Import Dataset

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

!pip install opendatasets
!pip install datasets
```

```
Requirement already satisfied: opendatasets in
/usr/local/lib/python3.10/dist-packages (0.1.22)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from opendatasets) (4.66.5)
Requirement already satisfied: kaggle in
/usr/local/lib/python3.10/dist-packages (from opendatasets) (1.6.17)
Requirement already satisfied: click in
/usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7)
Requirement already satisfied: six>=1.10 in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(2024.7.4)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(2.8.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(2.32.3)
Requirement already satisfied: python-slugify in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(8.0.4)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(2.0.7)
Requirement already satisfied: bleach in
/usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets)
(6.1.0)
Requirement already satisfied: webencodings in
/usr/local/lib/python3.10/dist-packages (from bleach->kaggle-
>opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle-
>opendatasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle-
>opendatasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle-
>opendatasets) (3.7)
```

```
Requirement already satisfied: datasets in
/usr/local/lib/python3.10/dist-packages (2.20.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.15.4)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Collecting pyarrow>=15.0.0 (from datasets)
  Using cached pyarrow-17.0.0-cp310-cp310-
manylinux_2_28_x86_64.whl.metadata (3.3 kB)
Requirement already satisfied: pyarrow-hotfix in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.6)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.1.4)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in
/usr/local/lib/python3.10/dist-packages (from datasets) (4.66.5)
Requirement already satisfied: xxhash in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.4.1)
Requirement already satisfied: multiprocess in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.5.0,>=2023.1.0 in
/usr/local/lib/python3.10/dist-packages (from
fsspec[http]<=2024.5.0,>=2023.1.0->datasets) (2024.5.0)
Requirement already satisfied: aiohttp in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.10.1)
Requirement already satisfied: huggingface-hub>=0.21.2 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.23.5)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from datasets) (24.1)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.10/dist-packages (from datasets) (6.0.1)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(2.3.4)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.3.1)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(24.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(6.0.5)
```

```
Requirement already satisfied: yarl<2.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.21.2-
>datasets) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2024.7.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.1)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.16.0)
Using cached pyarrow-17.0.0-cp310-cp310-manylinux_2_28_x86_64.whl
(39.9 MB)
Installing collected packages: pyarrow
  Attempting uninstall: pyarrow
    Found existing installation: pyarrow 14.0.1
    Uninstalling pyarrow-14.0.1:
      Successfully uninstalled pyarrow-14.0.1
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
cudf-cu12 24.4.1 requires pyarrow<15.0.0a0,>=14.0.1, but you have
pyarrow 17.0.0 which is incompatible.
ibis-framework 8.0.0 requires pyarrow<16,>=2, but you have pyarrow
17.0.0 which is incompatible.
Successfully installed pyarrow-17.0.0
```

```python
import opendatasets as od
import pandas as pd

# Download Hotel Review dataset from Kaggle
od.download(
    "https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-
reviews")
```

```
Please provide your Kaggle credentials to download this dataset. Learn
more: http://bit.ly/kaggle-creds
Your Kaggle username: hnewbold
Your Kaggle Key: ··········
Dataset URL: https://www.kaggle.com/datasets/andrewmvd/trip-advisor-
hotel-reviews
Downloading trip-advisor-hotel-reviews.zip to ./trip-advisor-hotel-
reviews

