s-review-eda-sentiment-analysis-1

April 21, 2024

1 Amazon Books Review Analysis

1.1 Importing Dependencies

[2]:
```
!pip install vaderSentiment
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from wordcloud import WordCloud
```

```
Collecting vaderSentiment
  Obtaining dependency information for vaderSentiment from https://files.pythonh
osted.org/packages/76/fc/310e16254683c1ed35eeb97386986d6c00bc29df17ce280aed64d55
537e9/vaderSentiment-3.3.2-py2.py3-none-any.whl.metadata
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl.metadata (572 bytes)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-
packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests->vaderSentiment) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from requests->vaderSentiment) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests->vaderSentiment)
(1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests->vaderSentiment)
(2023.11.17)
Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
                         126.0/126.0 kB
3.7 MB/s eta 0:00:00
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

1.2 Reading Data from file

[3]:
```
br = pd.read_csv(r"/kaggle/input/amazon-books-reviews/Books_rating.csv")
bd = pd.read_csv(r"/kaggle/input/amazon-books-reviews/books_data.csv")
```

[4]:
```
br.head(5)
```

[4]:
```
           Id                            Title  Price         User_id  \
0  1882931173  Its Only Art If Its Well Hung!    NaN   AVCGYZL8FQQTD
1  0826414346        Dr. Seuss: American Icon    NaN  A30TK6U7DNS82R
2  0826414346        Dr. Seuss: American Icon    NaN  A3UH4UZ4RSVO82
3  0826414346        Dr. Seuss: American Icon    NaN  A2MVUWT453QH61
4  0826414346        Dr. Seuss: American Icon    NaN  A22X4XUPKF66MR

                       profileName review/helpfulness  review/score  \
0              Jim of Oz "jim-of-oz"                7/7           4.0
1                      Kevin Killian              10/10           5.0
2                       John Granger              10/11           5.0
3  Roy E. Perry "amateur philosopher"                7/7           4.0
4     D. H. Richards "ninthwavestore"                3/3           4.0

   review/time                                  review/summary  \
0    940636800           Nice collection of Julie Strain images
1   1095724800                                Really Enjoyed It
2   1078790400  Essential for every personal and Public Library
3   1090713600  Phlip Nel gives silly Seuss a serious treatment
4   1107993600                           Good academic overview

                                         review/text
0  This is only for Julie Strain fans. It's a col…
1  I don't care much for Dr. Seuss but after read…
2  If people become the books they read and if "t…
3  Theodore Seuss Geisel (1904-1991), aka &quot;D…
4  Philip Nel - Dr. Seuss: American IconThis is b…
```

[5]:
```
bd.head(5)
```

[5]:
```
                                          Title  \
0                   Its Only Art If Its Well Hung!
1                        Dr. Seuss: American Icon
2            Wonderful Worship in Smaller Churches
3                     Whispers of the Wicked Saints
4  Nation Dance: Religion, Identity and Cultural …

                                         description              authors  \
0                                                NaN     ['Julie Strain']
1  Philip Nel takes a fascinating look into the k…       ['Philip Nel']
```

```
2  This resource includes twelve principles in un…     ['David R. Ray']
3  Julia Thomas finds her life spinning out of co…  ['Veronica Haddon']
4                                                NaN      ['Edward Long']

                                                image  \
0  http://books.google.com/books/content?id=DykPA…
1  http://books.google.com/books/content?id=IjvHQ…
2  http://books.google.com/books/content?id=2tsDA…
3  http://books.google.com/books/content?id=aRSIg…
4                                                NaN

                                         previewLink  publisher publishedDate  \
0  http://books.google.nl/books?id=DykPAAAACAAJ&d…         NaN           1996
1  http://books.google.nl/books?id=IjvHQsCn_pgC&p…   A&C Black    2005-01-01
2  http://books.google.nl/books?id=2tsDAAAACAAJ&d…         NaN           2000
3  http://books.google.nl/books?id=aRSIgJlq6JwC&d…   iUniverse       2005-02
4  http://books.google.nl/books?id=399SPgAACAAJ&d…         NaN    2003-03-01

