HOTEL RATING CLASSIFICATION

- **Mentor**: Mr. Karthik
-     MS. Dhanyapriya
- **Date: 16.07.2022**

- **Team Members(Group 4):**
- Aishwarya Upasani
- Mr. Chandramohan Pattanaik
- Mrs. Manisha Thete
- Onkar Gaikwad
- Samiksha Deshmukh
- Miss Shraddha Mali
- Ms. Vaibhavi Taide

BUSINESS OBJECTIVE

This is a sample dataset which consists of 20,000 reviews and ratings for different hotels and our goal is **to examine how travelers are communicating their positive and negative experiences** in online platforms for staying in a specific hotel and **major objective is what are the attributes that travelers are considering while selecting a hotel**. With this manager can understand which elements of their hotel influence more in forming a positive review or improves hotel brand image.

PROJECT ARCHITECTURE

STEP-1 :
DATA SET COLLECTION

STEP-2 :
DATA CLEANING

STEP-3 :
EDA PHASE

STEP-4 :
MODEL BUILDING

STEP-5 :
FINALIZE BEST MODEL

STEP-6 :
COMPARE ACTUAL AND
PREDICTED VALUES

STEP_7:
MODEL DEPLOYMENT

EXCELR
Raising Excellence

DATA SET DETAILS

- 20491 Observations &

  2 Features
- ‘No Null’ values.
- ‘No Duplicates’.
- Review column of OBJECT datatype.
- Rating column of FLOAT datatype.

```
data.info()
```

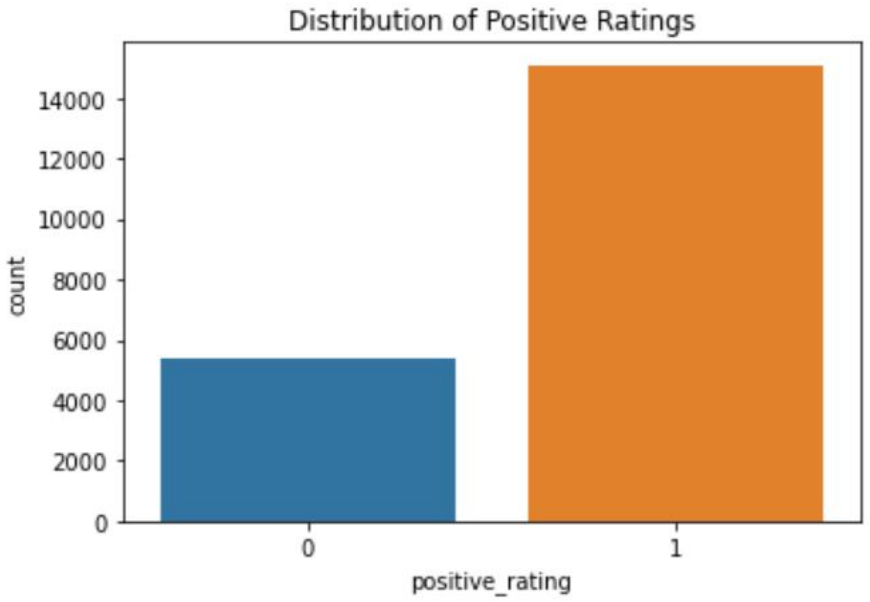
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20491 entries, 0 to 20490
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Review   20491 non-null  object 
 1   Rating   20491 non-null  float64
dtypes: float64(1), object(1)
memory usage: 320.3+ KB
```

```
In [14]: # any duplicate data
         hotel.duplicated().sum()

Out[14]: 0
```

No any duplicate data

VALUE COUNT

Split the positive ratings as rating above 3 and negative ratings as 3 and below



PUNCTUATION PERCENTAGE

This function returns the ratio of punctuation in the text compared to other characters.

```
# Apply the punct_perc function on all reviews in the data set
data['punct%'] = data['Review'].apply(lambda x: punct_perc(x))
data.head()
```

	Review	Rating	positive_rating	punct%
0	nice hotel expensive parking got good deal sta...	4.0	1	0.023762
1	ok nothing special charge diamond member hilto...	2.0	0	0.018081
2	nice rooms not 4* experience hotel monaco seat...	3.0	0	0.026468
3	unique, great stay, wonderful time hotel monac...	5.0	1	0.031373
4	great stay great stay, went seahawk game aweso...	5.0	1	0.036731

NUMBER PERCENTAGE

This function returns the ratio of digits compared to other characters.

```
# Apply the number_perc function on all reviews in the data set
data['number%'] = data['Review'].apply(lambda x: number_perc(x))
data.head()
```

	Review	Rating	positive_rating	punct%	number%
0	nice hotel expensive parking got good deal sta...	4.0	1	0.023762	0.003960
1	ok nothing special charge diamond member hilto...	2.0	0	0.018081	0.009040
2	nice rooms not 4* experience hotel monaco seat...	3.0	0	0.026468	0.019851
3	unique, great stay, wonderful time hotel monac...	5.0	1	0.031373	0.001961
4	great stay great stay, went seahawk game aweso...	5.0	1	0.036731	0.001837

TEXT LENGTH

```python
# Apply the len function to the data set to count the number of characters for each review
data['text_length'] = data['Review'].apply(len)
data.head()
```

	Review	Rating	positive_rating	punct%	number%	text_length
0	nice hotel expensive parking got good deal sta...	4.0	1	0.023762	0.003960	593
1	ok nothing special charge diamond member hilto...	2.0	0	0.018081	0.009040	1689
2	nice rooms not 4* experience hotel monaco seat...	3.0	0	0.026468	0.019851	1427
3	unique, great stay, wonderful time hotel monac...	5.0	1	0.031373	0.001961	600
4	great stay great stay, went seahawk game aweso...	5.0	1	0.036731	0.001837	1281

```python
# Find the review with the maximum text length
data[data['text_length'] == data['text_length'].max()]
```

	Review	Rating	positive_rating	punct%	number%	text_length
7072	honest review visit 5/21-5/28 let begin saying...	3.0	0	0.031896	0.006569	13501

MAKING CLEAN TEXT

```
# remove any symbol and cover letter to lowercase

a=re.sub('[^a-zA-Z0-9]',' ',a)
a=a.lower().split()
a
```

```
['nice',
 'hotel',
 'expensive',
 'parking',
```

```
# made loop for clean reviews - based on the stopwords

clean_word=[i for i in a if not i in sw]
clean_word
```

```
['nice',
 'hotel',
 'expensive',
 'parking',
 'got',
 'good',
```

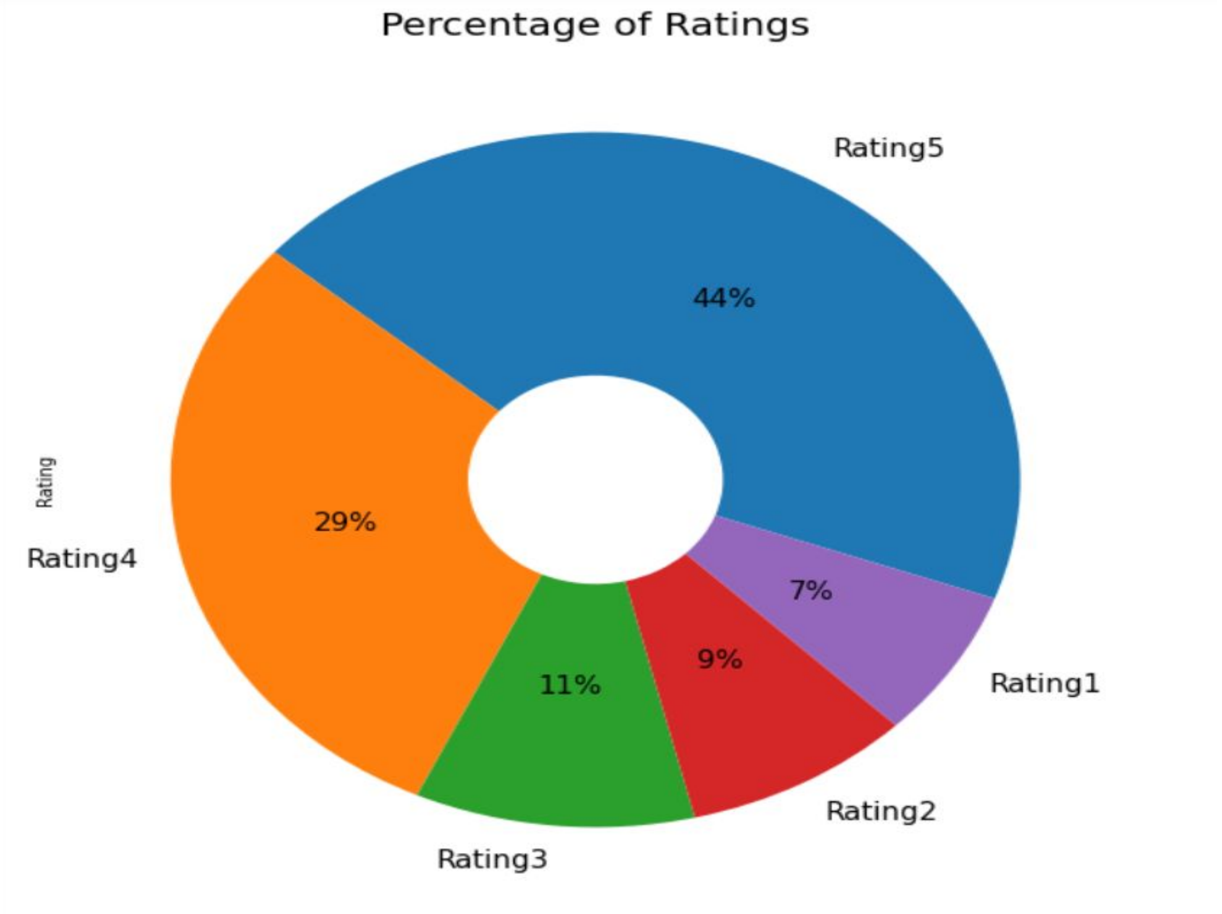
TEXT PREPROCESSING