100%|██████████| 5.14M/5.14M [00:01<00:00, 3.50MB/s]
```

```python
import pandas as pd

# Read in the downloaded dataset
file
=('/content/trip-advisor-hotel-reviews/tripadvisor_hotel_reviews.csv')
raw_dataset = pd.read_csv(file)
raw_dataset.head(10)
```

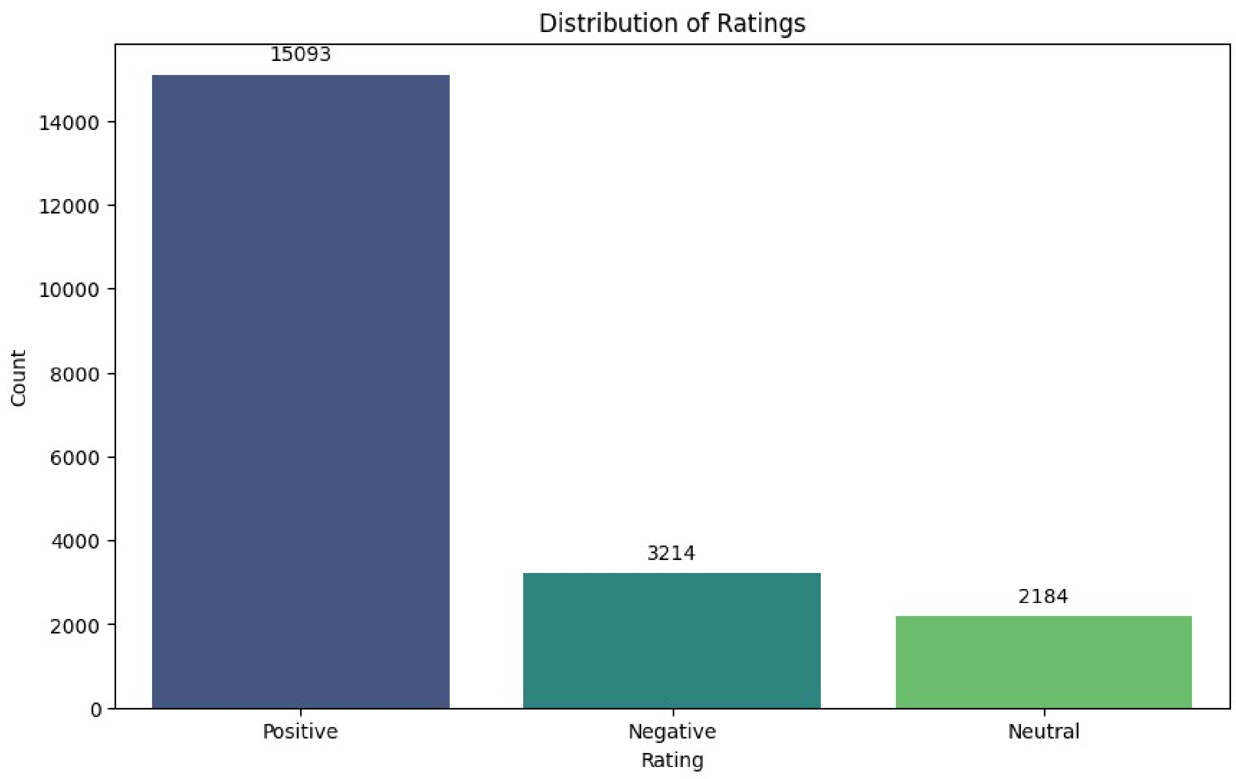
```
{"summary":"{\n  \"name\": \"raw_dataset\",\n  \"rows\": 20491,\n
\"fields\": [\n    {\n      \"column\": \"Review\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 20491,\n        \"samples\": [\n          \"not
recommend hotel did reviewers actually stay hotel did, good thing
hotel location really close leidseplein, shared facilities filthy got,
did not look toilet floor cleaned month, facilities not cleaned 3 days
got, disgusting, staff rude complained left night early refused refund
night, not recommend hotel,  \",\n          \"barcelona rocks, stayed
hotel jazz girlfriend 3 nights end august.the hotel excellent location
carrer pelai, close placa catalunya ramblas appreciate buzz city
removed respite mayhem crowds, caught airport bus barcelona costs 7
euros person return trip, hotel located 2 mins walk maximum placa
universitat stop 2nd route, hotel modern clean, air conditioning room
superb balcony looking street outside, room good size, bathroom fine
scrimp bit toiletries, bring shower gels unless consider washing
unnecessary luxury travels.the downside hotel possibly strengths
depending viewpoint, hotel superbly situated 2 different metro
stations extremely handy need city sights, downside feel rumbling
trains hotel 3rd floor located, worse incessant sound taxi horns hotel
mela times square nyc ca n't say disturbed consideration light
sleepers irritation, couple shops hotel useful snacks drinks want
avoid minibar prices, pool roof tiny n't recommend booking pretty nice
place kill time particularly check day just wanted sit chill, n't
expect able sit night, shuts fairly early.the hotel excellent 3 star
property suggest prices pretty 4 star level, aside hotel tourist bus
stopped directly opposite room recommend good way getting overview
timer city want possible different areas, think used barcelona tours
continuous loop alternative bus turistic 3 different loops involves
changing bus round different areas.see gaudi, no fan architecture
means, parc guell worth visit nice day n't mind walking uphill,
sagrada familia outstanding, couple boat tours port harbour,
personally bit underwhelming, save time money things.the nou camp tour
worth visit no fan football, stadium amazing history club fascinating,
plans redevelopment spectacle completed.ignore negative websites
sprung make barcelona akin early days wild west, n't, just sensible
major city, read ludicrous claims mark target muggers/pickpockets wore
shorts/looked map/did n't speak spanish, rubbish, area ramblas felt
safe, couple areas lower end raval little seedy late night just n't
stupid fine, worst crime encountered charged 10 euros 2 heinekens
irish bar just ramblas heard no-one eats restaurants 9 suffer lack
atmosphere did, did n't appear strictly true concede 3 nights, tell
restaurants open 8-8.30pm worth getting avoid queues, saturday night
arrived restaurant 8.30 offered seat terrace opting window seat
overlooking ramblas, 9 turning people away, 9.45 taken pick tables,
burst activity 9 n't bad.enjoy,  \",\n          \"ok hotel good
location stayed night way beijing rawa island, hotel service room ok.
location great shopping restaurants, probably stay, opinion nice 3
star hotel,  \"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Rating\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"raw_dataset"}
```

Clean and Explore the Dataset