                                           infoLink  \
0  http://books.google.nl/books?id=DykPAAAACAAJ&d…
1  http://books.google.nl/books?id=IjvHQsCn_pgC&d…
2  http://books.google.nl/books?id=2tsDAAAACAAJ&d…
3  http://books.google.nl/books?id=aRSIgJlq6JwC&d…
4  http://books.google.nl/books?id=399SPgAACAAJ&d…

                     categories  ratingsCount
0    ['Comics & Graphic Novels']           NaN
1  ['Biography & Autobiography']           NaN
2                   ['Religion']           NaN
3                    ['Fiction']           NaN
4                            NaN           NaN
```

1.3 Data Pre-Processing

1.3.1 Merging both the dataset

[6]:
```
books = pd.merge(br,bd, on = 'Title')
books.shape
```

[6]:
```
(3000000, 19)
```

1.3.2 extracting useful columns

[7]:
```
df = books[['Title','review/score','review/
  ↪text','authors','categories','ratingsCount']]
```

1.3.3 Dropping Duplicates

[8]:
```
df.drop_duplicates(inplace = True)
df.shape
```

```
/tmp/ipykernel_42/3208856916.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop_duplicates(inplace = True)
```

[8]:
```
(2617289, 6)
```

[9]:
```
df.isna().sum()
```

[9]:
```
Title              207
review/score         0
review/text          8
authors         350198
categories      465848
ratingsCount   1234897
dtype: int64
```

1.3.4 Dropping Null Values

[10]:
```
df.dropna(inplace = True)
df.isna().sum()
```

```
/tmp/ipykernel_42/1508174486.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.dropna(inplace = True)
```

[10]:
```
Title          0
review/score   0
review/text    0
authors        0
categories     0
ratingsCount   0
dtype: int64
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1325891 entries, 47 to 2999999
```

```
Data columns (total 6 columns):
 #   Column        Non-Null Count    Dtype
---  ------        --------------    -----
 0   Title         1325891 non-null  object
 1   review/score  1325891 non-null  float64
 2   review/text   1325891 non-null  object
 3   authors       1325891 non-null  object
 4   categories    1325891 non-null  object
 5   ratingsCount  1325891 non-null  float64
dtypes: float64(2), object(4)
memory usage: 70.8+ MB
```

[12]:
```
df.shape
```

[12]:
```
(1325891, 6)
```

1.3.5 Using only the fraction of dataset to reduce the running time on big data

[13]:
```
data = df.sample(15000)
data.shape
```

[13]:
```
(15000, 6)
```

1.3.6 Removing brackets and colons from authors name

[14]:
```
data['authors'] = data['authors'].str.extract(r'\'(.*)\'')
```

1.3.7 Removes brackets and colons from categories

[15]:
```
data['categories'] = data['categories'].str.extract(r'\'(.*)\'')
```

1.3.8 Counting the length of each review

[16]:
```
data['word_count'] = data['review/text'].apply(lambda x: len(x.split(' ')))
```

[17]:
```
data.head()
```

[17]:
```
                       Title  review/score  \
1123055          The Assassin           5.0
1180512      The Book of Ruth           3.0
28403          Fahrenheit 451           2.0
2280455             Odd Thomas           5.0
683668   The Dolphins of Pern           4.0