```
# add new column about the reviews after cleaning

hotel['clean_word']=hotel["Review"].apply(text_preprocessing)
hotel.head()
```

	Review	Rating	Length	clean_word
0	nice hotel expensive parking got good deal sta...	4.0	593	nice hotel expens park got good deal stay hote...
1	ok nothing special charge diamond member hilto...	2.0	1689	ok noth special charg diamond member hilton de...
2	nice rooms not 4* experience hotel monaco seat...	3.0	1427	nice room experi hotel monaco seattl good hote...
3	unique, great stay, wonderful time hotel monac...	5.0	600	uniqu great stay wonder time hotel monaco loca...
4	great stay great stay, went seahawk game aweso...	5.0	1281	great stay great stay went seahawk game awesom...

VISUALIZATION

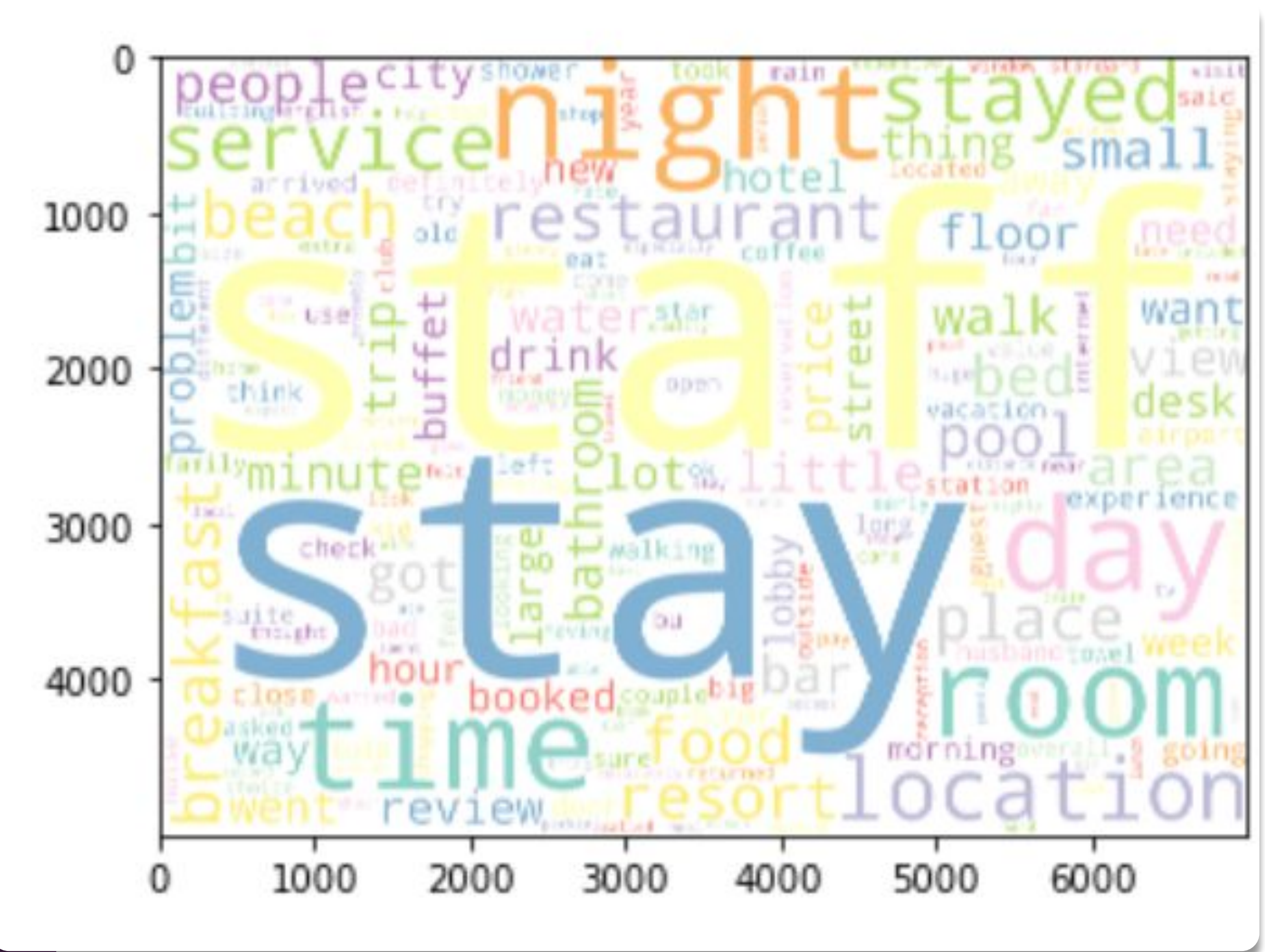
- 44% of total reviews has given 5 rating
- 29% of total reviews has given 4 rating
- 11% of total reviews has given 3 rating
- 9% of total reviews has given 2 rating
- 7% of total reviews has given 1 rating



EXCELR
Raising Excellence

POSITIVE WORLDCLOUD

- STAFF, STAY, LOCATION, SERVICE, TIME are some of the mostly used positive words.



NEGATIVE WORDCLOUD

- GREAT, STAFF, STAY, ROOM, DAY, GOOD are some of the mostly used negative words.

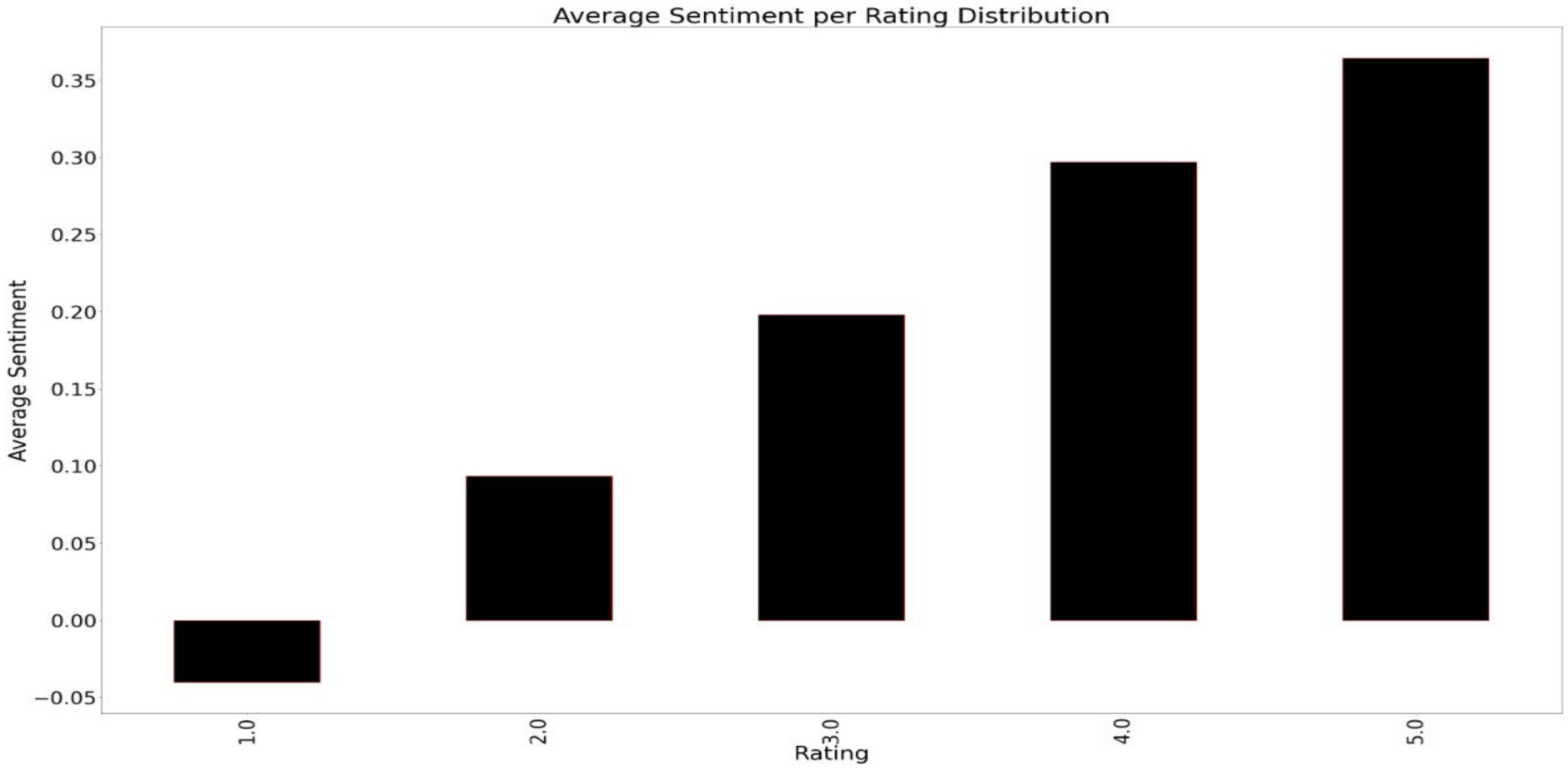
REVIEWS WITH HIGHEST POLARITY

