hotel-review-sentimental-analysis

April 2, 2024

1 Exploring Guest Sentiments at Hotel Pullman Paris Tour Eiffel: A Comprehensive Sentiment Analysis of European Hotel Reviews

1.1 Context

In this project, we aim to delve into the sentiments expressed by guests in their reviews of the renowned Hotel Pullman Paris Tour Eiffel, situated in Europe. Leveraging the power of sentiment analysis, we seek to understand the overarching emotions and opinions conveyed by guests through their feedback. By analyzing the textual content of the reviews, we aim to uncover patterns, trends, and key insights that can provide valuable information to the hotel management for enhancing guest satisfaction and improving overall guest experience. This study not only contributes to the understanding of guest sentiments but also provides actionable recommendations for the hospitality industry, with a focus on Hotel Pullman Paris Tour Eiffel.

[1]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from summarytools import dfSummary
```

[2]:
```
# Importing Dataset
Data = pd.read_csv(r"C:\Users\ABC\Downloads\hotel_reviews.csv")
Data.sample(5)
```

[2]:

	Review	Rating
17245	malaria shmalaria mass confusion staff arrived…	4
1254	need street wise, stayed 2 weeks 35 night dbl …	2
14280	sylish great value heart gothic quarter just r…	5
18573	great hotel, just returned majestic colonial, …	5
14073	perfect location ideal exploring foot just day…	4

1.2 About Dataset

Hotels play a crucial role in traveling and with the increased access to information new pathways of selecting the best ones emerged. With this dataset, consisting of 20k reviews crawled from Tripadvisor, you can explore what makes a great hotel and maybe even use this model in your travels!

[3]:
```
dfSummary(Data)
```

[3]:
```
<pandas.io.formats.style.Styler at 0x1b732e7e450>
```

[4]:
```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20491 entries, 0 to 20490
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  20491 non-null  object
 1   Rating  20491 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 320.3+ KB
```

[5]:
```
# Checking null values
Data.isna().sum()
```

[5]:
```
Review    0
Rating    0
dtype: int64
```

[6]:
```
Data.describe().round(2)
```

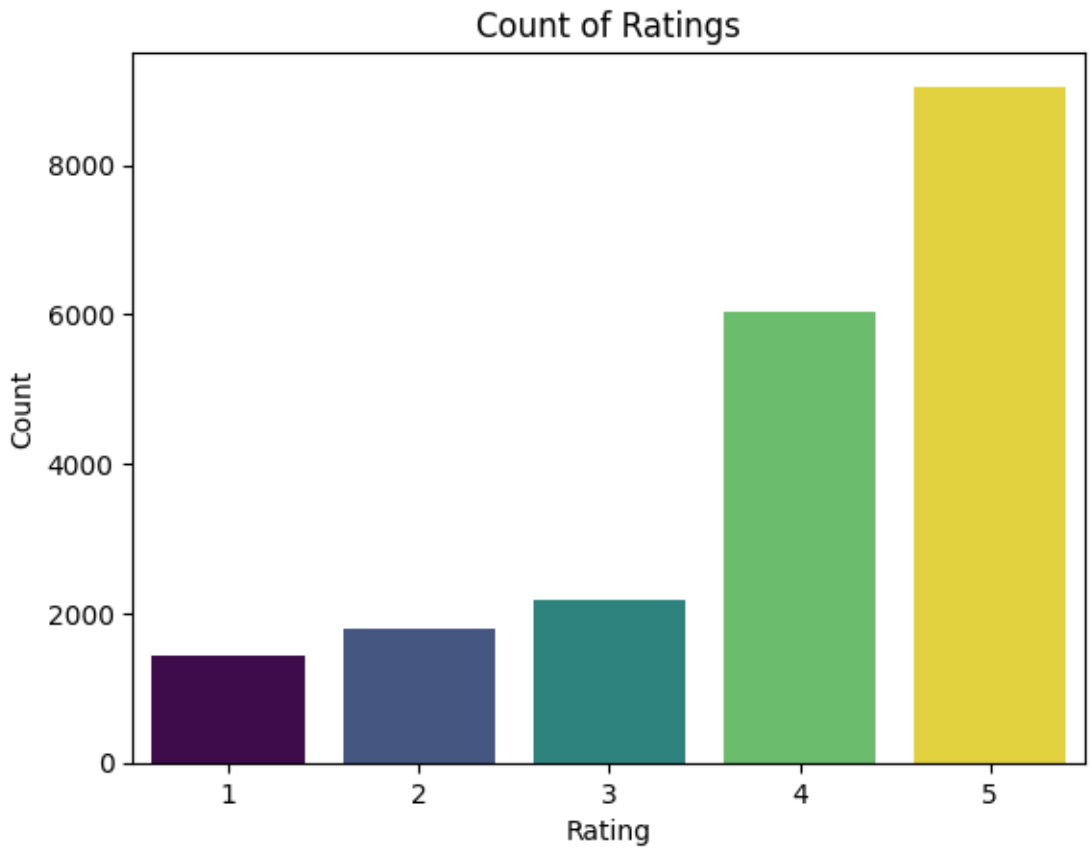
[6]:
```
          Rating
count  20491.00
mean       3.95
std        1.23
min        1.00
25%        3.00
50%        4.00
75%        5.00
max        5.00
```

[7]:
```
# Checking for any duplicate rows
Data.duplicated().sum()
```

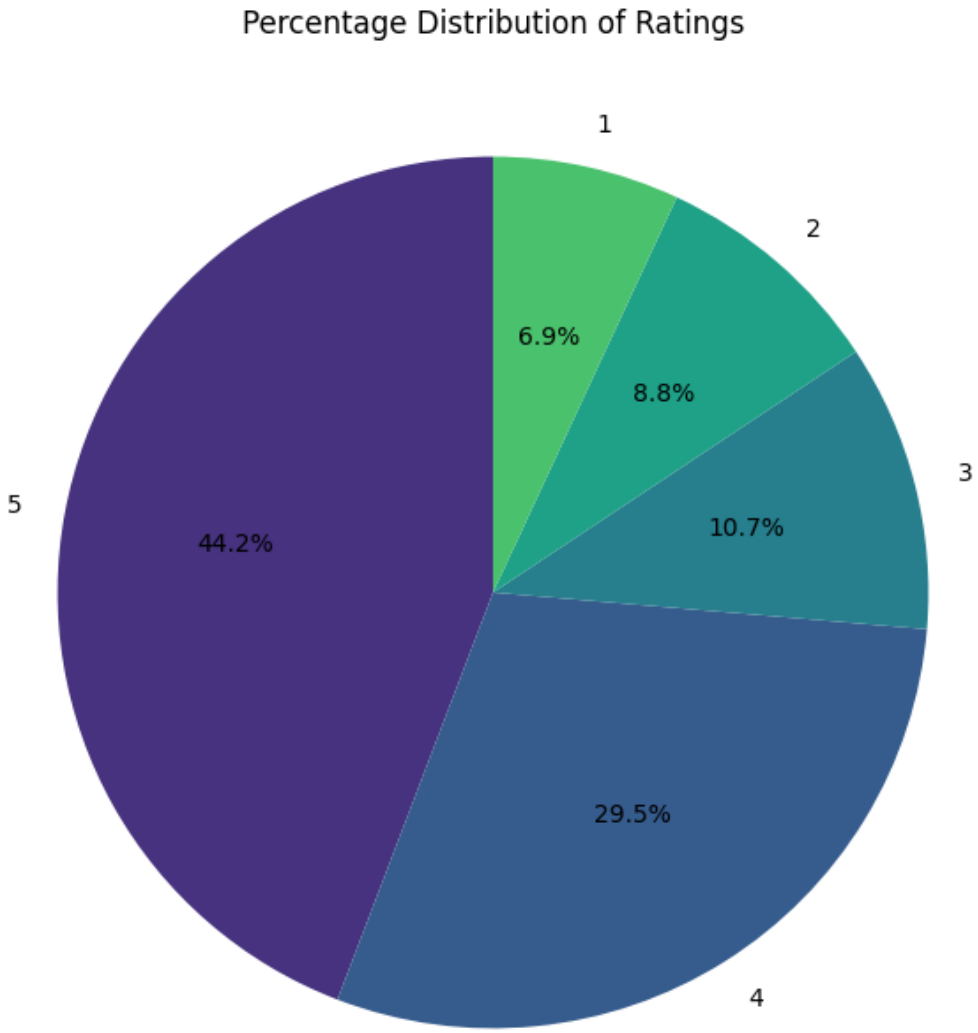
[7]:
```
0
```

1.3 Visualizing Trends & Patterns

[8]:
```
sns.countplot(x='Rating', data=Data, hue='Rating', palette='viridis',␣
 ↪legend=False)
plt.title('Count of Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



