Rodriguez_Felipe_DSC680_Final_Code

May 31, 2024

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv('reviews.csv')
```

[5]:
```
description = {
    'index': 'index',
    'listing_id': 'Identifier for listing',
    'id': 'Identifier for review',
    'date': 'Date of review',
    'reviewer_id': 'Identifier for reviewer',
    'reviewer_name': 'Reviewer Name',
    'comments': 'Comments made by the reviewer about the listing'
              }
# Initialize an empty dictionary to store data types
dtype_dict = {}

# Iterate through each column and store its data type in the dictionary
for col in df.columns:
    dtype_dict[col] = str(df[col].dtype)

series1 = pd.Series(description, name='description')
series1 = series1.rename_axis('column')
series2 = pd.Series(dtype_dict, name='data_type')
series2 = series2.rename_axis('column')



# Combining the Series into a DataFrame using pd.merge()
data_dictionary = pd.merge(series1, series2, left_index=True, right_index=True)
print('Data Dictionary\n')
print(data_dictionary.to_markdown())
```

```
Data Dictionary

| column        | description                                     | data_type
|
|:--------------|:------------------------------------------------|:-----------
|
| index         | index                                          | int64
|
| listing_id    | Identifier for listing                         | int64
|
| id            | Identifier for review                          | int64
|
| date          | Date of review                                 | object
|
| reviewer_id   | Identifier for reviewer                        | int64
|
| reviewer_name | Reviewer Name                                  | object
|
| comments      | Comments made by the reviewer about the listing | object
|
```

[3]:
```
df.head()
```

[3]:
```
   index  listing_id        id        date  reviewer_id reviewer_name  \
0      0        3781  37776825  2015-07-10     36059247          Greg
1      1        3781  41842494  2015-08-09     10459388           Tai
2      2        3781  45282151  2015-09-01     12264652        Damien
3      3        3781  49022647  2015-09-30     41426327          Mike
4      4        3781  52503327  2015-10-30     15151513          Ivan

                                            comments
0  The apartment was as advertised and Frank was …
1  It was a pleasure to stay at Frank's place. Th…
2  The apartment description is entirely faithful…
3  Thoroughly enjoyed my time at Frank's home. Ha…
4  Great value for the money! This location has e…
```

[126]:
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Download NLTK stopwords
nltk.download('stopwords')
nltk.download('punkt')

# Get the list of English stopwords
stop_words = set(stopwords.words('english'))

# Function to remove stop words from text
def remove_stopwords(text):
    tokens = word_tokenize(str(text))  # Ensure text is converted to string
    filtered_tokens = [word for word in tokens if word.lower() not in␣
 ↪stop_words]
```

```
    return ' '.join(filtered_tokens)

# Remove stop words from the 'comments' column
df['comments_clean'] = df['comments'].apply(remove_stopwords)

# Display the DataFrame with stop words removed
df.head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/feliperodriguez/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/feliperodriguez/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[126]:
```
   index  listing_id        id       date  reviewer_id reviewer_name  \
0      0        3781  37776825 2015-07-10     36059247          Greg
1      1        3781  41842494 2015-08-09     10459388           Tai
2      2        3781  45282151 2015-09-01     12264652        Damien
3      3        3781  49022647 2015-09-30     41426327          Mike
4      4        3781  52503327 2015-10-30     15151513          Ivan

                                            comments  \
0  The apartment was as advertised and Frank was …
1  It was a pleasure to stay at Frank's place. Th…
2  The apartment description is entirely faithful…
3  Thoroughly enjoyed my time at Frank's home. Ha…
4  Great value for the money! This location has e…

                                      comments_clean sentiment  \
0  apartment advertised Frank incredibly helpful …  Positive
1  pleasure stay Frank 's place . place everythin…  Positive
2  apartment description entirely faithful , buil…  Positive
3  Thoroughly enjoyed time Frank 's home . amenit…  Positive
4  Great value money ! location exceeding expecta…  Positive

   sentiment_numerical
0                    2
1                    2
2                    2
3                    2
4                    2
```

[127]:
```
import re

# Function to remove special characters from text
def remove_special_characters(text):
    # Ensure text is converted to string
    text = str(text)
    # Define regex pattern to match non-alphanumeric characters
    pattern = r'[^a-zA-Z0-9\s]'
    # Replace special characters with empty string
    text = re.sub(pattern, '', text)
    return text