```
5 Random Reviews with Highest Polarity:
Review 1:
 absolutely wonderful wonderful serene oasis city millions steps away times square entering hotel pea
cefulness eveloping lounge wonderful treat morning afternoon wonderful treated meet travellers busine
ss people share experiencesloved
Review 2:
 best went boxing day  weeks best time food pool staff beach room    kids age   didnt want come homei
againloved xx
Review 3:
 excellent excellent excellent stayed nights start november excellent location excellent staff excell
ent price
Review 4:
 number hotel number ranking perfect way best breakfast world
Review 5:
 did not disappoint superb hotel needs lifts elevators prefer smiles staff hong kong inspired pastry
chefs poached hyatt make heavenly
```

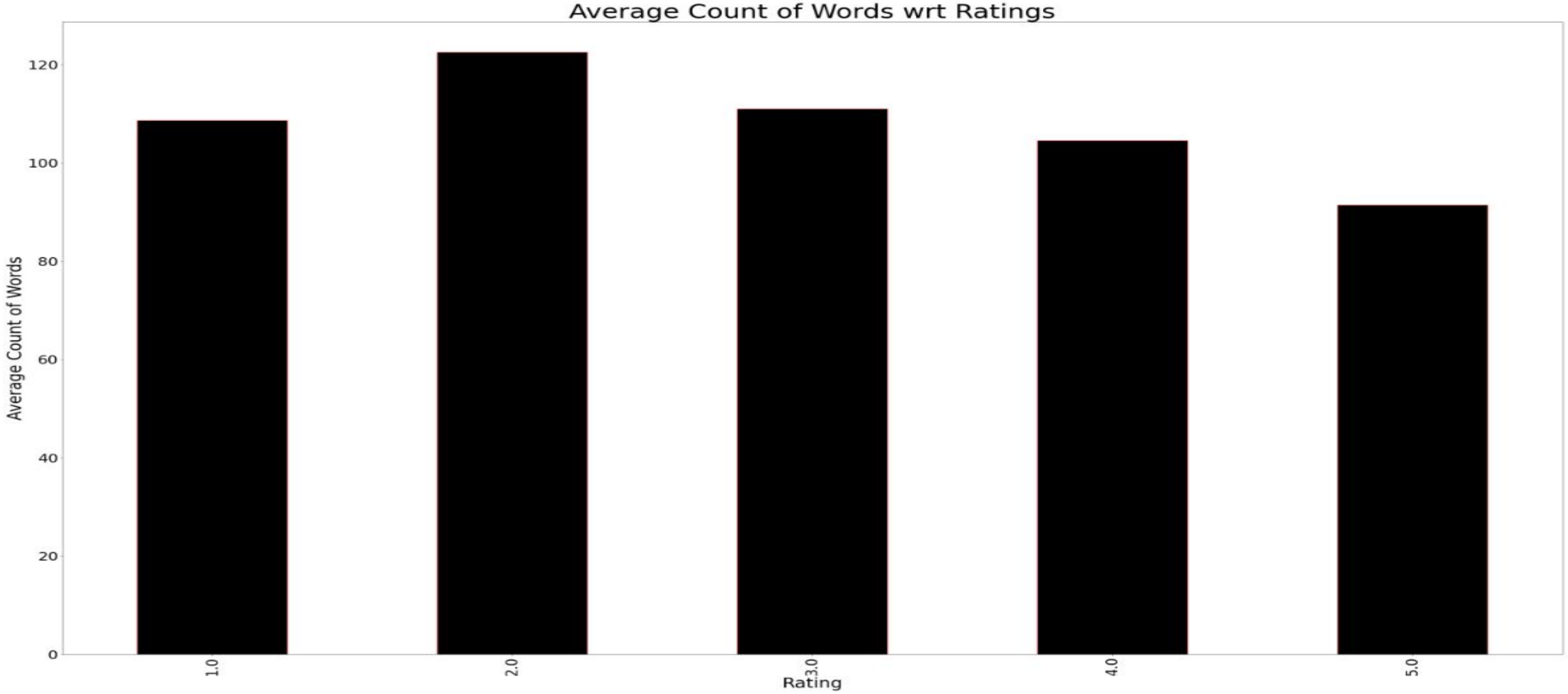
REVIEWS WITH LOWEST POLARITY