```
# Check observation counts and datatypes
raw_dataset.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
```

```
# Check for duplicates
raw_dataset.duplicated().sum()

0

# Make a copy of the dataset to be cleaned
cleaned_dataset = raw_dataset.copy(deep=True)

# Create function to place ratings into a group
def grouper(score):
    if score in [1, 2]:
        return 'Negative'
    elif score == 3:
        return 'Neutral'
    else:
        return 'Positive'

# Create a grouped rating columm
cleaned_dataset['grouped_rating'] =
cleaned_dataset['Rating'].apply(grouper)

import seaborn as sns
import matplotlib.pyplot as plt

# Show distribution of grouped ratings
plt.figure(figsize=(10, 6))
ax = sns.countplot(data=cleaned_dataset, x='grouped_rating',
palette='viridis')

# Add labels on top of the bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10),
textcoords='offset points')

plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



```
# Take samples of each of the different ratings groups
sampled_negative = cleaned_dataset[cleaned_dataset['grouped_rating']
== 'Positive'].sample(2000, random_state=15)
sampled_neutral = cleaned_dataset[cleaned_dataset['grouped_rating'] ==
'Neutral'].sample(2000, random_state=15)
sampled_positive = cleaned_dataset[cleaned_dataset['grouped_rating']
== 'Negative'].sample(2000, random_state=15)

# Combine the sampled instances into a single balanced dataset
cleaned_dataset_balanced_sample = pd.concat([sampled_negative,
sampled_neutral, sampled_positive]).reset_index(drop=True)

import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

### Create functions to clean and lemmatize reviews

nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)
    # Remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text,
flags=re.MULTILINE)
    # Remove special characters and numbers
    text = re.sub(r'\W', ' ', text)
    # Remove single characters
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)
    # Remove multiple spaces
    text = re.sub(r'\s+', ' ', text, flags=re.I)
    # Convert to lowercase
    text = text.lower()
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in
stop_words])
    return text

def lemmatize_text(text):
    return ' '.join([lemmatizer.lemmatize(word) for word in
text.split()])

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...

# Apply cleaning function to balanced sample
cleaned_dataset_balanced_sample['cleaned_review'] =
cleaned_dataset_balanced_sample['Review'].apply(clean_text)

## Check that data was adequately cleaned

pd.set_option('display.max_colwidth', None)

# Filter DataFrame to rows with less than 100 characters where stop
words were removed
mask = cleaned_dataset_balanced_sample['Review'].str.contains(r'\b(?:
{})\b'.format('|'.join(stop_words)))
length_mask = cleaned_dataset_balanced_sample['Review'].str.len() <=
100
combined_mask = mask & length_mask
filtered_df = cleaned_dataset_balanced_sample[combined_mask]
[['Review', 'cleaned_review']]

# Display the filtered DataFrame
print(filtered_df)


Review  \
1279  loved art, comfortable chic hotel huge bed, late checking-in
time hinderance excellent experience,
2010                                           nice odd nice place
```