[9]:
```
# Calculating the percentage distribution
rating_counts = Data['Rating'].value_counts(normalize=True) * 100

# Plotting the pie chart with seaborn
plt.figure(figsize=(8, 8))
sns.set_palette("viridis")
plt.pie(rating_counts, labels=rating_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90)
plt.title('Percentage Distribution of Ratings')
plt.show()
```

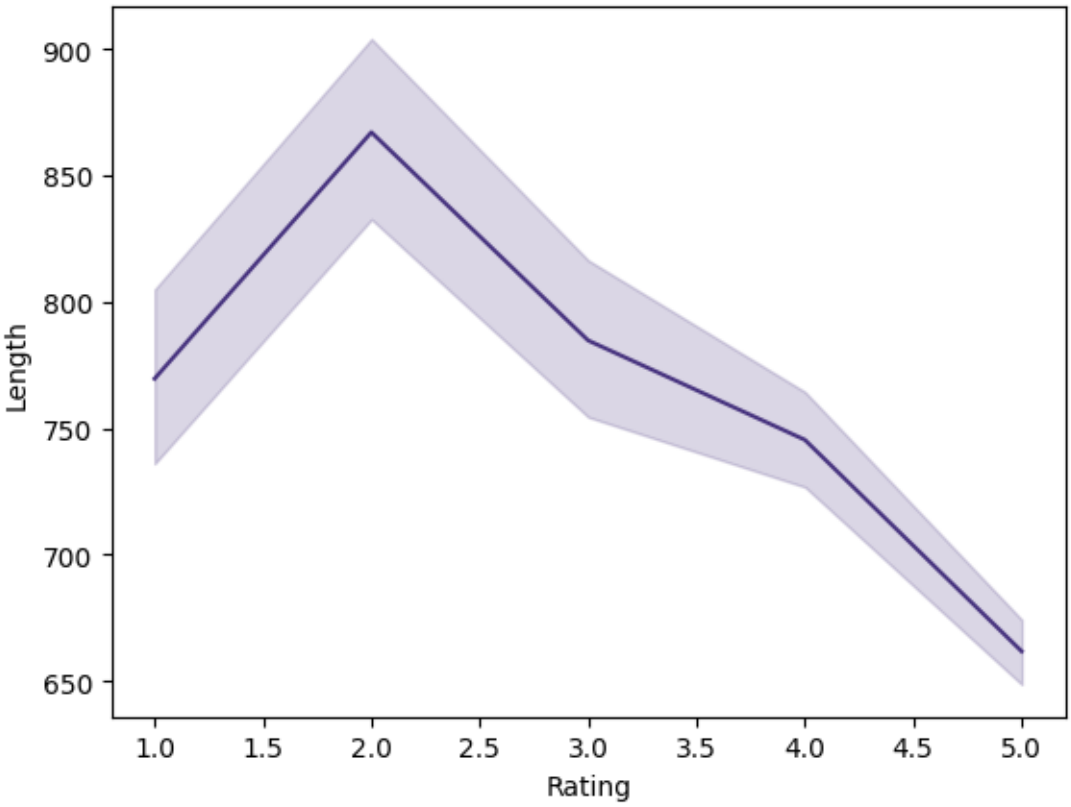


[10]:
```
# Adding a new column Length of word in sentence
Data['Length'] = Data['Review'].apply(len)
Data.head()
```

[10]:

	Review	Rating	Length
0	nice hotel expensive parking got good deal sta…	4	593
1	ok nothing special charge diamond member hilto…	2	1689
2	nice rooms not 4* experience hotel monaco seat…	3	1427
3	unique, great stay, wonderful time hotel monac…	5	600
4	great stay great stay, went seahawk game aweso…	5	1281

[11]:
```
# Plotting the relationship between Rating and Length
sns.set_palette("viridis")
sns.lineplot(data = Data,x = 'Rating',y = 'Length')
plt.show()
```



[12]:
```
# Extracting the rows with rating 5
Data_5 = Data[Data['Rating']==5]
Data_5.head()
```

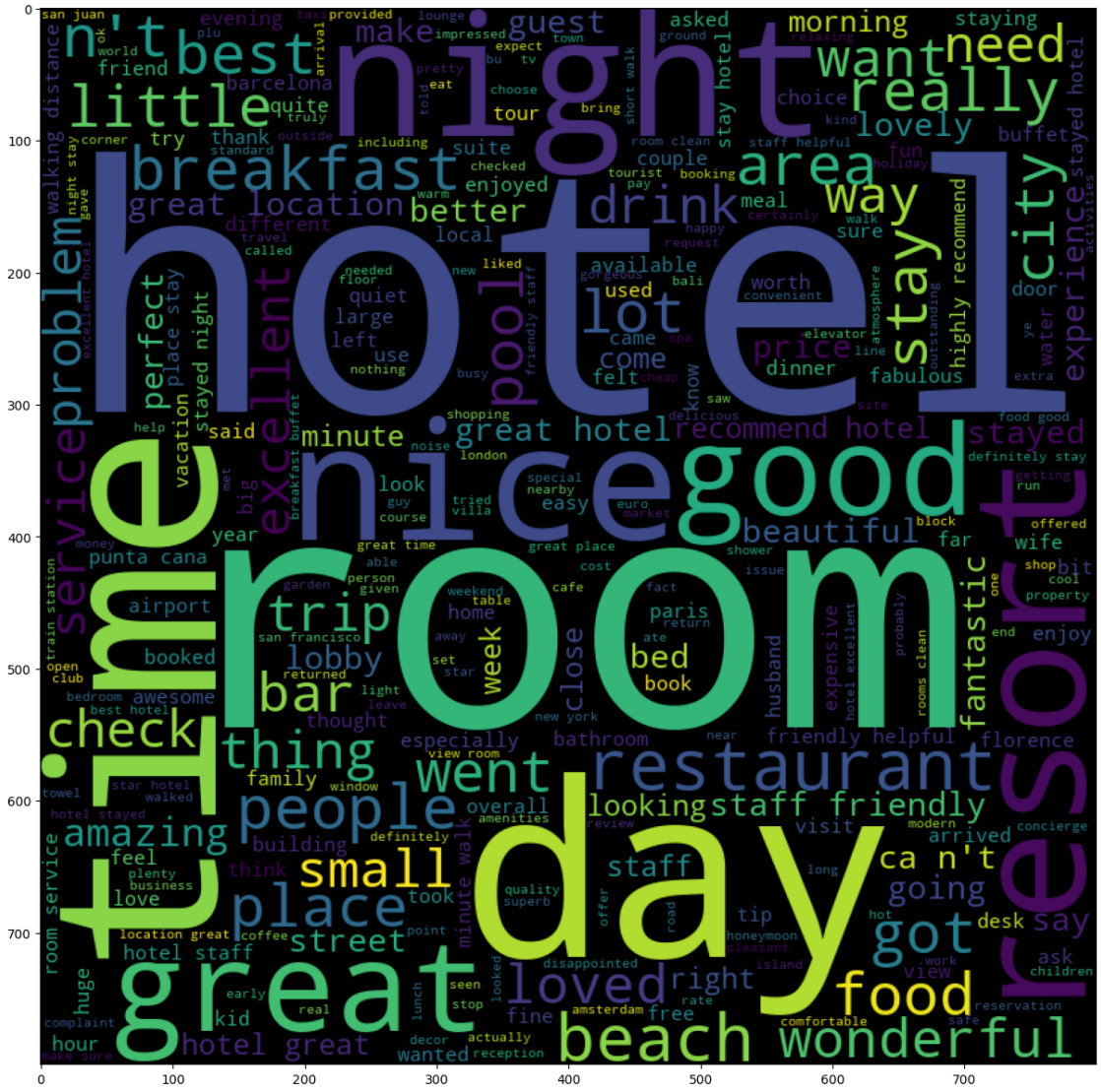
[12]:

	Review	Rating	Length
3	unique, great stay, wonderful time hotel monac…	5	600
4	great stay great stay, went seahawk game aweso…	5	1281
5	love monaco staff husband stayed hotel crazy w…	5	1002
6	cozy stay rainy city, husband spent 7 nights m…	5	748
8	hotel stayed hotel monaco cruise, rooms genero…	5	419

[13]:
```
# Visualizing most common words from the reviews with rating 5
plt.figure(figsize=(15,15))
wc1 = WordCloud(max_words=1000, min_font_size=10,␣
 ↪height=800,width=800,background_color="black").generate(' '.
 ↪join(Data_5['Review']))
```

```
plt.imshow(wc1)
```

[13]: <matplotlib.image.AxesImage at 0x1b732de7080>



[14]:
```
# Extracting the rows with rating 4
Data_4 = Data[Data['Rating']==4]
Data_4.head()
```