```
5 Random Reviews with Lowest Polarity:
Review 1:
 worst location does say place eat sub place make order bullet proof glass
Review 2:
 not stay stayed group people  person ceiling bathroom fell rooms dirty musty overpriced pina colada
no rum  service terrible check charged maid gratuity bell man gratuity absolutely terrible place
Review 3:
 frontdesk extremely bad service checkin onebedroom sept stay  months people desk horrible not answer
calls guest room request items received requested bowl  days reminding morning evening bowl turn upth
e desk said conclusion dont trust
Review 4:
 worst worst hotel experiences life moment wife walked knew trouble carpet filthy odor lobby switched
room odor room got gum stains carpet not mention stains bed spreadsimply horrible
Review 5:
 greta london base stayed london bridge march impresses hotel point view location staff accomodationi
recommend highyl anybody going london weekend
```

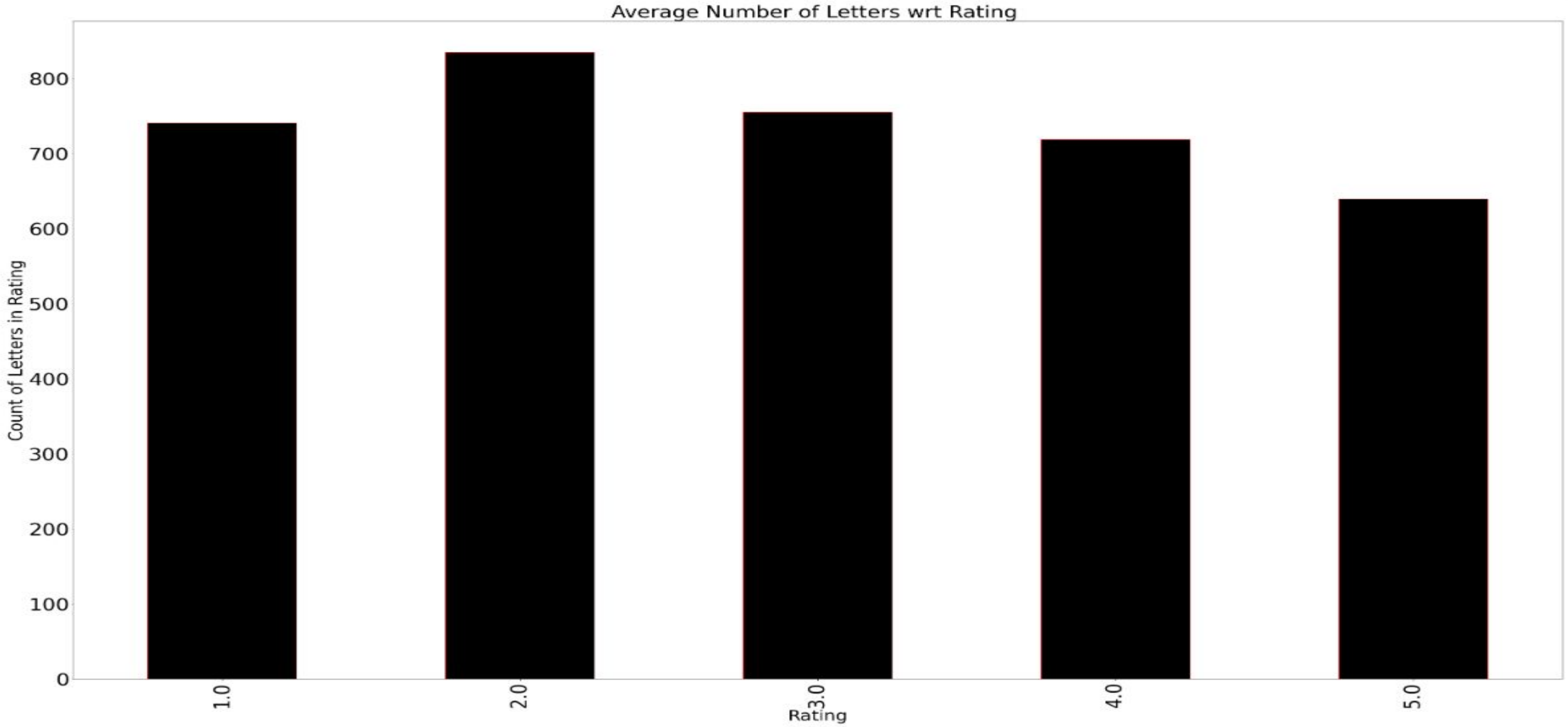
AVERAGE SENTIMENT PER RATING DISTRIBUTION



AVERAGE COUNT OF WORDS WITH RESPECT TO RATINGS



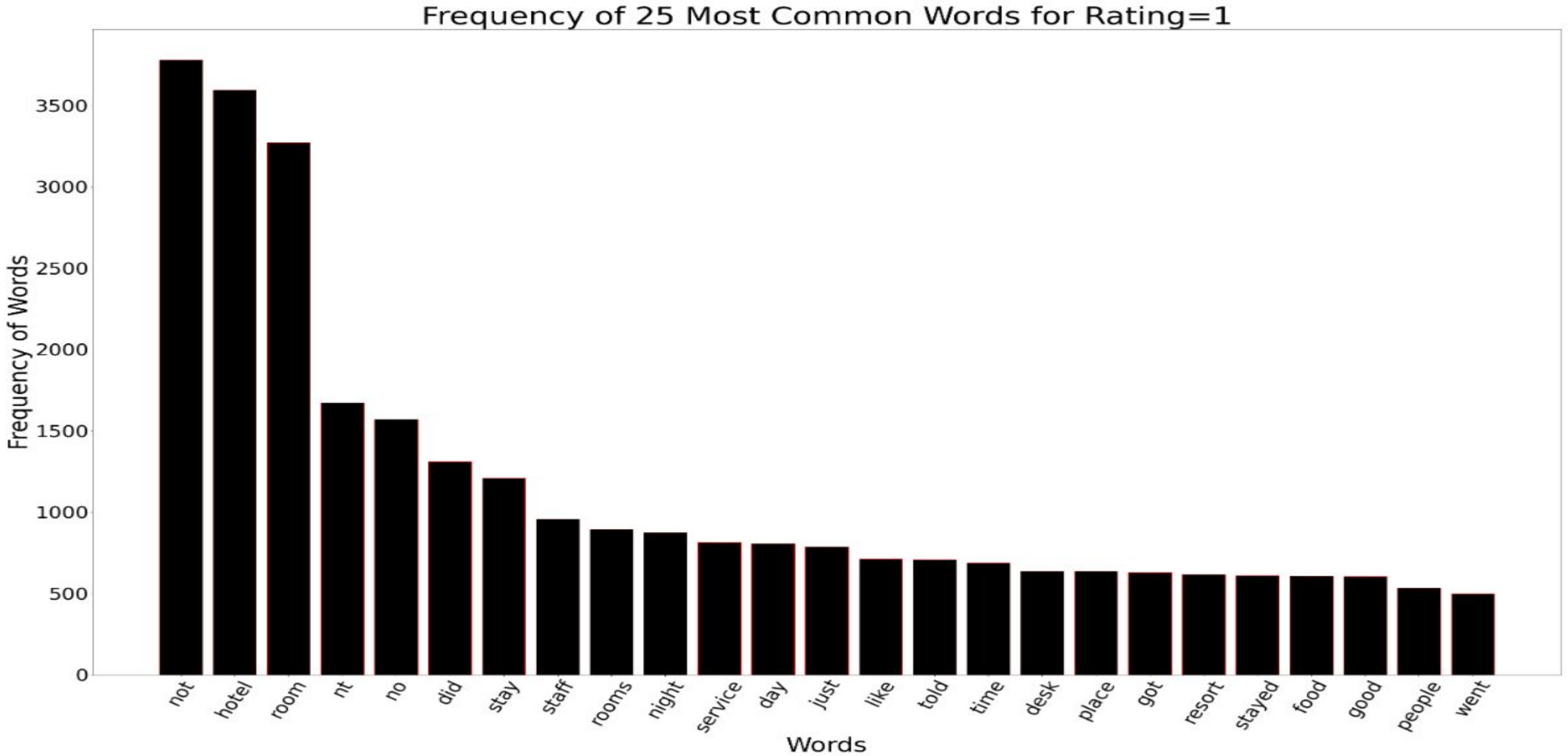
AVERAGE NUMBER OF LETTERS WITH RESPECT TO RATING



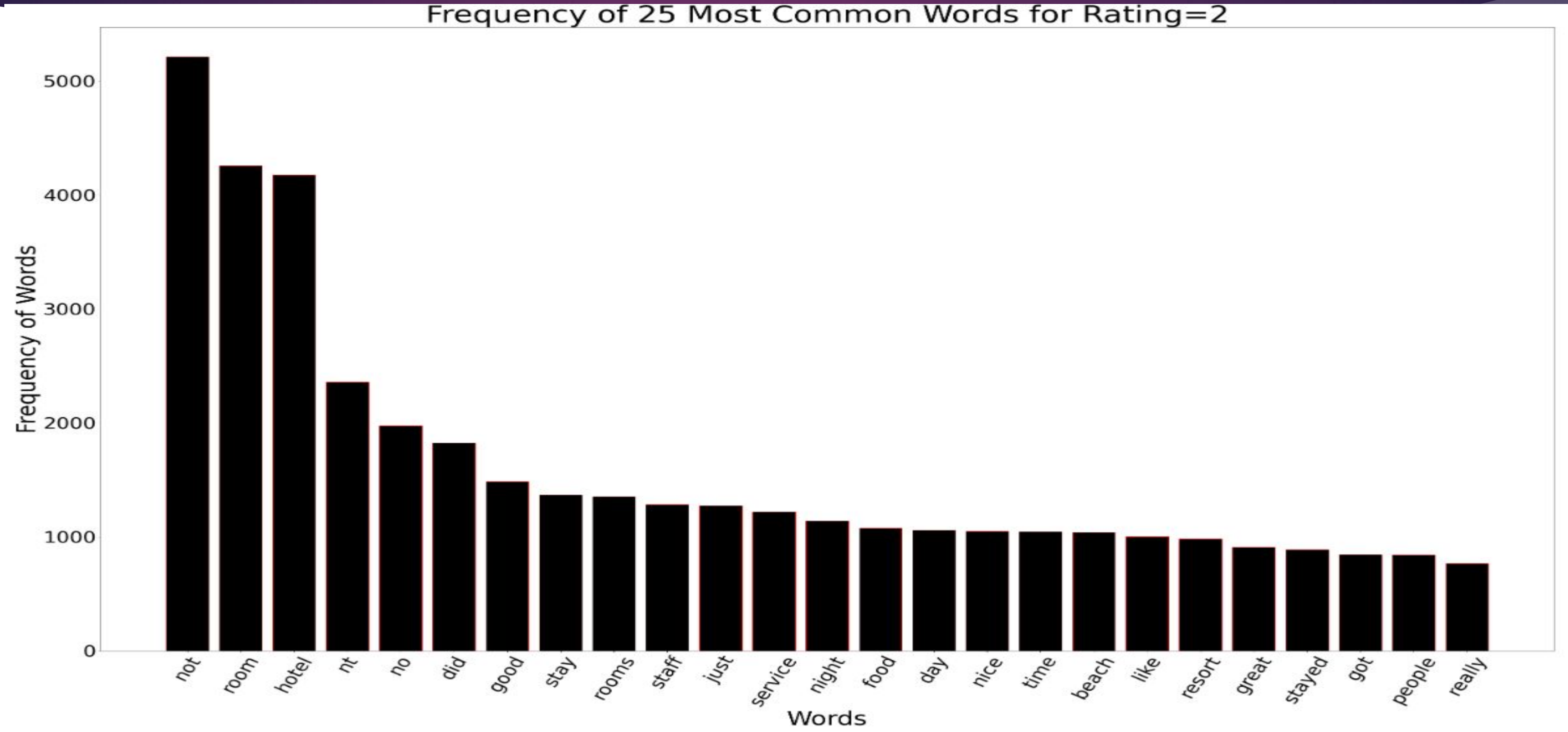
EXCELR
Raising Excellence

TOP 100 COMMONLY USED WORDS

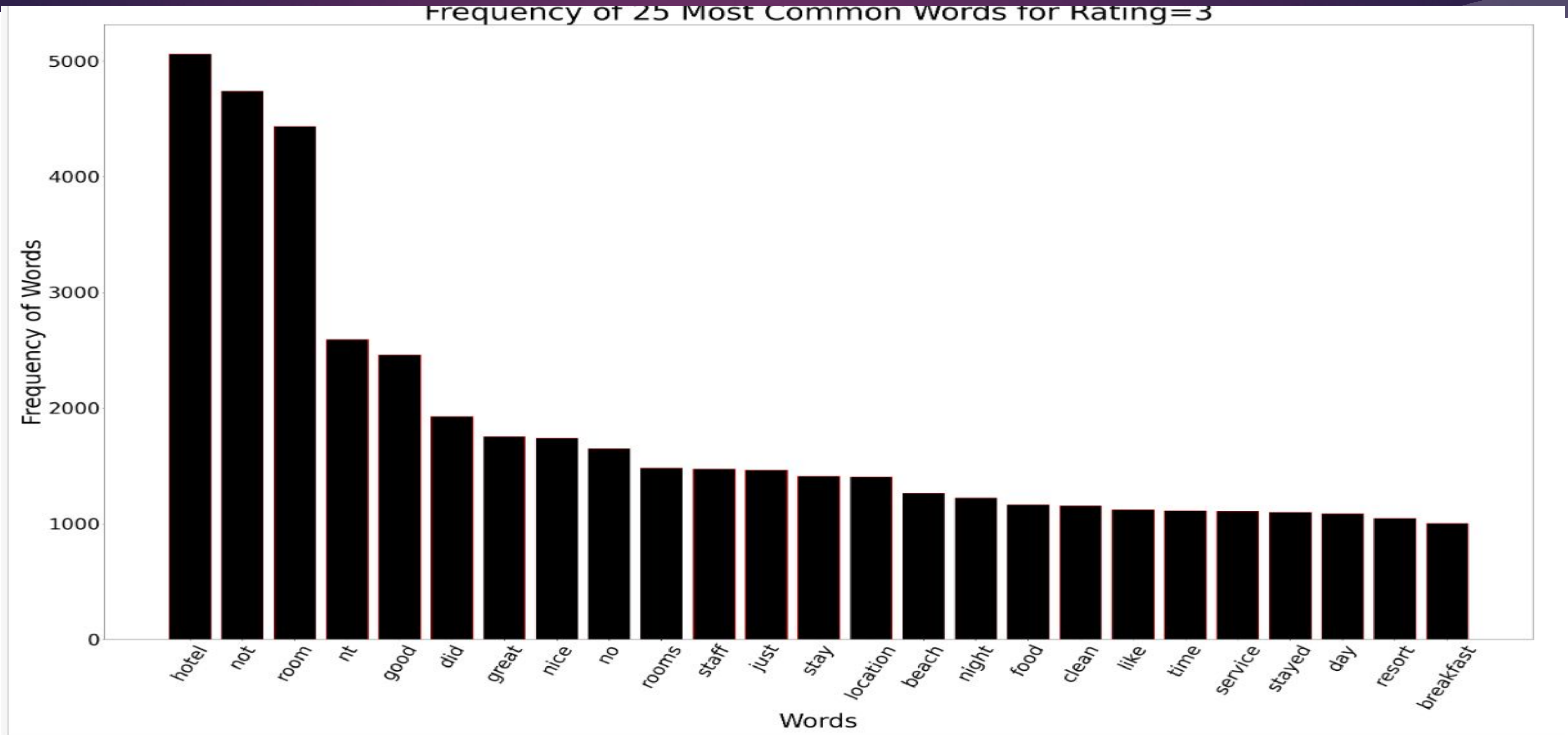
COMMON WORDS FOR RATING 1



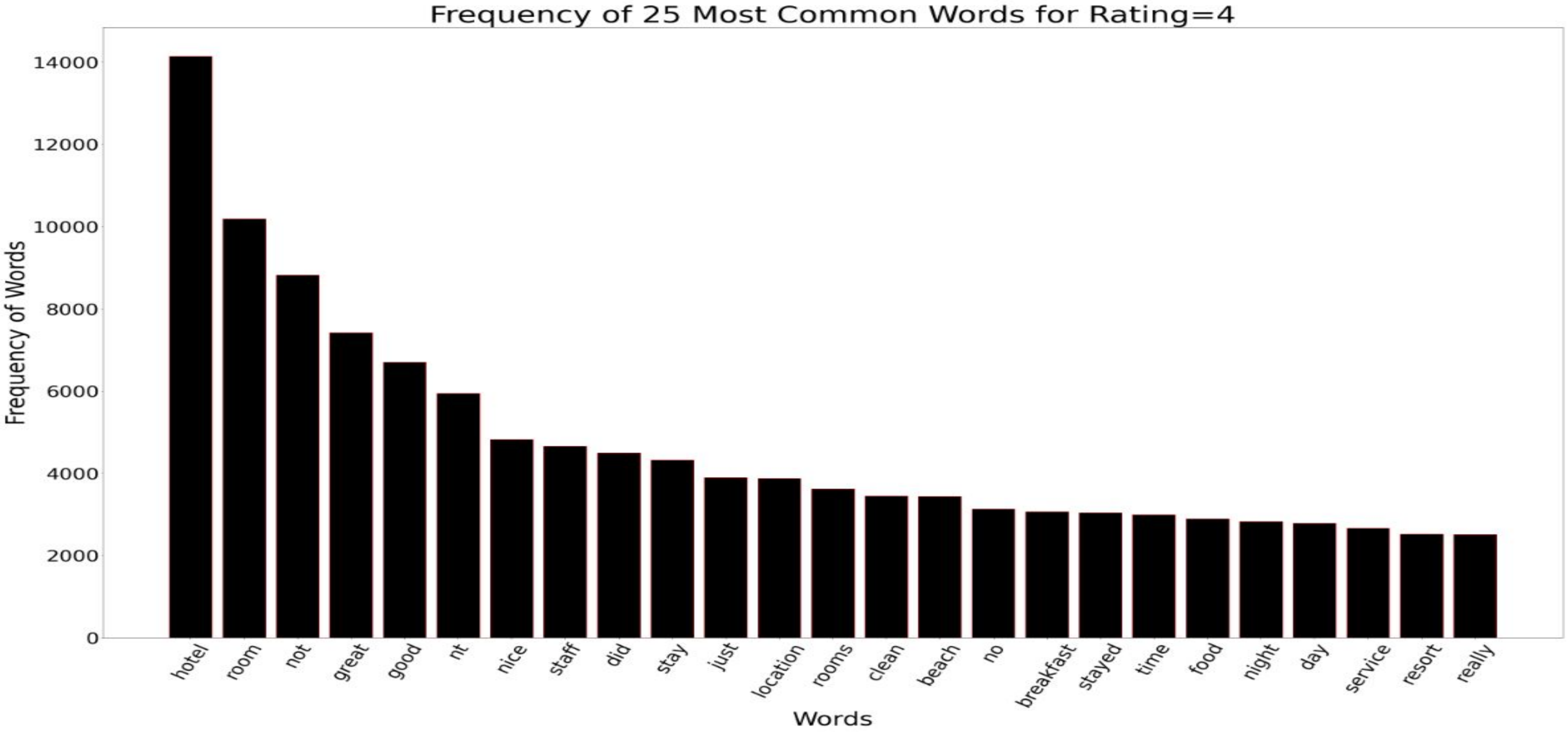