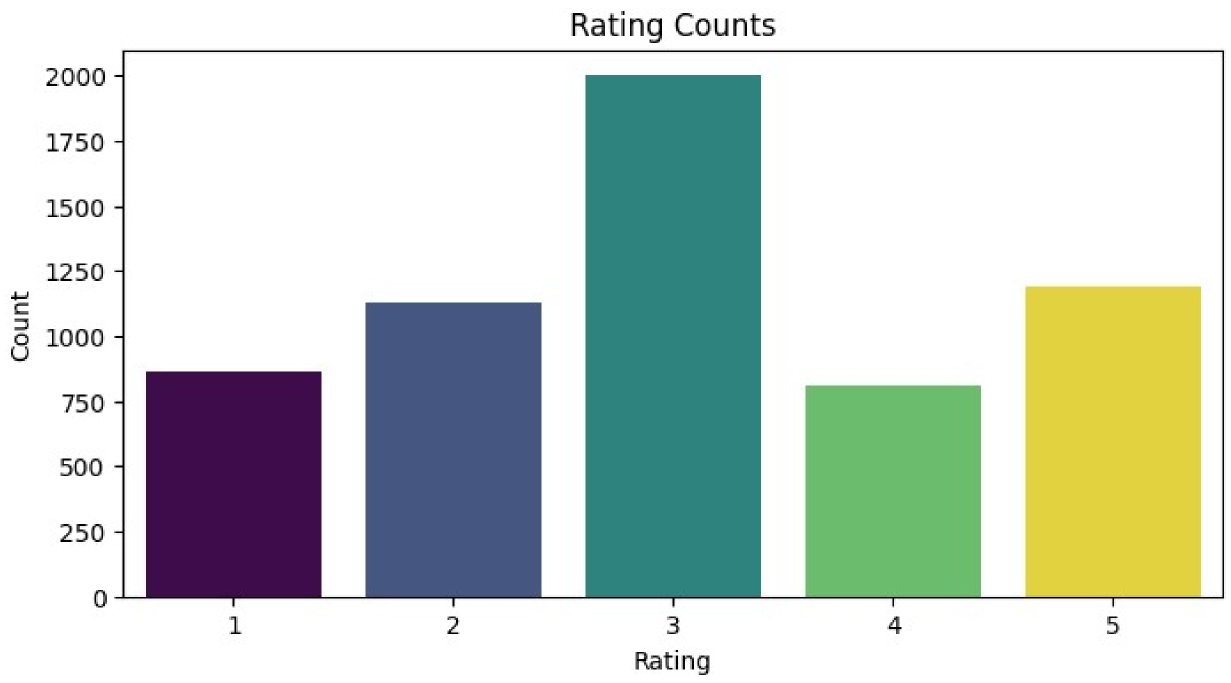
```
stay, just buddies seeing paris, good,
2768       bathroom shower hotel staff good.you sit bathtub order use
hand shower, no refrigerator room,
2793                not bad, not bad hotel needs renovation beach pool
good restaurant rooms need touch,
3224                basic nice, basic hotel clean tidy, did n't think
staff friendly honest n't helpful,
3578                                                      issues n't
say 4 star service great pool bar,
3615                                        big price small room price
hype expect sooooo just adequate,
4201             warning, careful staying place, hotel residence,
people place charge day not residence,
4403                 pass ach leidse square pass hotel, rooms small
dingy, moved nh just corner happier,
4858             bad hotel really awful place dirty room rude staff
desperatley bad breakfast, not stay,
5143    not good, long stairs impossibly steep, room tiny person stay
bed let room, not recommend hotel,
5660      skip just returned puerto rico inform going skip hotel,
enjoy beautiful island homework hotel,
5721              ugh, nasty filthy smelly jaded faded, great
location, did week better federal prisons,
5786               definite no, stay away, terrible food service, yes
beach beautiful pay 3 star resort,


cleaned_review
1279  loved art comfortable chic hotel huge bed late checking time
hinderance excellent experience
2010                                                nice odd nice place
stay buddies seeing paris good
2768          bathroom shower hotel staff good sit bathtub order use
hand shower refrigerator room
2793                    bad bad hotel needs renovation beach pool good
restaurant rooms need touch
3224                         basic nice basic hotel clean tidy think
staff friendly honest helpful
3578                                                         issues say 4
star service great pool bar
3615                                        big price small room price
hype expect sooooo adequate
4201               warning careful staying place hotel residence
people place charge day residence
4403                   pass ach leidse square pass hotel rooms small
dingy moved nh corner happier
4858             bad hotel really awful place dirty room rude staff
desperatley bad breakfast stay
5143          good long stairs impossibly steep room tiny person stay
```

```
bed let room recommend hotel
5660       skip returned puerto rico inform going skip hotel enjoy
beautiful island homework hotel
5721                ugh nasty filthy smelly jaded faded great location
week better federal prisons
5786                definite stay away terrible food service yes beach
beautiful pay 3 star resort

# Apply lemmatization function to balanced sample
cleaned_dataset_balanced_sample['lemmatized_text'] =
cleaned_dataset_balanced_sample['cleaned_review'].apply(lemmatize_text
)

# Check for sucessful lemmatization
cleaned_dataset_balanced_sample[['cleaned_review','lemmatized_text']].
head(3)
```

