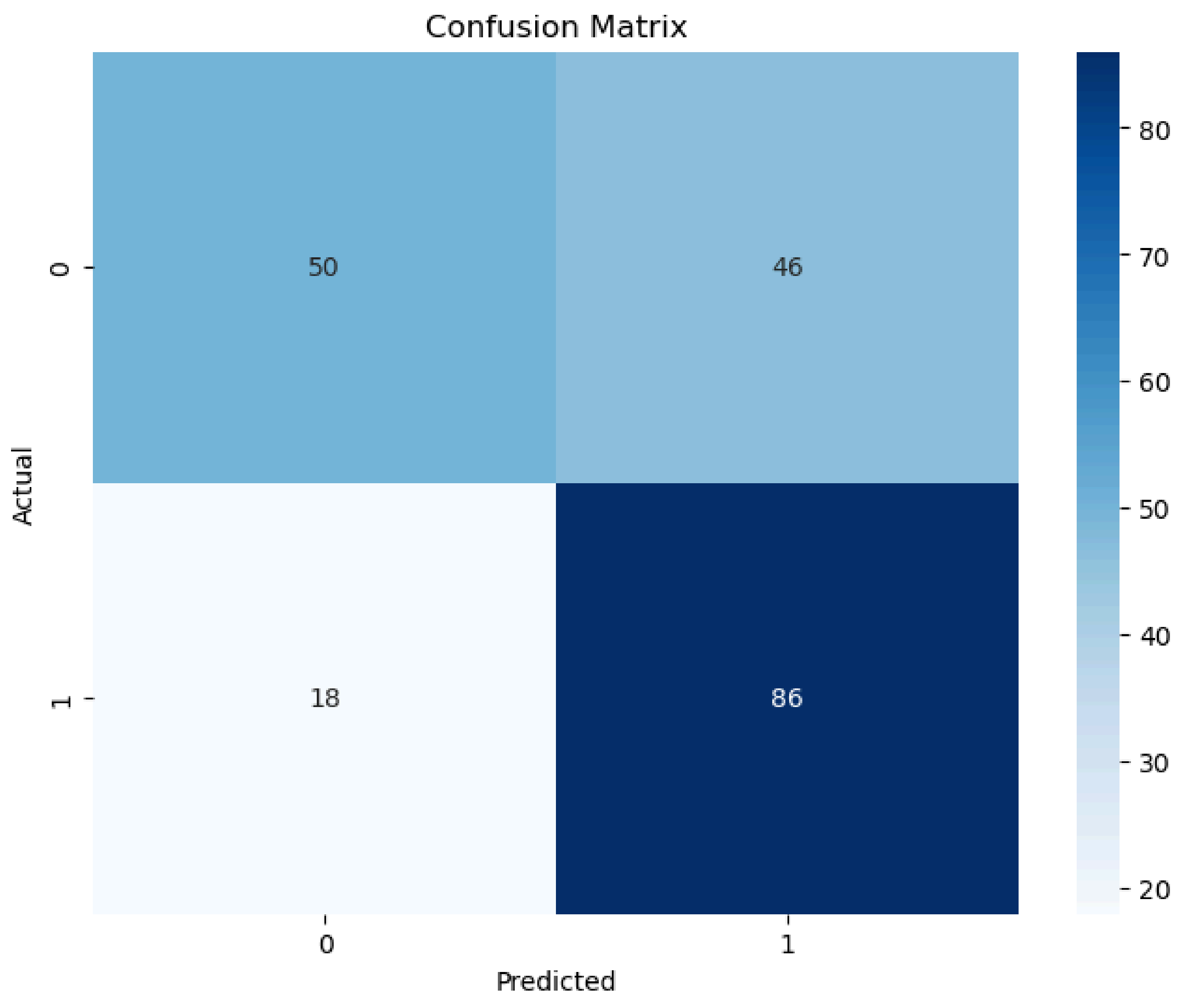
```
In [68]: accuracy_score(y_test,y_pred)
```

```
Out[68]: 0.68
```

Confusion Matrix

In [69]:
```
#create confusion matrix
cm=confusion_matrix(y_test,y_pred)

#plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm,annot=True,fmt="d",cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [70]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
score1=accuracy_score(y_test,y_pred)
score2=precision_score(y_test,y_pred)
score3=recall_score(y_test,y_pred)
print("\n")
print("Accuracy is ",round(score1*100,2),"%")
print("Precision is ",round(score2,2))
print("Recall is ",round(score3,2))
```

```


Accuracy is  68.0 %
Precision is  0.65
Recall is  0.83
```