COMMON WORDS FOR RATING 2



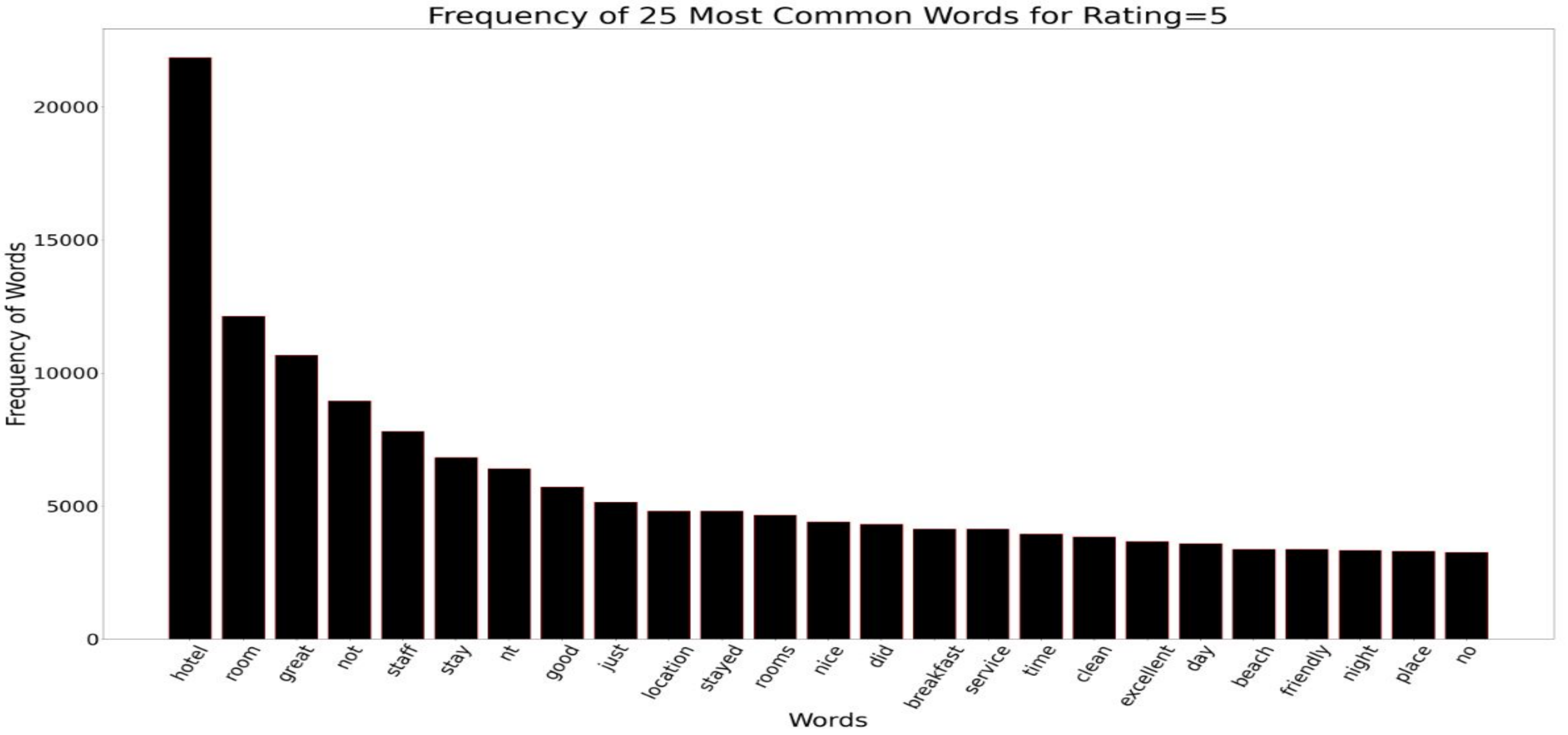
COMMON WORDS FOR RATING 3



COMMON WORDS FOR RATING 4



COMMON WORDS FOR RATING 5



EXCELR
Raising Excellence

Modeling & Evaluation

Modeling and Evaluation

Logistic Regression
Binary outcome

Naïve Bayes
Probability of difft. classes

Support Vector Machine
Linear separation of data in HD feature space

XGBoost Model
Implementation of Gradient Boosted Decision trees

Decision Tree
Split to nodes and sub-nodes to increase homogeneity

Random Forest Classifier
Performing both classification & Regression task

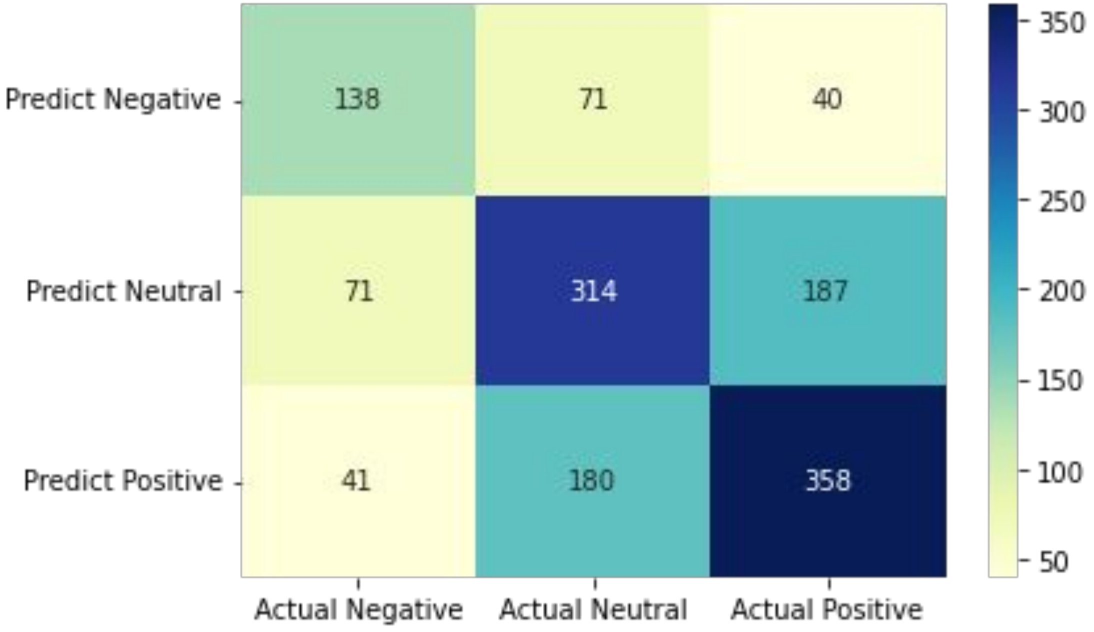
1. Decision Tree Classifier