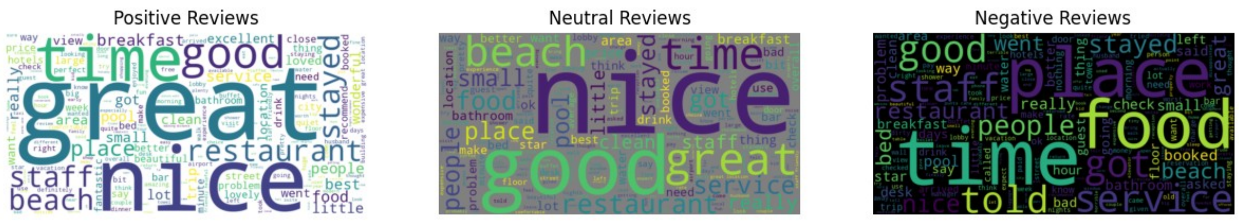
{"summary":"{\n \"name\":
\"cleaned_dataset_balanced_sample[['cleaned_review','lemmatized_text']
]\",\n \"rows\": 3,\n \"fields\": [\n   {\n   \"column\":
\"cleaned_review\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   \"sabrina concierge tremendous spent week le littre
october concierge staff helpful arranging transportation airport
various small things story end left wallet blouse room checked called
sabrina airport located items shipped home new jersey leaving wallet
potentially disaster intact sabrina nothing short wonderful\",\n
\"nice choice enjoyed staying hotel room mate mario room simple nice
clean location wonderful near opera plaza del sol breakfast wonderful
staff kind\",\n   \"great great hotel rooms nice breakfast
great service high standardof course free bus airport long parking
free book arrangement etcto city center 8 km free rental bike close
beautifal forest jogging walking\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"lemmatized_text\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n   \"sabrina
concierge tremendous spent week le littre october concierge staff
helpful arranging transportation airport various small thing story end
left wallet blouse room checked called sabrina airport located item
shipped home new jersey leaving wallet potentially disaster intact
sabrina nothing short wonderful\",\n   \"nice choice enjoyed
staying hotel room mate mario room simple nice clean location
wonderful near opera plaza del sol breakfast wonderful staff kind\",\n
\"great great hotel room nice breakfast great service high standardof
course free bus airport long parking free book arrangement etcto city
center 8 km free rental bike close beautifal forest jogging walking\"\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

Explore Data

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

# Plot the rating counts of the balanced dataset
plt.figure(figsize=(8, 4))
sns.countplot(x='Rating', data=cleaned_dataset_balanced_sample, hue =
'Rating', legend=False, palette='viridis')
plt.title('Rating Counts')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



```
from wordcloud import WordCloud

## Plot word clouds

# Define additional stopwords
additional_stopwords = set(['hotel',
'room','rooms','stay','resort','day','night'])
stopwords = WordCloud().stopwords.union(additional_stopwords)

# Create text for each category using the cleaned_review column
positive_text = '
'.join(cleaned_dataset_balanced_sample[cleaned_dataset_balanced_sample
['Rating'].isin([4, 5])]['cleaned_review'])
neutral_text = '
'.join(cleaned_dataset_balanced_sample[cleaned_dataset_balanced_sample
['Rating'] == 3]['cleaned_review'])
negative_text = '
'.join(cleaned_dataset_balanced_sample[cleaned_dataset_balanced_sample
['Rating'].isin([1, 2])]['cleaned_review'])

# Generate word clouds
wordcloud_positive = WordCloud(width=800, height=400,
background_color='white', stopwords=stopwords).generate(positive_text)
wordcloud_neutral = WordCloud(width=800, height=400,
background_color='grey', stopwords=stopwords).generate(neutral_text)
wordcloud_negative = WordCloud(width=800, height=400,
background_color='black', stopwords=stopwords).generate(negative_text)

# Plot the word clouds
plt.figure(figsize=(15, 10))

plt.subplot(1, 3, 1)
plt.imshow(wordcloud_positive, interpolation='bilinear')
plt.title('Positive Reviews')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(wordcloud_neutral, interpolation='bilinear')
plt.title('Neutral Reviews')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(wordcloud_negative, interpolation='bilinear')
plt.title('Negative Reviews')
plt.axis('off')

plt.show()
```