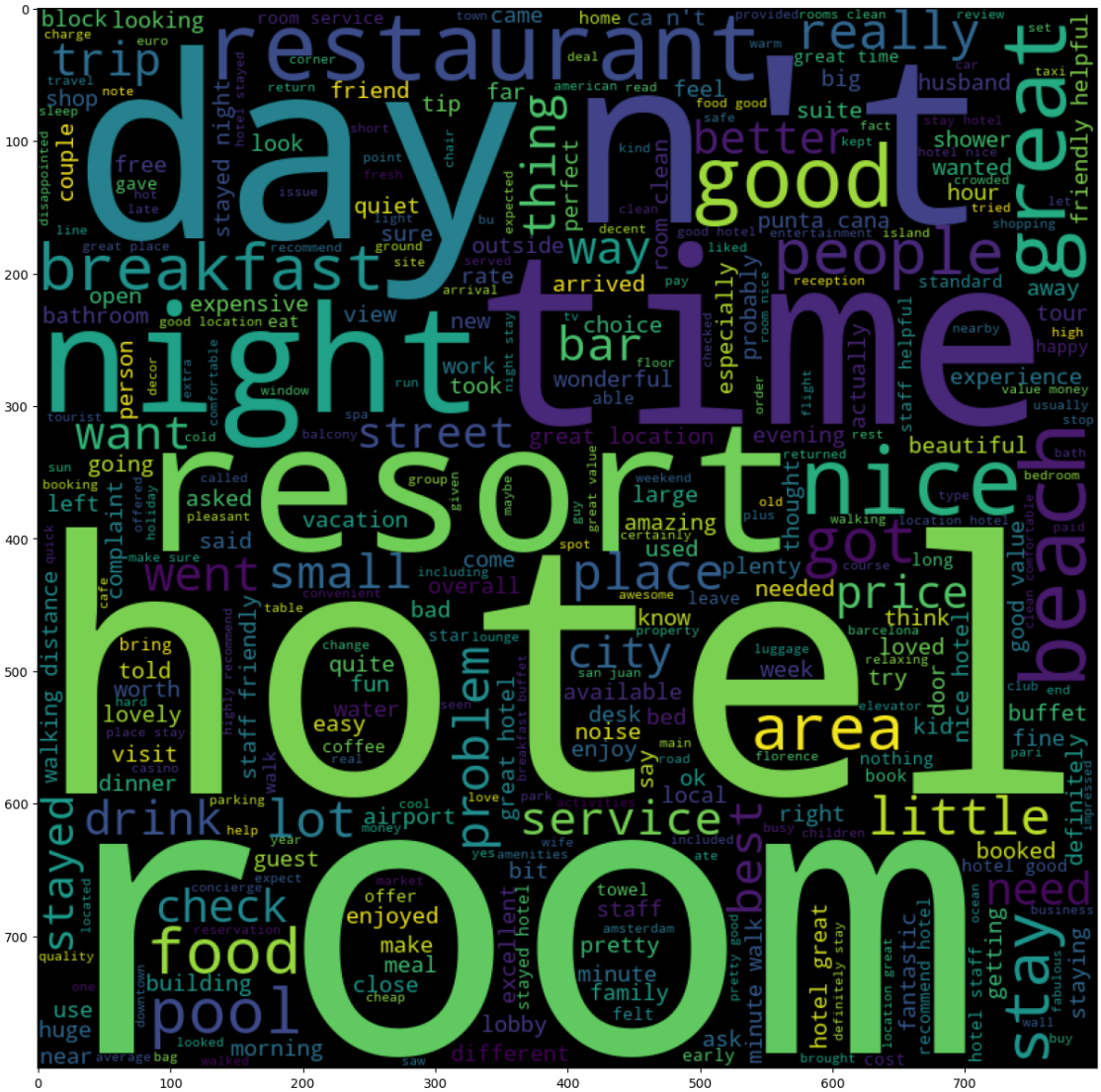
[14]:

	Review	Rating	Length
0	nice hotel expensive parking got good deal sta…	4	593
7	excellent staff, housekeeping quality hotel ch…	4	597
11	nice value seattle stayed 4 nights late 2007. …	4	364
12	nice hotel good location hotel kimpton design …	4	569

```
14  great hotel night quick business trip, loved l…       4      202
```

[15]:
```
# Visualizing most common words from the reviews with rating 4
plt.figure(figsize=(15,15))
wc2 = WordCloud(max_words=1000, min_font_size=10,␣
 ↪height=800,width=800,background_color="black").generate(' '.
 ↪join(Data_4['Review']))
plt.imshow(wc2)
```

[15]:
```
<matplotlib.image.AxesImage at 0x1b732dc2cf0>
```



[16]:
```
# Extracting the rows with rating 3
Data_3 = Data[Data['Rating']==3]
```