# Remove special characters from the 'comments' column
df['comments_clean'] = df['comments_clean'].apply(remove_special_characters)

df.head()
```

[127]:
```
   index  listing_id        id        date  reviewer_id reviewer_name  \
0      0        3781  37776825  2015-07-10     36059247          Greg
1      1        3781  41842494  2015-08-09     10459388           Tai
2      2        3781  45282151  2015-09-01     12264652        Damien
3      3        3781  49022647  2015-09-30     41426327          Mike
4      4        3781  52503327  2015-10-30     15151513          Ivan

                                            comments  \
0  The apartment was as advertised and Frank was …
1  It was a pleasure to stay at Frank's place. Th…
2  The apartment description is entirely faithful…
3  Thoroughly enjoyed my time at Frank's home. Ha…
4  Great value for the money! This location has e…

                                      comments_clean sentiment  \
0  apartment advertised Frank incredibly helpful …  Positive
1  pleasure stay Frank s place  place everything …  Positive
2  apartment description entirely faithful  build…  Positive
3  Thoroughly enjoyed time Frank s home  amenitie…  Positive
4  Great value money  location exceeding expectat…  Positive

   sentiment_numerical
0                    2
1                    2
2                    2
3                    2
4                    2
```

[128]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize the VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Function to predict sentiment label for tokenized text
```

```
def predict_sentiment(text):
    # Get the sentiment score of the text
    sentiment_score = sid.polarity_scores(text)
    # Determine the sentiment label based on the compound score
    if sentiment_score['compound'] >= 0.05:
        return 'Positive'
    elif sentiment_score['compound'] <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

# Create a new column 'sentiment' with sentiment labels for tokenized text
df['sentiment'] = df['comments_clean'].apply(predict_sentiment)
```

[129]:
```
df.head()
```

[129]:
```
   index  listing_id        id        date  reviewer_id reviewer_name  \
0      0        3781  37776825  2015-07-10     36059247          Greg
1      1        3781  41842494  2015-08-09     10459388           Tai
2      2        3781  45282151  2015-09-01     12264652        Damien
3      3        3781  49022647  2015-09-30     41426327          Mike
4      4        3781  52503327  2015-10-30     15151513          Ivan

                                            comments  \
0  The apartment was as advertised and Frank was …
1  It was a pleasure to stay at Frank's place. Th…
2  The apartment description is entirely faithful…
3  Thoroughly enjoyed my time at Frank's home. Ha…
4  Great value for the money! This location has e…

                                      comments_clean sentiment  \
0  apartment advertised Frank incredibly helpful …  Positive
1  pleasure stay Frank s place  place everything …  Positive
2  apartment description entirely faithful  build…  Positive
3  Thoroughly enjoyed time Frank s home  amenitie…  Positive
4  Great value money  location exceeding expectat…  Positive

   sentiment_numerical
0                    2
1                    2
2                    2
3                    2
4                    2
```

[130]:
```
from sklearn.preprocessing import LabelEncoder
```

```
[131]: # Initialize LabelEncoder
label_encoder = LabelEncoder()

# Encode the 'sentiment' column into numerical values
df['sentiment_numerical'] = label_encoder.fit_transform(df['sentiment'])
```

```
[132]: df['date'] = pd.to_datetime(df['date'])
```

```
[133]: df.head()
```

```
[133]:    index  listing_id        id       date  reviewer_id reviewer_name  \
0      0        3781  37776825 2015-07-10     36059247          Greg
1      1        3781  41842494 2015-08-09     10459388           Tai
2      2        3781  45282151 2015-09-01     12264652        Damien
3      3        3781  49022647 2015-09-30     41426327          Mike
4      4        3781  52503327 2015-10-30     15151513          Ivan

                                            comments  \
0  The apartment was as advertised and Frank was …
1  It was a pleasure to stay at Frank's place. Th…
2  The apartment description is entirely faithful…
3  Thoroughly enjoyed my time at Frank's home. Ha…
4  Great value for the money! This location has e…

                                      comments_clean sentiment  \
0  apartment advertised Frank incredibly helpful …  Positive
1  pleasure stay Frank s place  place everything …  Positive
2  apartment description entirely faithful  build…  Positive
3  Thoroughly enjoyed time Frank s home  amenitie…  Positive
4  Great value money  location exceeding expectat…  Positive

   sentiment_numerical
0                    2
1                    2
2                    2
3                    2
4                    2
```

```
[34]: import matplotlib.pyplot as plt
```