Build Model

```
from transformers import AutoTokenizer,
AutoModelForSequenceClassification
import torch

# Initialize the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained('nlptown/bert-base-
multilingual-uncased-sentiment')
```

```
model =
AutoModelForSequenceClassification.from_pretrained('nlptown/bert-base-
multilingual-uncased-sentiment')
```

{"model_id":"08e025aca9f7456ebbdfce9cb3a6af64","version_major":2,"vers
ion_minor":0}

{"model_id":"05710ce31c6e4802833e3cd41f4c4c38","version_major":2,"vers
ion_minor":0}

{"model_id":"29dd11e6d68d4f7f952ad3ccb86bc600","version_major":2,"vers
ion_minor":0}

{"model_id":"fa6c33022b174d6a9388c2c33f655db4","version_major":2,"vers
ion_minor":0}

{"model_id":"21724ba851ca422886ac2e0c9784ff3b","version_major":2,"vers
ion_minor":0}

```
# Function to tokenize the dataset
def tokenize_function(examples):
    return tokenizer(examples['lemmatized_text'], truncation=True,
padding='max_length', max_length=512)

# Convert categorical labels to numerical labels
label_map = {'Negative': 0, 'Neutral': 1, 'Positive': 2}
cleaned_dataset_balanced_sample['label'] =
cleaned_dataset_balanced_sample['grouped_rating'].map(label_map)

from transformers import Trainer, TrainingArguments,
EarlyStoppingCallback
from datasets import Dataset

# Convert to Hugging Face Dataset
dataset =
Dataset.from_pandas(cleaned_dataset_balanced_sample[['lemmatized_text'
, 'label']])
tokenized_datasets = dataset.map(tokenize_function, batched=True)

# Split the dataset
train_test_split = tokenized_datasets.train_test_split(test_size=0.2)
train_dataset = train_test_split['train']
val_dataset = train_test_split['test']
```

{"model_id":"3b8d98e180aa455f9be53657411cd25f","version_major":2,"vers
ion_minor":0}

```
# Define training arguments
training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy='epoch',
    save_strategy='epoch',
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=5,
    weight_decay=0.01,
    load_best_model_at_end=True,
)

# Define the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=2)]
)

# Fine-tune the model
trainer.train()
```

```
<IPython.core.display.HTML object>
```

```
TrainOutput(global_step=1200, training_loss=0.4239166291554769,
metrics={'train_runtime': 1896.2322, 'train_samples_per_second':
12.657, 'train_steps_per_second': 0.791, 'total_flos':
5051868335308800.0, 'train_loss': 0.4239166291554769, 'epoch': 4.0})
```

Evaluate Model

```
trainer.evaluate()
```

```
<IPython.core.display.HTML object>
```