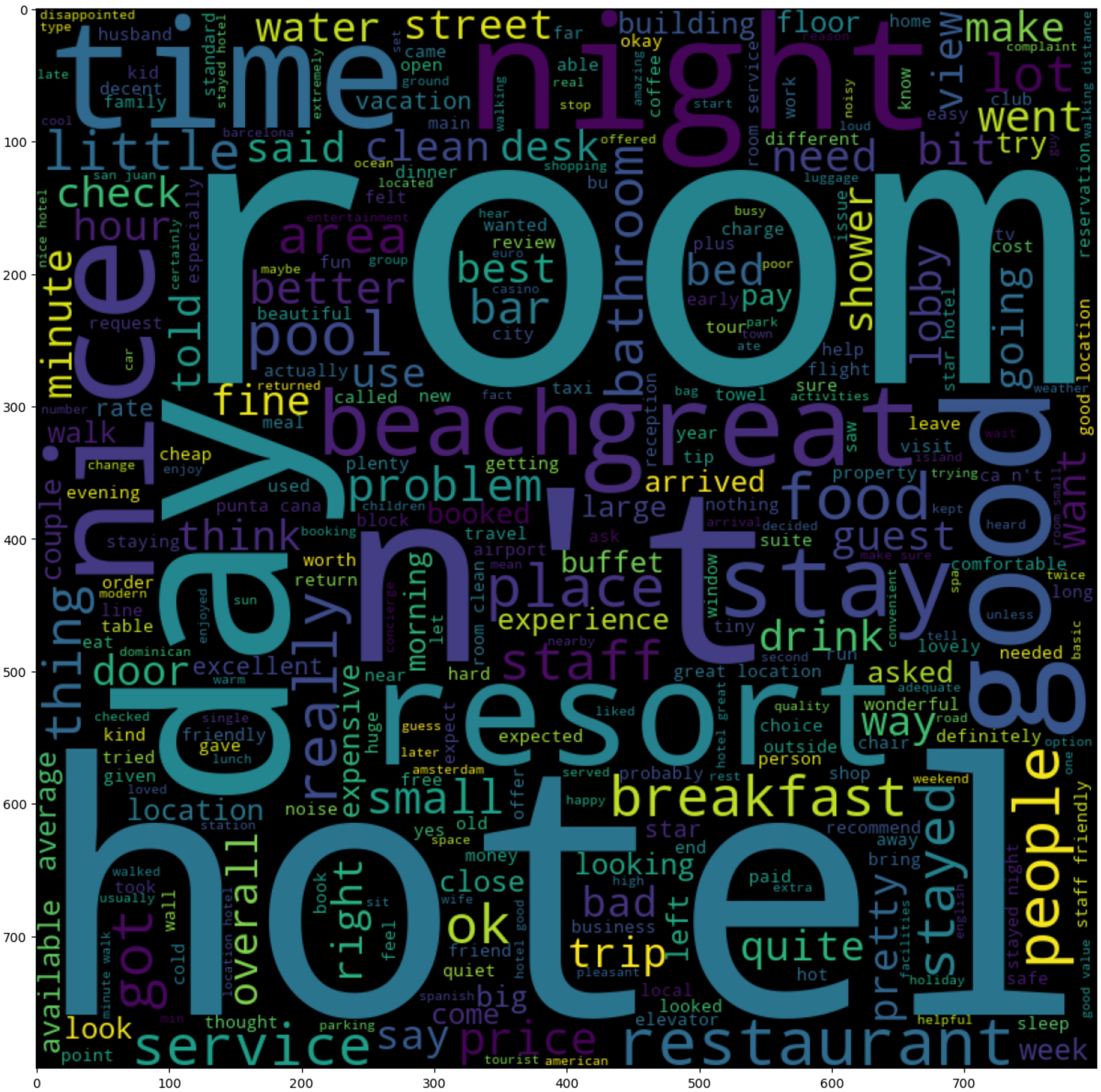
```
Data_3.head()
```

[16]:

	Review	Rating	Length
2	nice rooms not 4* experience hotel monaco seat…	3	1427
13	nice hotel not nice staff hotel lovely staff q…	3	417
19	hmmmmm say really high hopes hotel monaco chos…	3	1025
25	n't mind noise place great, read reviews noise…	3	482
27	met expectations centrally located hotel block…	3	538

[17]:
```
# Visualizing most common words from the reviews with rating 3
plt.figure(figsize=(15,15))
wc3 = WordCloud(max_words=1000, min_font_size=10,␣
 ↪height=800,width=800,background_color="black").generate(' '.
 ↪join(Data_3['Review']))
plt.imshow(wc3)
```

[17]:
```
<matplotlib.image.AxesImage at 0x1b732d94410>
```




```
[18]: # Extracting the rows with rating 2
      Data_2 = Data[Data['Rating']==2]
      Data_2.head()
```

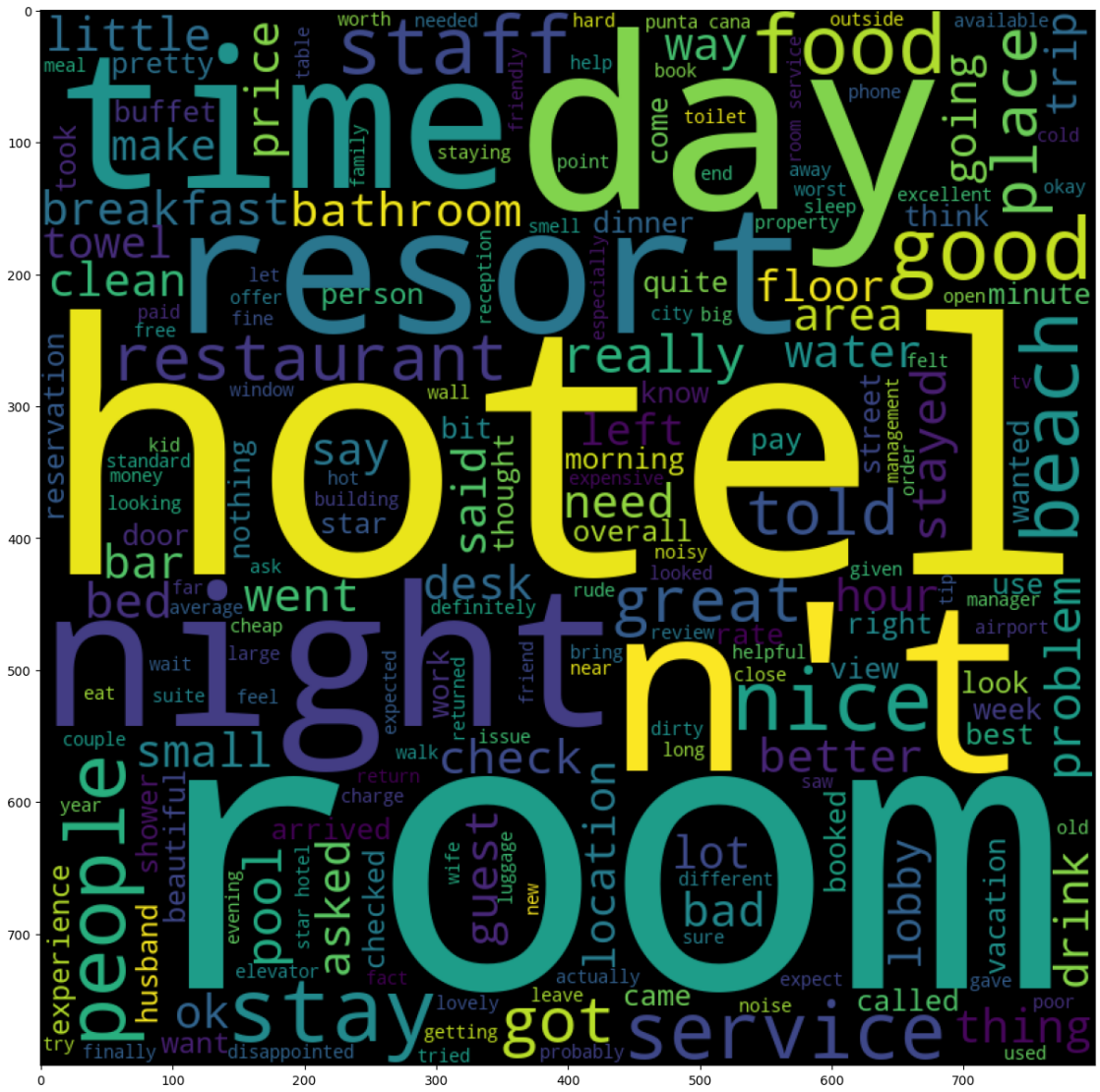
[18]:

	Review	Rating	Length
1	ok nothing special charge diamond member hilto...	2	1689
10	poor value stayed monaco seattle july, nice ho...	2	333
16	disappointed say anticipating stay hotel monac...	2	1662
24	great location need internally upgrade advanta...	2	250
33	good location poor cleanliness warwick hotel g...	2	576

```
[19]: # Visualizing most common words from the reviews with rating 2
      plt.figure(figsize=(15,15))
```

```
wc4 = WordCloud(max_words=1000, min_font_size=10,␣
 ↪height=800,width=800,background_color="black").generate(' '.
 ↪join(Data_2['Review']))
plt.imshow(wc4)
```

[19]: <matplotlib.image.AxesImage at 0x1b732de14f0>



[20]:
```
# Extracting the rows with rating 1
Data_1 = Data[Data['Rating']==1]
Data_1.head()
```

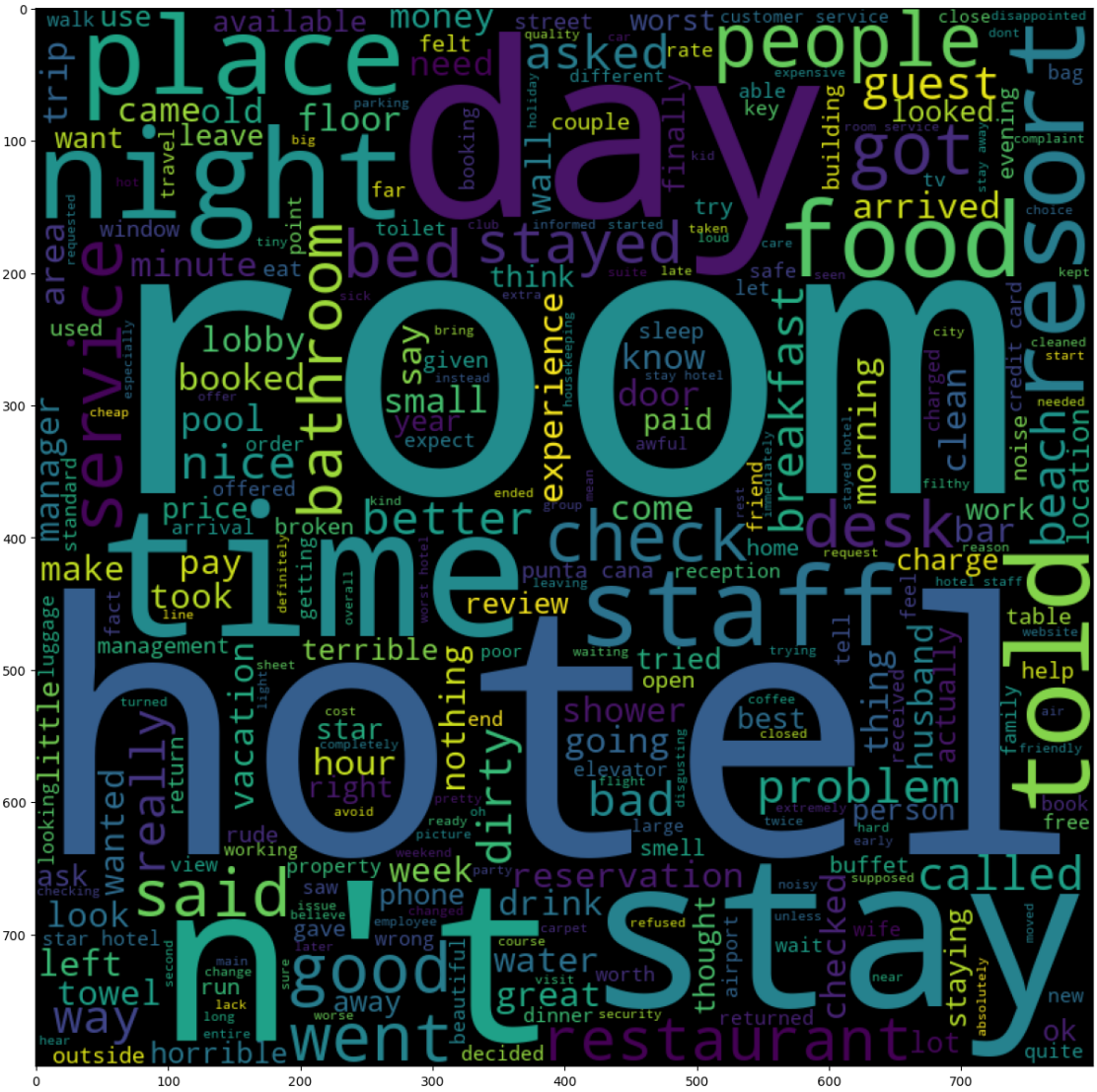
[20]:

	Review	Rating	Length
15	horrible customer service hotel stay february …	1	1411

32	noise airconditioner-a standard, arranged stay…	1	614
40	bad choice, booked hotel hot wire called immed…	1	861
65	hated inn terrible, room-service horrible staf…	1	133
69	ace grunge lives does mold mildew tiny bed met…	1	84

[21]:
```
# Visualizing most common words from the reviews with rating 1
plt.figure(figsize=(15,15))
wc5 = WordCloud(max_words=1000, min_font_size=10,␣
↪height=800,width=800,background_color="black").generate(' '.
↪join(Data_1['Review']))
plt.imshow(wc5)
```

[21]: <matplotlib.image.AxesImage at 0x1b732d3e5a0>



1.4 Making text clean for ML

[22]:
```
Data.head()
```

[22]:

	Review	Rating	Length
0	nice hotel expensive parking got good deal sta…	4	593
1	ok nothing special charge diamond member hilto…	2	1689
2	nice rooms not 4* experience hotel monaco seat…	3	1427
3	unique, great stay, wonderful time hotel monac…	5	600
4	great stay great stay, went seahawk game aweso…	5	1281

[23]:
```
#Importing necessary libraries from NLTK
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
```

[24]:
```
# downloading stopwords
nltk.download('stopwords')
```

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

[24]:
```
True
```

[25]:
```
sw = set(stopwords.words('english'))
sw
```

[25]:
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
```

```
'been',
'before',
'being',
'below',
'between',
'both',
'but',
'by',
'can',
'couldn',
"couldn't",
'd',
'did',
'didn',
"didn't",
'do',
'does',
'doesn',
"doesn't",
'doing',
'don',
"don't",
'down',
'during',
'each',
'few',
'for',
'from',
'further',
'had',
'hadn',
"hadn't",
'has',
'hasn',
"hasn't",
'have',
'haven',
"haven't",
'having',
'he',
'her',
'here',
'hers',
'herself',
'him',
'himself',
'his',
```

```
'how',
'i',
'if',
'in',
'into',
'is',
'isn',
"isn't",
'it',
"it's",
'its',
'itself',
'just',
'll',
'm',
'ma',
'me',
'mightn',
"mightn't",
'more',
'most',
'mustn',
"mustn't",
'my',
'myself',
'needn',
"needn't",
'no',
'nor',
'not',
'now',
'o',
'of',
'off',
'on',
'once',
'only',
'or',
'other',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
're',
's',
```

```
'same',
'shan',
"shan't",
'she',
"she's",
'should',
"should've",
'shouldn',
"shouldn't",
'so',
'some',
'such',
't',
'than',
'that',
"that'll",
'the',
'their',
'theirs',
'them',
'themselves',
'then',
'there',
'these',
'they',
'this',
'those',
'through',
'to',
'too',
'under',
'until',
'up',
've',
'very',
'was',
'wasn',
"wasn't",
'we',
'were',
'weren',
"weren't",
'what',
'when',
'where',
'which',
'while',
```

```
 'who',
 'whom',
 'why',
 'will',
 'with',
 'won',
 "won't",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

These are some of the common English stopwords that NLTK provides. They are often removed from text data during preprocessing to focus on the more meaningful words for analysis.

[26]:
```
# Defining a function for cleaning all the reviews
def text_preprocessing(a):
  a=re.sub('[^a-zA-Z0-9]',' ',a)
  a=a.lower().split()
  ps=PorterStemmer()
  clean_word=[ps.stem(i) for i in a if not i in sw]
  sen=' '.join(clean_word)
  return sen
```

[27]:
```
Data['Cleaned_Review'] = Data['Review'].apply(text_preprocessing)
Data.head()
```

[27]:

	Review	Rating	Length	Cleaned_Review
0	nice hotel expensive parking got good deal sta…	4	593	nice hotel expens park got good deal stay hote…
1	ok nothing special charge diamond member hilto…	2	1689	ok noth special charg diamond member hilton de…
2	nice rooms not 4* experience hotel monaco seat…	3	1427	nice room 4 experi hotel monaco seattl good ho…
3	unique, great stay, wonderful time hotel monac…	5	600	uniqu great stay wonder time hotel monaco loca…
4	great stay great stay, went seahawk game aweso…	5	1281	

```
4  great stay great stay went seahawk game awesom…
```

[28]:
```
# Adding a new column named 'Length2' that will denote the length of cleaned review
Data['Length2'] = Data['Cleaned_Review'].apply(len)
Data.head()
```

[28]:

	Review	Rating	Length	Cleaned_Review	Length2
0	nice hotel expensive parking got good deal sta…	4	593	nice hotel expens park got good deal stay hote…	484
1	ok nothing special charge diamond member hilto…	2	1689	ok noth special charg diamond member hilton de…	1428
2	nice rooms not 4* experience hotel monaco seat…	3	1427	nice room 4 experi hotel monaco seattl good ho…	1216
3	unique, great stay, wonderful time hotel monac…	5	600	uniqu great stay wonder time hotel monaco loca…	508
4	great stay great stay, went seahawk game aweso…	5	1281	great stay great stay went seahawk game awesom…	1062

[29]:
```
Data.describe().round(2)
```

[29]:

	Rating	Length	Length2
count	20491.00	20491.0	20491.00
mean	3.95	724.9	606.51
std	1.23	689.1	573.95
min	1.00	44.0	31.00
25%	3.00	339.0	284.50
50%	4.00	537.0	450.00
75%	5.00	859.0	721.00
max	5.00	13501.0	11302.00

1.5 Comparative Performance of Different Machine Learning Models on Rating Prediction

[67]:
```
#Importing necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import  KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import svm
```

[57]:
```
cv = CountVectorizer()

# Fit and transform the text data
X = cv.fit_transform(Data["Cleaned_Review"])

# Target variable
Y = Data["Rating"]

# Train-Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)
#for storing accuracies
accuracy_dict ={}
```

1.5.1 Logistic Regression

[62]:
```
# Initialize LogisticRegression model
model = LogisticRegression(max_iter=1000)

# Train the model
model.fit(X_train, Y_train)

# Predict on the test set
Y_pred = model.predict(X_test)

# Calculate accuracy
accuracy1 = accuracy_score(Y_test, Y_pred)

#storing accuracy
accuracy_dict["LogisticRegression"] = accuracy1
print("Accuracies:", accuracy1)
```

```
Accuracies: 0.5869724323005611
```

1.5.2 Random Forest Classifier

[64]:
```
# Initialize RandomForestClassifier model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, Y_train)

# Predict on the test set
Y_pred = rf_model.predict(X_test)

# Calculate accuracy
accuracy2 = accuracy_score(Y_test, Y_pred)
```

```
# Store accuracy
accuracy_dict["RandomForestClassifier"] = accuracy2

print("Accuracies:", accuracy2)
```

```
Accuracies: 0.5164674310807514
```