```
In [20]: dt = DecisionTreeClassifier(random_state=SEED)
         dt.fit(X_train,y_train)
         y_pred_test = dt.predict(X_test)
         print("Training Accuracy score: "+str(round(accuracy_score(y_train,dt.predict(X_train)),4)))
         print("Testing Accuracy score: "+str(round(accuracy_score(y_test,dt.predict(X_test)),4)))

         Training Accuracy score: 1.0
         Testing Accuracy score: 0.5786
```

```
In [21]: print(classification_report(y_test, y_pred_test, target_names=['positive', 'neutral', 'negative']))

                       precision    recall  f1-score   support

             positive       0.55      0.55      0.55       249
              neutral       0.56      0.55      0.55       572
             negative       0.61      0.62      0.62       579

             accuracy                           0.58      1400
            macro avg       0.57      0.57      0.57      1400
         weighted avg       0.58      0.58      0.58      1400
```

Decision Tree Classifier

```
cm = confusion_matrix(y_test, y_pred_test)
#print('Confusion matrix\n', cm)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual Neutral', 'Actual Positive'],
                         index=['Predict Negative', 'Predict Neutral', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()
```



2. Naive Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_train = gnb.predict(X_train)
y_pred_test = gnb.predict(X_test)
print("Training Accuracy score: "+str(round(accuracy_score(y_train,gnb.predict(X_train)),4)))
print("Testing Accuracy score: "+str(round(accuracy_score(y_test,gnb.predict(X_test)),4)))
```

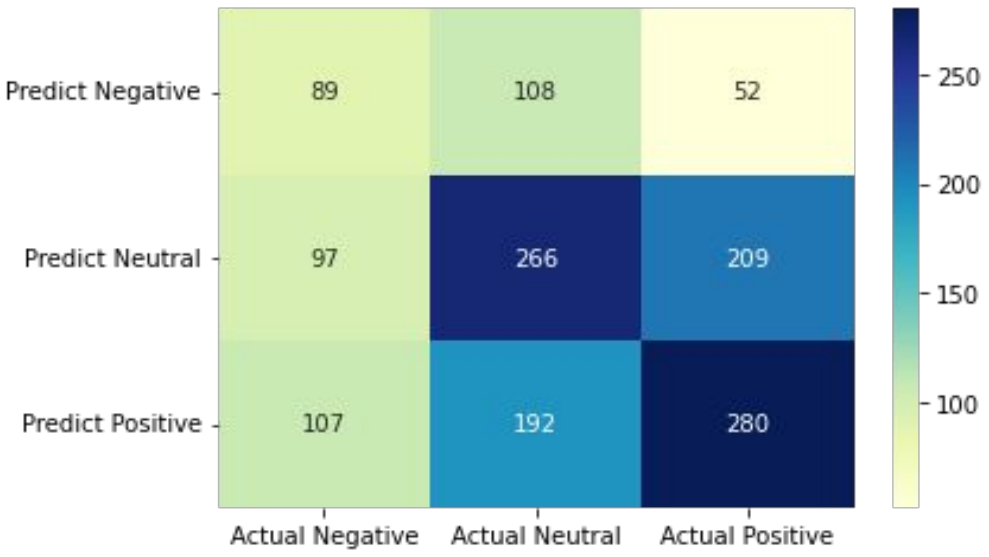
```
Training Accuracy score: 0.8705
Testing Accuracy score: 0.4536
```

```
print(classification_report(y_test, y_pred_test, target_names=['positive', 'neutral', 'negative']))
```

	precision	recall	f1-score	support
positive	0.30	0.36	0.33	249
neutral	0.47	0.47	0.47	572
negative	0.52	0.48	0.50	579
accuracy			0.45	1400
macro avg	0.43	0.44	0.43	1400
weighted avg	0.46	0.45	0.46	1400

Naive Bayes Classifier

```
cm = confusion_matrix(y_test, y_pred_test)
#print('Confusion matrix\n', cm)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual Neutral', 'Actual Positive'],
                         index=['Predict Negative', 'Predict Neutral', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()
```



3. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=SEED).fit(X_train, y_train)
y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
print("Training Accuracy score: "+str(round(accuracy_score(y_train,lr.predict(X_train)),4)))
print("Testing Accuracy score: "+str(round(accuracy_score(y_test,lr.predict(X_test)),4)))
```

```
/Users/vaibhavitaide/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

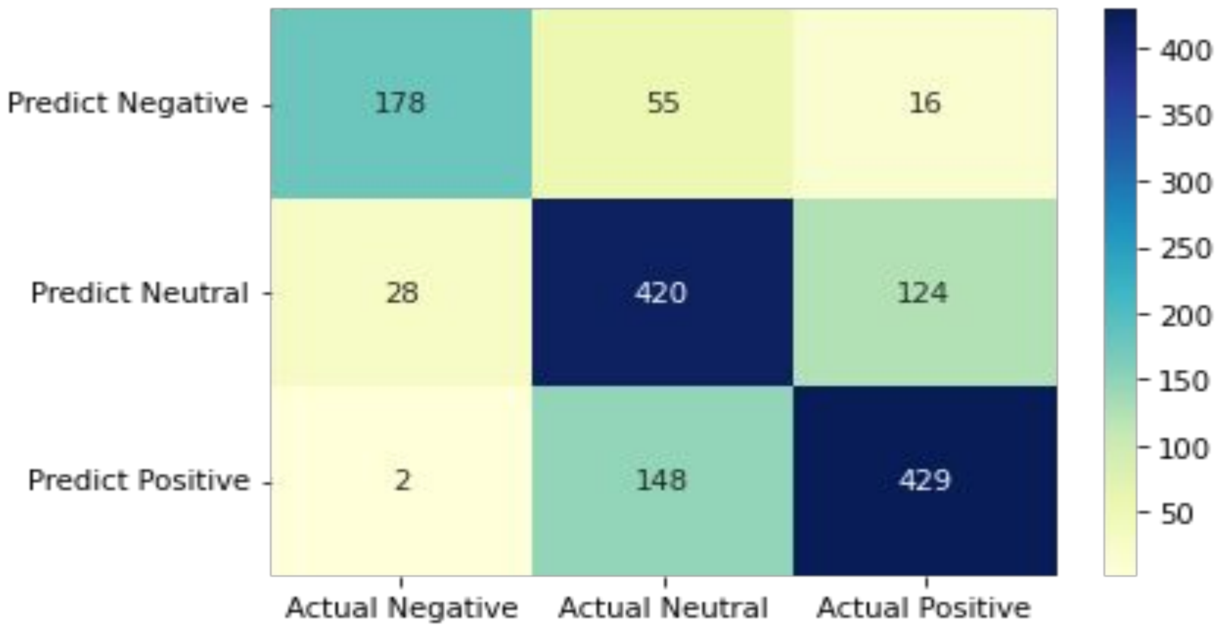
```
Training Accuracy score: 0.8962
Testing Accuracy score: 0.7336
```

```
print(classification_report(y_test, y_pred_test, target_names=['positive', 'neutral', 'negative']))
```

	precision	recall	f1-score	support
positive	0.86	0.71	0.78	249
neutral	0.67	0.73	0.70	572
negative	0.75	0.74	0.75	579
accuracy			0.73	1400
macro avg	0.76	0.73	0.74	1400
weighted avg	0.74	0.73	0.73	1400

Logistic Regression

```
cm = confusion_matrix(y_test, y_pred_test)
#print('Confusion matrix\n', cm)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual Neutral', 'Actual Positive'],
                        index=['Predict Negative', 'Predict Neutral', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()
```



4. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
print("Training Accuracy score: "+str(round(accuracy_score(y_train,clf.predict(X_train)),4)))
print("Testing Accuracy score: "+str(round(accuracy_score(y_test,clf.predict(X_test)),4)))
```

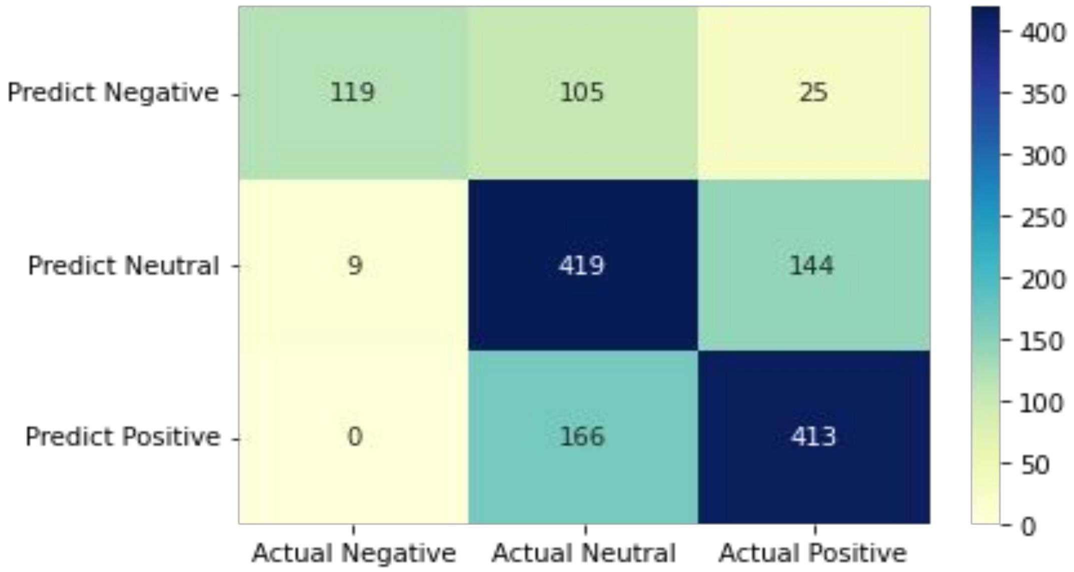
```
Training Accuracy score: 1.0
Testing Accuracy score: 0.6793
```