2.Logistic Regression

In [71]:
```
from sklearn.linear_model import LogisticRegression
lr =LogisticRegression()
lr.fit(X_train,y_train)
```

Out[71]:
```
▾ LogisticRegression
LogisticRegression()
```

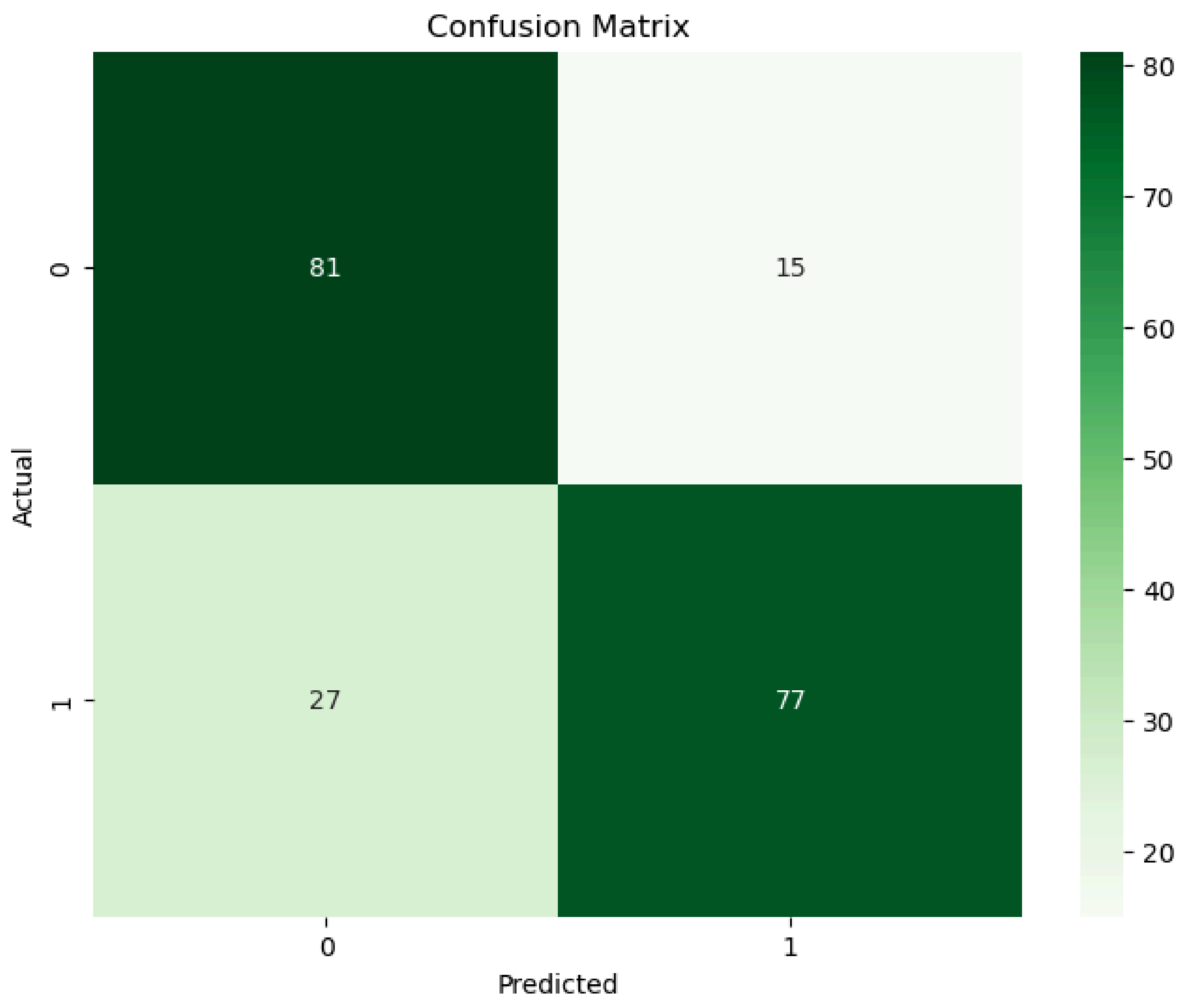
In [72]:
```
y_pred=lr.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[72]:
```
0.79
```

Confusion Matrix

In [73]:
```
#create confusion matrix
cm=confusion_matrix(y_test,y_pred)

#plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm,annot=True,fmt="d",cmap='Greens')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [74]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
score1=accuracy_score(y_test,y_pred)
score2=precision_score(y_test,y_pred)
score3=recall_score(y_test,y_pred)
print("\n")
print("Accuracy is ",round(score1*100,2),"%")
print("Precision is ",round(score2,2))
print("Recall is ",round(score3,2))
```

```


Accuracy is  79.0 %
Precision is  0.84
Recall is  0.74
```

3.Random Forest Classifier

In [75]:
```
from sklearn.ensemble import RandomForestClassifier
rf =RandomForestClassifier()
rf.fit(X_train,y_train)
```

Out[75]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

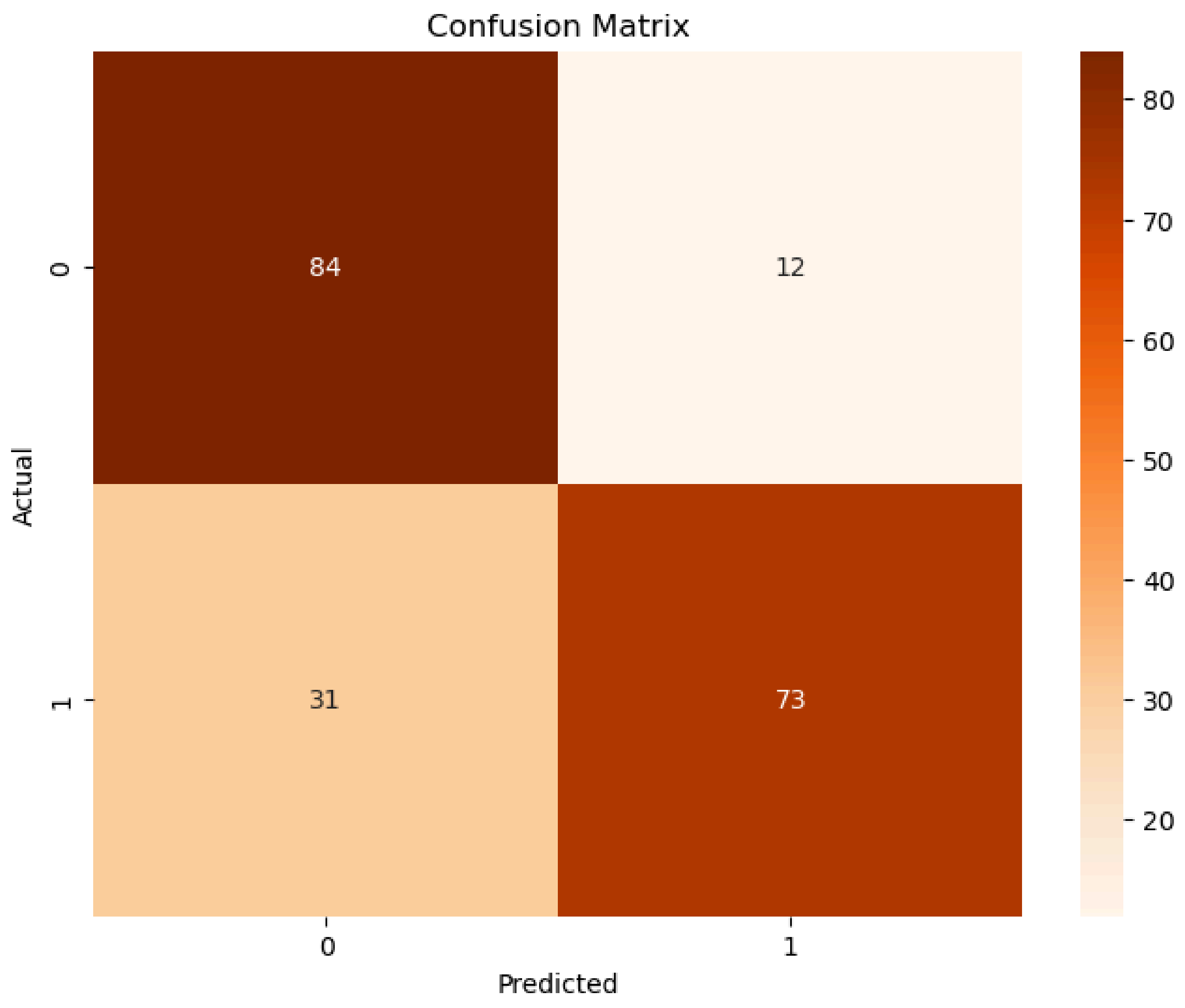
```
In [76]: y_pred = rf.predict(X_test)
         accuracy_score(y_test,y_pred)