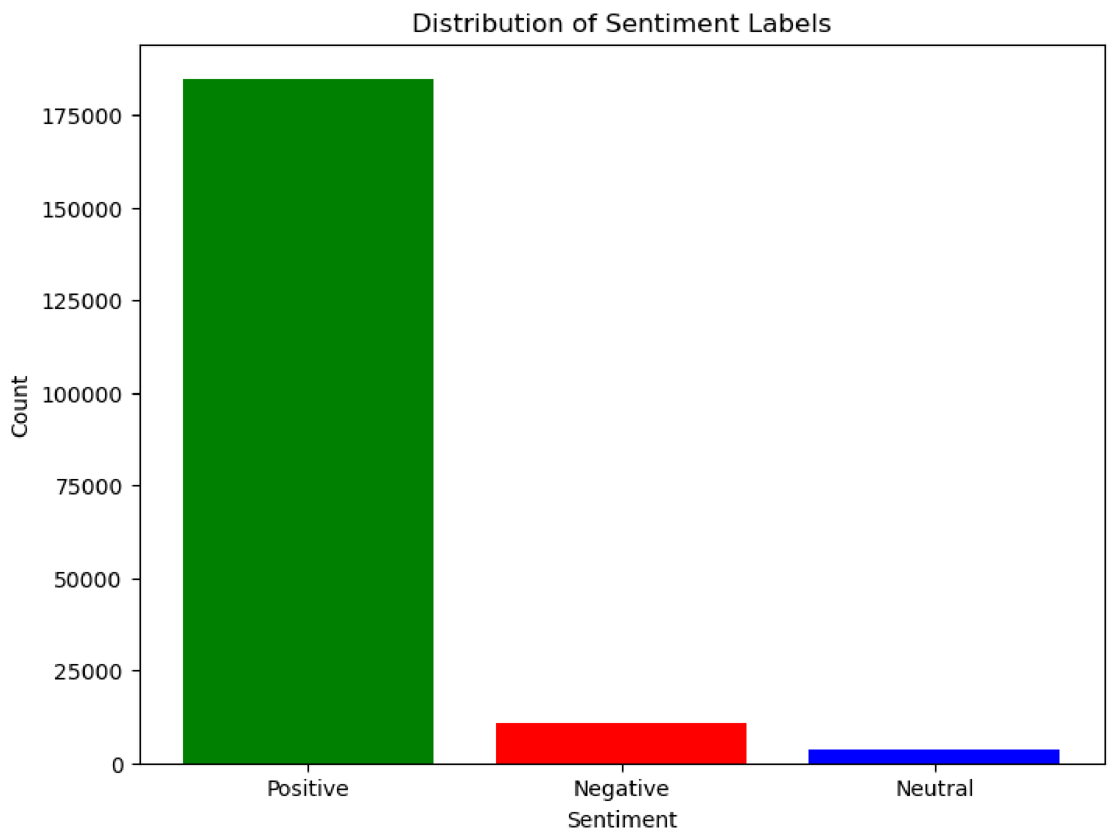
```
[134]: # Convert the 'sentiment' column to categorical data type
df['sentiment'] = df['sentiment'].astype('category')

# Perform one-hot encoding to convert the 'sentiment' column into numerical␣
 ↪columns
test = pd.get_dummies(df, columns=['sentiment'], prefix='sentiment')
```

```
columns_to_convert = ['sentiment_Negative', 'sentiment_Neutral',␣
  ↪'sentiment_Positive']
test[columns_to_convert] = test[columns_to_convert].astype(int)
```