```
print(classification_report(y_test, y_pred_test, target_names=['positive', 'neutral', 'negative']))
```

```
              precision    recall  f1-score   support

    positive       0.93      0.48      0.63       249
     neutral       0.61      0.73      0.66       572
    negative       0.71      0.71      0.71       579

    accuracy                           0.68      1400
   macro avg       0.75      0.64      0.67      1400
weighted avg       0.71      0.68      0.68      1400
```

Random Forest Classifier

```
cm = confusion_matrix(y_test, y_pred_test)
#print('Confusion matrix\n', cm)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual Neutral', 'Actual Positive'],
                         index=['Predict Negative', 'Predict Neutral', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()
```



5. XGBoost Model

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /Users/vaibhavitaide/opt/anaconda3/lib/python3.9/site-packages (1.6.1)
Requirement already satisfied: scipy in /Users/vaibhavitaide/opt/anaconda3/lib/python3.9/site-packages (from xgboost) (1.7.1)
Requirement already satisfied: numpy in /Users/vaibhavitaide/opt/anaconda3/lib/python3.9/site-packages (from xgboost) (1.20.3)
WARNING: You are using pip version 22.0.4; however, version 22.2.1 is available.
You should consider upgrading via the '/Users/vaibhavitaide/opt/anaconda3/bin/python -m pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

```
from xgboost import XGBClassifier
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=25)
model_xgb = XGBClassifier()
model_xgb.fit(X_train,Y_train)
# make predictions for test data
y_preds = model_xgb.predict(X_test)
predictions = [round(value) for value in y_preds]
```

```
# evaluate predictions
accuracy = accuracy_score(Y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 68.63%
```

XGBoost Model

```
# Confusion Matrix for the model accuracy
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y_test,y_preds)
print (confusion_matrix)
```

```
[[219 104  16]
 [ 37 475 179]
 [ 10 203 507]]
```

```
print(classification_report(Y_test,y_preds, target_names=['positive', 'neutral', 'negative']))
```

```
              precision    recall  f1-score   support

    positive       0.82      0.65      0.72       339
     neutral       0.61      0.69      0.64       691
    negative       0.72      0.70      0.71       720

    accuracy                           0.69      1750
   macro avg       0.72      0.68      0.69      1750
weighted avg       0.70      0.69      0.69      1750
```

6. Support Vector Machine Model

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

```
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size = 0.3)
```

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((4900, 20000), (4900,), (2100, 20000), (2100,))
```

```
model_svm = SVC()
model_svm.fit(x_train , y_train)
```

```
SVC()
```

```
y_pred = model_svm.predict(x_test)# predicting on test data set
pd.Series(y_pred).value_counts() # getting the count of each category
```

```
1    929
2    864
0    307
dtype: int64
```

```
# Confusion Matrix for the model accuracy
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,y_pred)
print (confusion_matrix)
```

```
[[246  97  18]
 [ 54 609 210]
 [  7 223 636]]
```

```
acc = accuracy_score(y_test, y_pred) * 100
print("Accuracy =", acc)
```

```
Accuracy = 71.0
```

```
print("The accuracy of train data is {:.2f} out of 1".format(model_svm.score(x_test,y_test)))
```

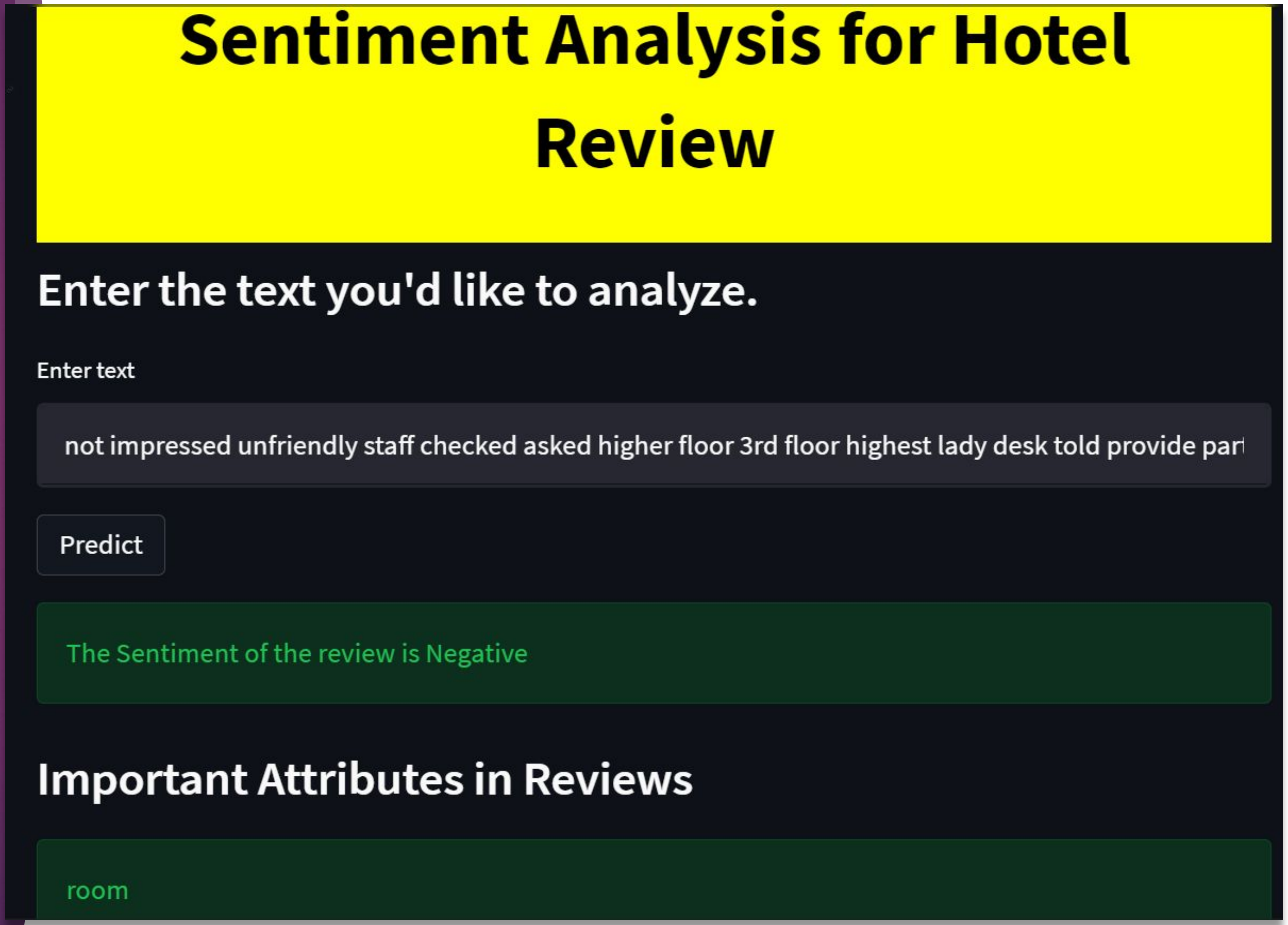
```
The accuracy of train data is 0.71 out of 1
```

```
print(classification_report(y_test,y_pred, target_names=['positive', 'neutral', 'negative']))
```

```
              precision    recall  f1-score   support

    positive       0.80      0.68      0.74       361
     neutral       0.66      0.70      0.68       873
    negative       0.74      0.73      0.74       866

    accuracy                           0.71      2100
   macro avg       0.73      0.70      0.72      2100
weighted avg       0.71      0.71      0.71      2100
```

DEPLOYMENT WITH STREAMLIT

- The deployment shows whether the sentiment is Positive or Negative.
- It also shows the important attributes in the review which can help the management to focus on the part which they have to improve and eventually improve their brand image.



CHALLENGES faced

- The main challenge faced was that the dataset given was highly IMBALANCED.
- A problem with imbalanced classification is that there are too few examples of the minority class for a model to effectively learn the decision boundary. Out of around 20000 reviews, nearly 90% of the reviews were positive.
- One way to solve this problem is to oversample the examples in the minority class. This can be achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. This can balance the class distribution but does not provide any additional information to the model.
- Hence, we used SMOTE technique, balanced the dataset and then used it for deployment.

THANK . YOU