                                                review/text         authors  \
1123055  I have to say, this is one of the best suspens…   Matthew Quirk
```

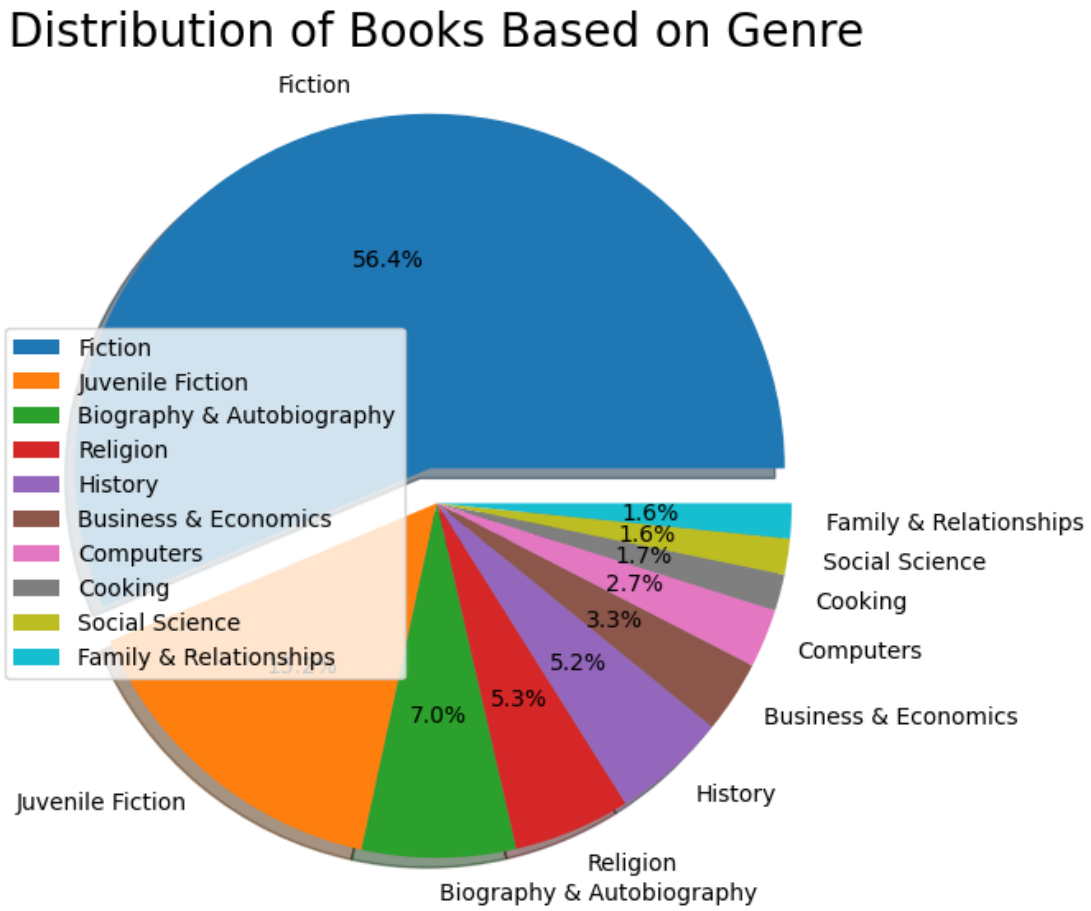
```
1180512  It was an ok book. I read it last week, and I …   Jane Hamilton
28403    If you look at Farenheit 451 from the perspect…    Ray Bradbury
2280455  This was my first exposure to Dean Koontz and …     Dean Koontz
683668   i really liked the book. but my only problem i…  Anne McCaffrey