Out[76]: 0.785
```

Confusion Matrix

```
In [77]: #create confusion matrix
         cm=confusion_matrix(y_test,y_pred)

         #plot confusion matrix
         plt.figure(figsize=(8, 6))
         sns.heatmap(cm,annot=True,fmt="d",cmap='Oranges')
         plt.title('Confusion Matrix')
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.show()
```



```
In [78]: from sklearn.metrics import precision_score
         from sklearn.metrics import recall_score
         score1=accuracy_score(y_test,y_pred)
         score2=precision_score(y_test,y_pred)
         score3=recall_score(y_test,y_pred)
         print("\n")
         print("Accuracy is ",round(score1*100,2),"%")
         print("Precision is ",round(score2,2))
         print("Recall is ",round(score3,2))


         Accuracy is  78.5 %
         Precision is  0.86
         Recall is  0.7

In [79]: import joblib

In [80]: joblib.dump(rf,'Restaurant_review_model')

Out[80]: ['Restaurant_review_model']
```

In [1]:
```python
import tkinter as tk
from tkinter import ttk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
import joblib
import re

class RestaurantReviewApp:
    def __init__(self, master):
        self.master = master
        master.title("Restaurant Review Classification App")
        master.geometry("400x300")  # Set a custom size for the window

        # Load your pre-trained Random Forest model and CountVectorizer
        # Replace 'your_model.pkl' and 'your_vectorizer.pkl' with the actual filenames
        self.model = joblib.load('Restaurant_review_model')
        self.vectorizer = joblib.load('count_v_res')

        # Create and set up widgets
        title_font = ('Helvetica', 16, 'bold')  # Larger font for the title
        self.label = ttk.Label(master, text="Enter your restaurant review:", font=title_font)
        self.label.pack(pady=10)

        self.text_entry = tk.Text(master, height=5, width=40)
        self.text_entry.pack(pady=10)

        # Increase button size and change color on press
        self.classify_button = ttk.Button(master, text="Classify", command=self.classify_review, style='Cu
        self.classify_button.pack(pady=10)

        self.result_label = ttk.Label(master, text="")
        self.result_label.pack(pady=10)

        # Style configuration for the button
        self.style = ttk.Style()
        self.style.configure('Custom.TButton', font=('Helvetica', 12), width=15, foreground='black', backg
        self.style.map('Custom.TButton', foreground=[('pressed', 'black'), ('active', 'white')], backgrour

    def preprocess_text(self, text):
        custom_stopwords = {'don', "don't", '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', "must
                           'needn', "needn't", 'shan', "shan't", 'no', 'nor', 'not', 'shouldn', "shouldn'
                           'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"}
        ps = PorterStemmer()
        stop_words = set(stopwords.words("english")) - custom_stopwords

        review = re.sub('[^a-zA-Z]', ' ', text)
        review = review.lower()
        review = review.split()
        review = [ps.stem(word) for word in review if word not in stop_words]
        review = " ".join(review)

        return review

    def classify_review(self):
        user_input = self.text_entry.get("1.0", "end-1c")
        if user_input:
            processed_input = self.preprocess_text(user_input)
            # Transform the processed_input using the CountVectorizer
            processed_input_vectorized = self.vectorizer.transform([processed_input])
            prediction = self.model.predict(processed_input_vectorized)[0]
            sentiment = "Positive" if prediction == 1 else "Negative"
            self.result_label.config(text=f"Predicted Sentiment: {sentiment}")
        else:
            self.result_label.config(text="Please enter a review before clicking 'Classify'.")

if __name__ == "__main__":
    root = tk.Tk()
    app = RestaurantReviewApp(root)
    root.mainloop()
```

```
In [ ]:
```