1.5.3 Decision Tree Classifier

[65]:
```
# Initialize DecisionTreeClassifier model
dt_model = DecisionTreeClassifier(random_state=42)

# Train the model
dt_model.fit(X_train, Y_train)

# Predict on the test set
Y_pred = dt_model.predict(X_test)

# Calculate accuracy
accuracy3 = accuracy_score(Y_test, Y_pred)

# Store accuracy in the dictionary
accuracy_dict["DecisionTreeClassifier"] = accuracy3

print("Accuracies:", accuracy3)
```

```
Accuracies: 0.4627958038545987
```

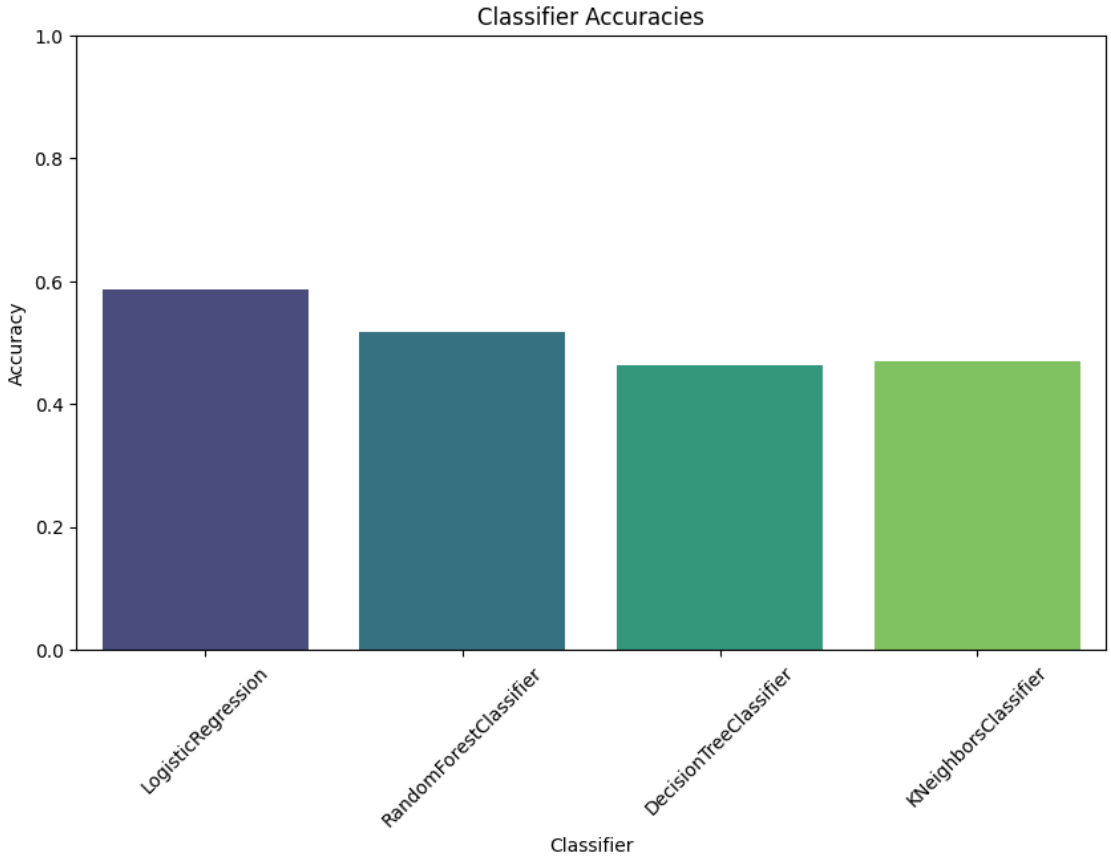
1.5.4 KNeighbors Classifier

[68]:
```
# Initialize KNeighborsClassifier model
knn_model = KNeighborsClassifier()

# Train the model
knn_model.fit(X_train, Y_train)

# Predict on the test set
Y_pred = knn_model.predict(X_test)

# Calculate accuracy
accuracy4 = accuracy_score(Y_test, Y_pred)

# Store accuracy in the dictionary
accuracy_dict["KNeighborsClassifier"] = accuracy4

print("Accuracies:", accuracy4)
```

```
Accuracies: 0.46987070017077337
```

1.5.5 Plotting the graph Models V/S Accuracies

[73]:
```
# Convert dictionary to DataFrame
df = pd.DataFrame(list(accuracy_dict.items()), columns=['Classifier',␣
 ↪'Accuracy'])

# Plotting
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='Classifier', y='Accuracy', hue='Classifier',␣
 ↪palette='viridis', legend=False)
plt.title('Classifier Accuracies')
plt.xlabel('Classifier')
plt.ylabel('Accuracy')
plt.ylim(0, 1)  # Set y-axis limit to [0, 1] for better visualization
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



1.6 Conclusion

Based on the accuracies obtained from the classifiers, we observe the following: Logistic Regression achieved the highest accuracy among the classifiers, with an accuracy of approximately 58.70%. Logistic Regression is a powerful linear model that is well-suited for binary classification tasks, and in this case, it has demonstrated its effectiveness in predicting ratings based on the provided dataset.

RandomForestClassifier and KNeighborsClassifier performed relatively well but achieved lower accuracies compared to Logistic Regression, with accuracies of approximately 51.65% and 46.99%, respectively. RandomForestClassifier is an ensemble learning method that builds multiple decision trees and combines their predictions, while KNeighborsClassifier is a non-parametric method based on the similarity of data points. Although these classifiers performed reasonably well, they were outperformed by Logistic Regression in this tas.3.

DecisionTreeClassifier obtained the lowest accuracy among the classifiers, with an accuracy of approximately 46.28%. Decision trees are prone to overfitting and may not generalize well to unseen data, which could explain the lower accuracy observed in this c

data.

*** In conclusion, Logistic Regression emerges as the preferred choice for this classification task, as it achieved the highest accuracy among the classifiers. Its linear nature and ability to model relationships between features make it suitable for predicting ratings based on the provided dataset. Additionally, RandomForestClassifier and KNeighborsClassifier performed reasonably well but were outperformed by Logistic Regression. DecisionTreeClassifier, while intuitive, obtained the lowest accuracy, suggesting that its simplistic nature may not be ideal for this particular task. ***