           categories  ratingsCount  word_count
1123055       Fiction           2.0          99
1180512       Fiction          39.0          82
28403    Book burning           1.0         214
2280455       Fiction         168.0          93
683668        Fiction          12.0          43
```

[18]:
```
## data.to_csv('sample.csv', index=False)
```

2 EDA

2.1 Distribution of Books in Market based on Genre

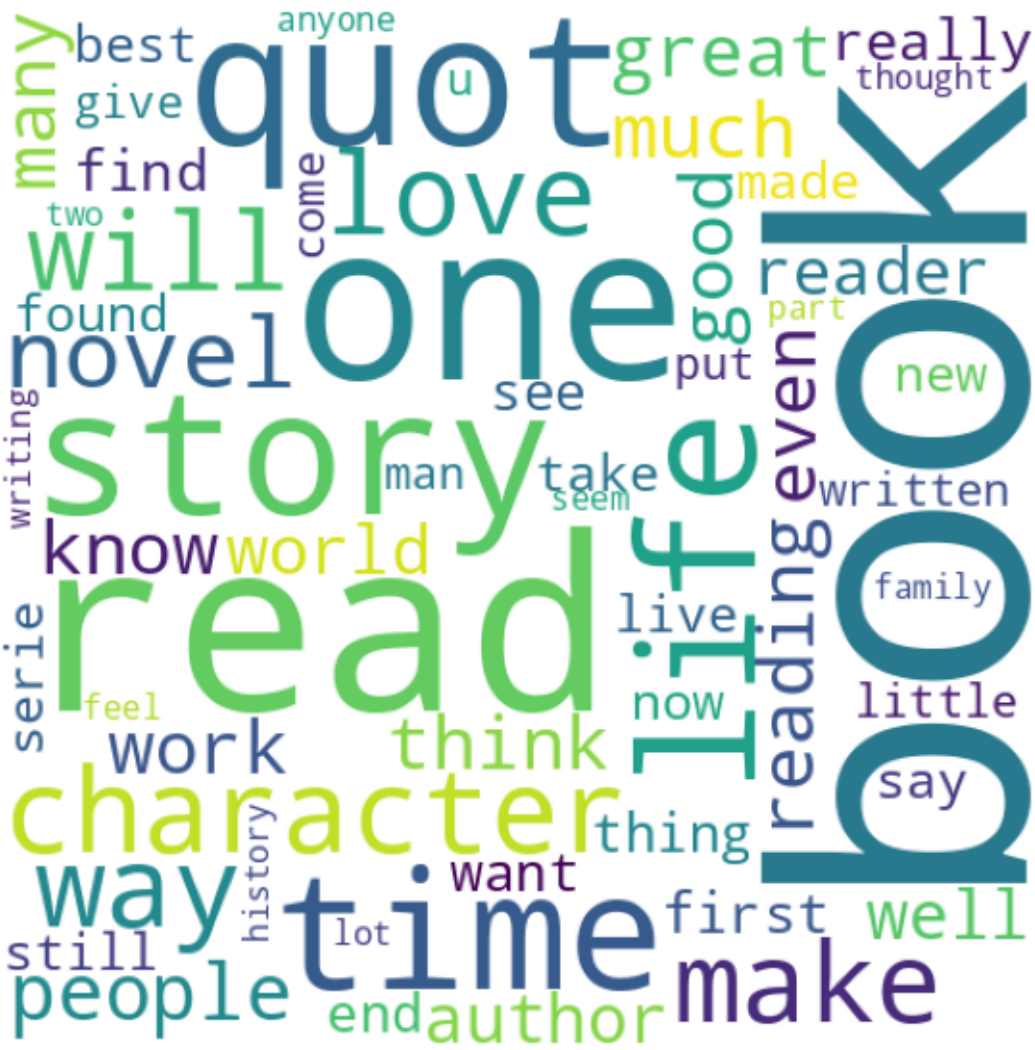
[19]:
```
plt.figure(figsize=(7,7))
labels=['Fiction','Juvenile Fiction','Biography &␣