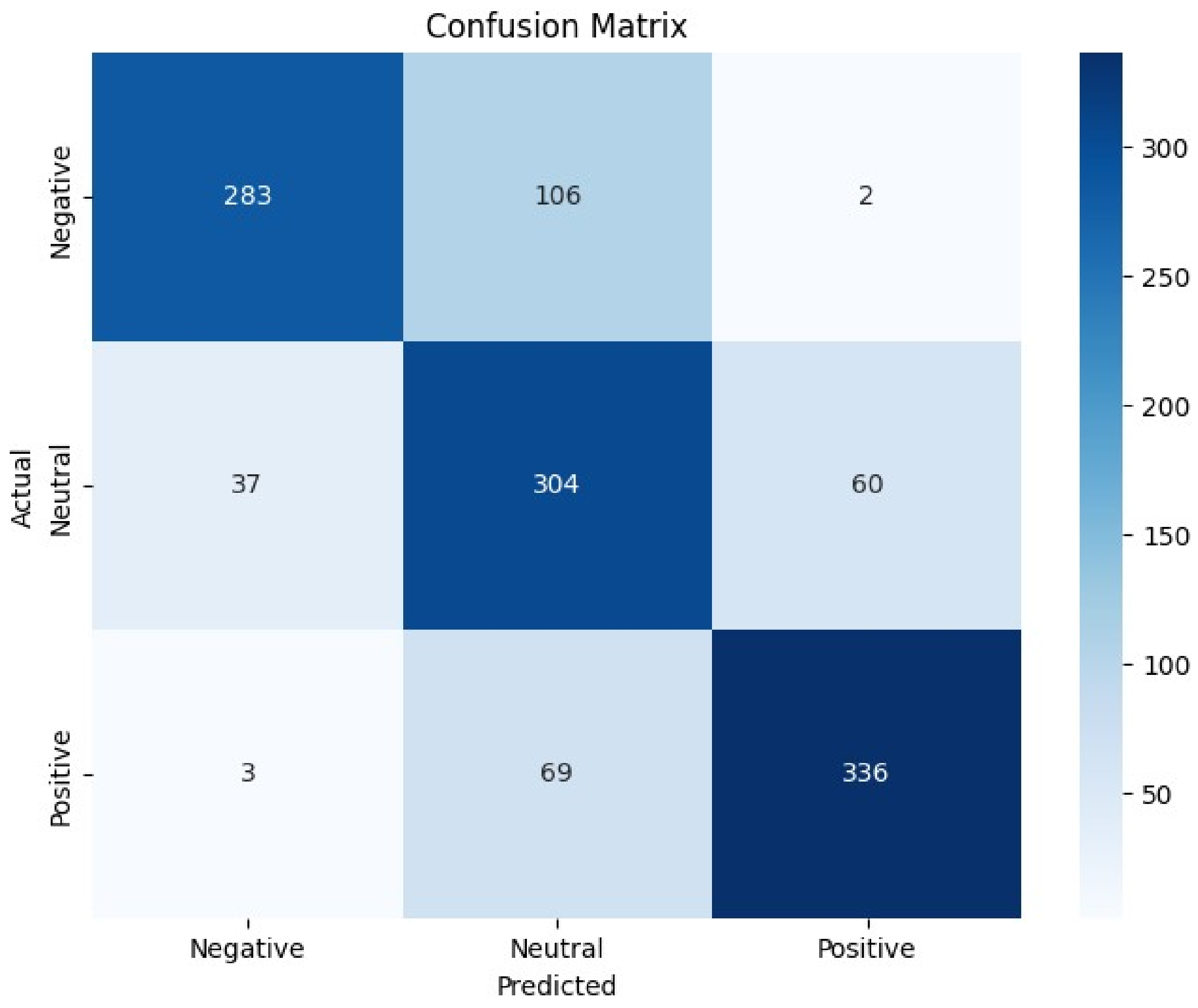
```
{'eval_loss': 0.5410590767860413,
 'eval_runtime': 34.362,
 'eval_samples_per_second': 34.922,
 'eval_steps_per_second': 2.183,
 'epoch': 4.0}
```

```
import numpy as np

# Make predictions on the validation set
predictions = trainer.predict(val_dataset)
preds = np.argmax(predictions.predictions, axis=1)
```

```
<IPython.core.display.HTML object>
```

```
# Print classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(val_dataset['label'], preds,
target_names=['Negative', 'Neutral', 'Positive']))

# Plot confusion matrix
conf_matrix = confusion_matrix(val_dataset['label'], preds)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['Negative', 'Neutral', 'Positive'],
yticklabels=['Negative', 'Neutral', 'Positive'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
              precision    recall  f1-score   support

    Negative       0.88      0.72      0.79       391
     Neutral       0.63      0.76      0.69       401
    Positive       0.84      0.82      0.83       408

    accuracy                           0.77      1200
   macro avg       0.79      0.77      0.77      1200
weighted avg       0.78      0.77      0.77      1200
```


Confusion Matrix
283
106
2
37
304
60
3
69
336
Negative
Neutral
Positive
Actual
Negative
Neutral
Positive
Predicted
300
250
200
150
100
50