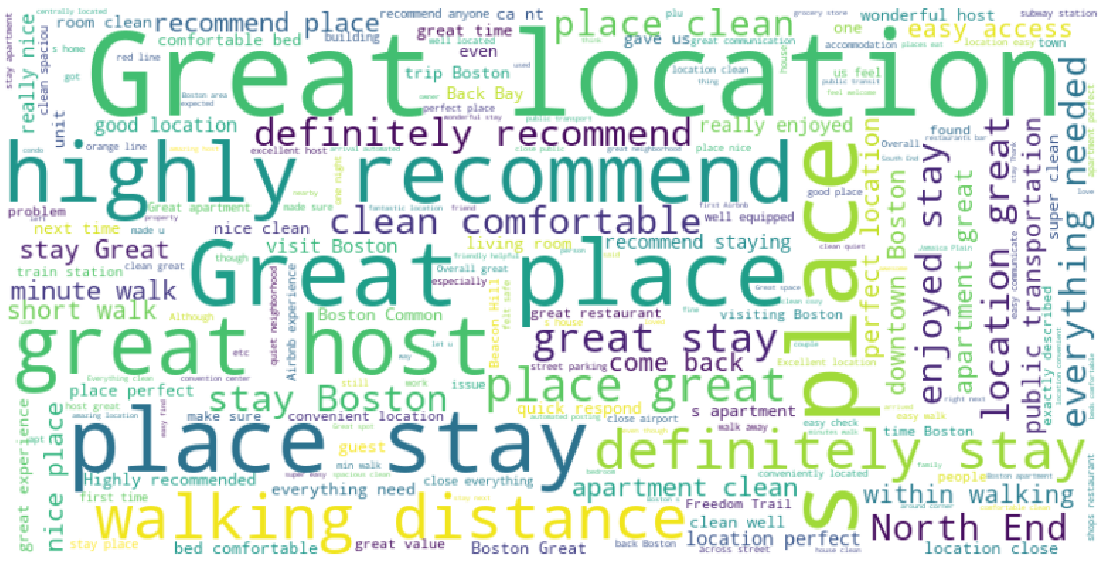
[135]:
```
test.head()
```

[135]:
```
   index  listing_id        id       date  reviewer_id reviewer_name  \
0      0        3781  37776825 2015-07-10     36059247          Greg
1      1        3781  41842494 2015-08-09     10459388           Tai
2      2        3781  45282151 2015-09-01     12264652        Damien
3      3        3781  49022647 2015-09-30     41426327          Mike
4      4        3781  52503327 2015-10-30     15151513          Ivan

                                            comments  \
0  The apartment was as advertised and Frank was …
1  It was a pleasure to stay at Frank's place. Th…
2  The apartment description is entirely faithful…
3  Thoroughly enjoyed my time at Frank's home. Ha…
4  Great value for the money! This location has e…

                                      comments_clean  sentiment_numerical  \
0  apartment advertised Frank incredibly helpful …                    2
1  pleasure stay Frank s place  place everything …                    2
2  apartment description entirely faithful  build…                    2
3  Thoroughly enjoyed time Frank s home  amenitie…                    2
4  Great value money  location exceeding expectat…                    2

   sentiment_Negative  sentiment_Neutral  sentiment_Positive
0                   0                  0                   1
1                   0                  0                   1
2                   0                  0                   1
3                   0                  0                   1
4                   0                  0                   1
```

[136]:
```
# Sample data
sentiment_counts = test['sentiment_numerical'].value_counts()

# Create a bar plot
plt.figure(figsize=(8, 6))
plt.bar(['Positive', 'Negative', 'Neutral'], sentiment_counts, color=['green',␣
  ↪'red', 'blue'])
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Distribution of Sentiment Labels')
plt.show()
```



[138]:
```
from wordcloud import WordCloud
# Sample data
comments_clean = df['comments_clean'].astype(str).str.cat(sep=' ')

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
  ↪generate(comments_clean)

# Display the word cloud image
plt.figure(figsize=(10, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



```
[139]: from sklearn.feature_extraction.text import CountVectorizer
       from sklearn.model_selection import train_test_split
       from sklearn.naive_bayes import MultinomialNB
       from sklearn.metrics import accuracy_score

       test['comments'] = test['comments'].fillna('')

       # Split data into training and testing sets
       X = test['comments']
       y = test['sentiment_numerical']
       X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
        ↪random_state=42)

       # Vectorize comments
       vectorizer = CountVectorizer()
       X_train_counts = vectorizer.fit_transform(X_train)
       X_test_counts = vectorizer.transform(X_test)