 ↪Autobiography','Religion','History','Business &␣
 ↪Economics','Computers','Cooking','Social Science','Family & Relationships']
plt.pie(data['categories'].value_counts().head(10),explode=(0.
 ↪1,0,0,0,0,0,0,0,0,0), labels=labels,autopct='%1.1f%%', shadow=True)
plt.title('Distribution of Books Based on Genre', fontsize = 20)
plt.axis('off')
plt.legend()
plt.show()
```



2.2 Most frequent Words in over 3 rated Reviews

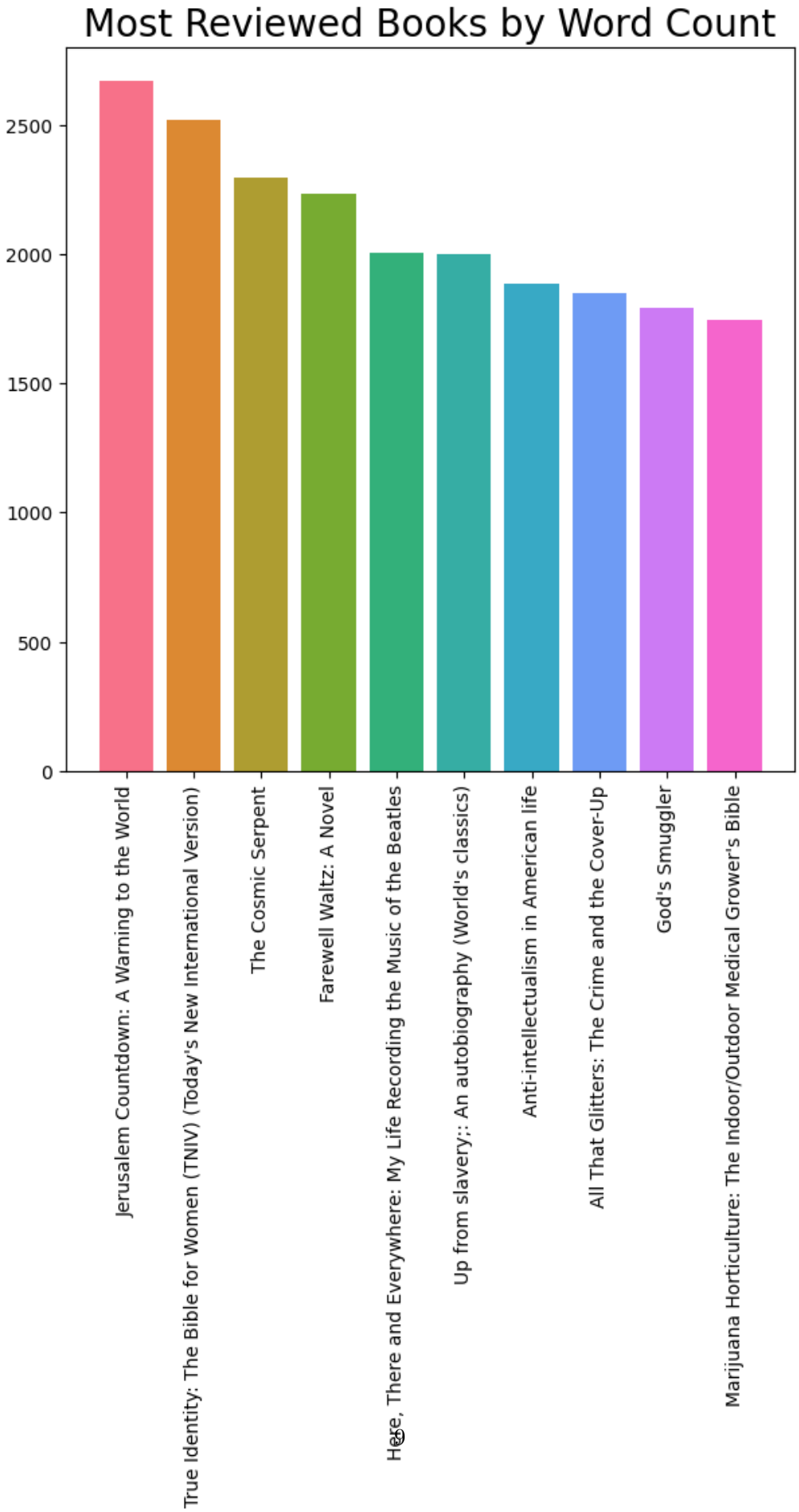
[20]:
```
from wordcloud import WordCloud