       # Train Naive Bayes classifier
       clf = MultinomialNB()
       clf.fit(X_train_counts, y_train)

       # Predict sentiment on test data
       y_pred = clf.predict(X_test_counts)

       # Calculate accuracy
       accuracy = accuracy_score(y_test, y_pred)
       print("Accuracy:", accuracy)
```

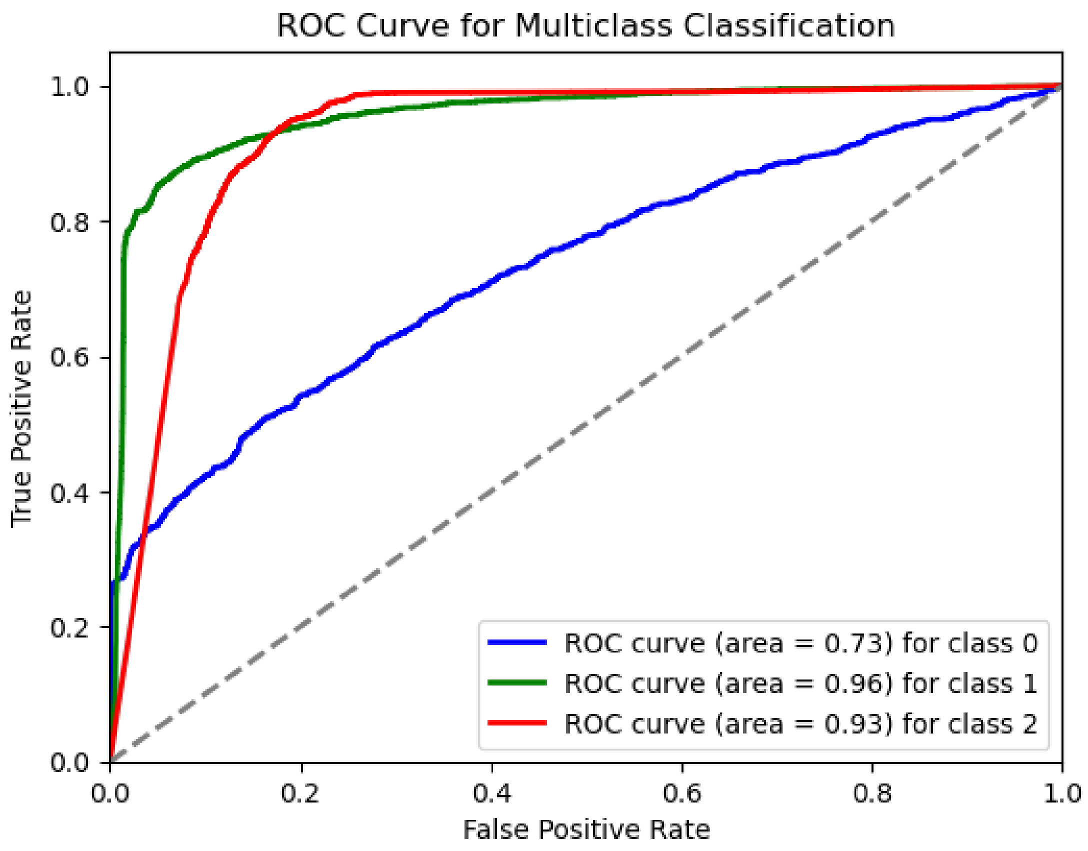
```
Accuracy: 0.9565566772136005
```

[140]:
```python
from sklearn.metrics import classification_report, confusion_matrix

# Calculate additional evaluation metrics
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.25      0.37       763
           1       0.71      0.63      0.67      2127
           2       0.97      0.99      0.98     36932

    accuracy                           0.96     39822
   macro avg       0.79      0.62      0.67     39822
weighted avg       0.95      0.96      0.95     39822


Confusion Matrix:
[[  194   206   363]
 [   44  1332   751]
 [   41   325 36566]]
```

[146]:
```python
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc

# Binarize the labels for each class
y_bin = label_binarize(y_test, classes=[0, 1, 2])

# Get predicted probabilities for each class
y_prob = clf.predict_proba(X_test_counts)

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(3):  # 3 classes: Negative, Neutral, Positive
    fpr[i], tpr[i], _ = roc_curve(y_bin[:, i], y_prob[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot ROC curve for each class
plt.figure()
```

```
colors = ['blue', 'green', 'red']
for i, color in zip(range(3), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2, label='ROC curve (area = %0.2f)␣
 ↪for class %d' % (roc_auc[i], i))
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Multiclass Classification')
plt.legend(loc="lower right")
plt.show()
```


ROC Curve for Multiclass Classification
True Positive Rate
False Positive Rate
ROC curve (area = 0.73) for class 0
ROC curve (area = 0.96) for class 1
ROC curve (area = 0.93) for class 2