wc = WordCloud(width=500,height=500,min_font_size=15,background_color='white')
spam_wc = wc.generate(data[data['review/score'] > 3]['review/text'].str.
 ↪cat(sep=" "))

plt.figure(figsize=(7,7))
plt.axis('off')
plt.imshow(spam_wc)
```

[20]:
```
<matplotlib.image.AxesImage at 0x78642b137610>
```



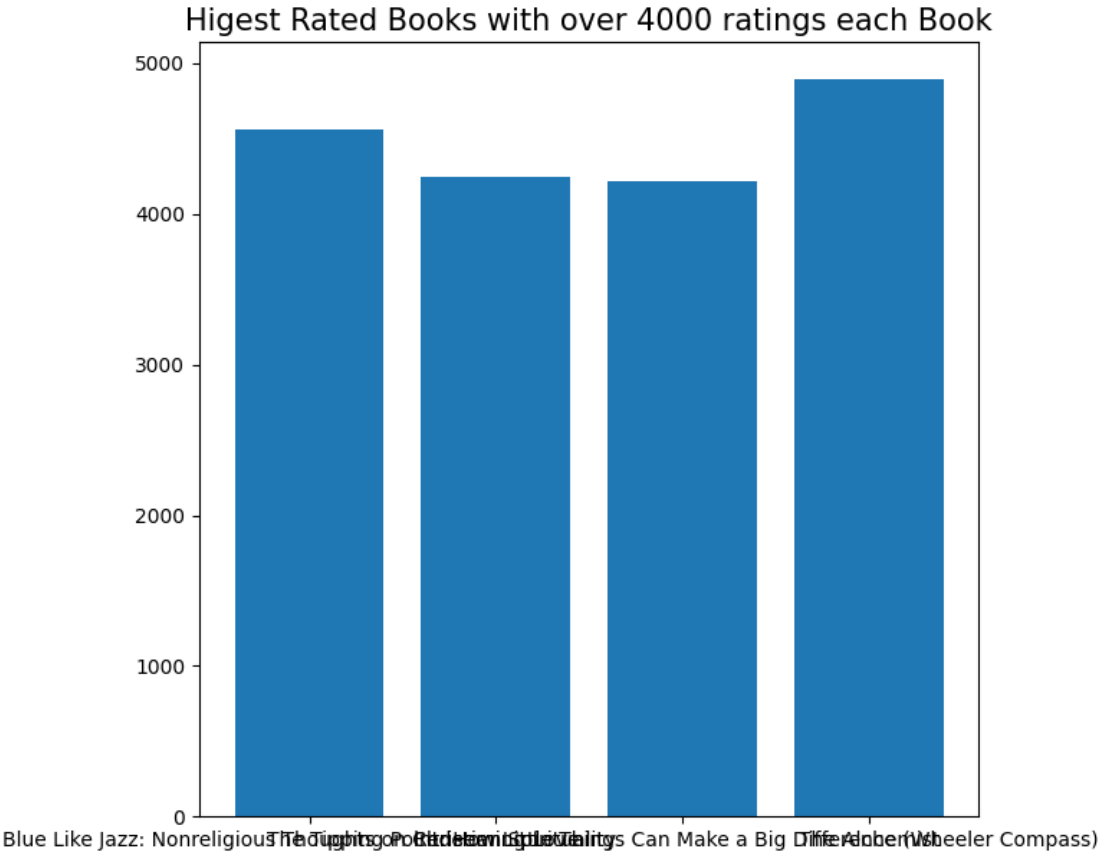
2.3 Most Reviewed Books

[21]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(7, 7))
cou = data[data['word_count'] > 1707][['Title', 'word_count']].
 ↪sort_values(by='word_count', ascending=False)
colors = sns.color_palette('husl', n_colors=len(cou))
bars = plt.bar(cou['Title'], cou['word_count'], color=colors)
plt.title('Most Reviewed Books by Word Count', fontsize=20)
plt.xticks(rotation=90)
plt.show()
```


Most Reviewed Books by Word Count
2500
2000
1500
1000
500
0
Jerusalem Countdown: A Warning to the World
True Identity: The Bible for Women (TNIV) (Today's New International Version)
The Cosmic Serpent
Farewell Waltz: A Novel
Here, There and Everywhere: My Life Recording the Music of the Beatles
Up from slavery;: An autobiography (World's classics)
Anti-intellectualism in American life
All That Glitters: The Crime and the Cover-Up
God's Smuggler
Marijuana Horticulture: The Indoor/Outdoor Medical Grower's Bible

2.4 Highest Rated Books with over 4000 ratings each Book

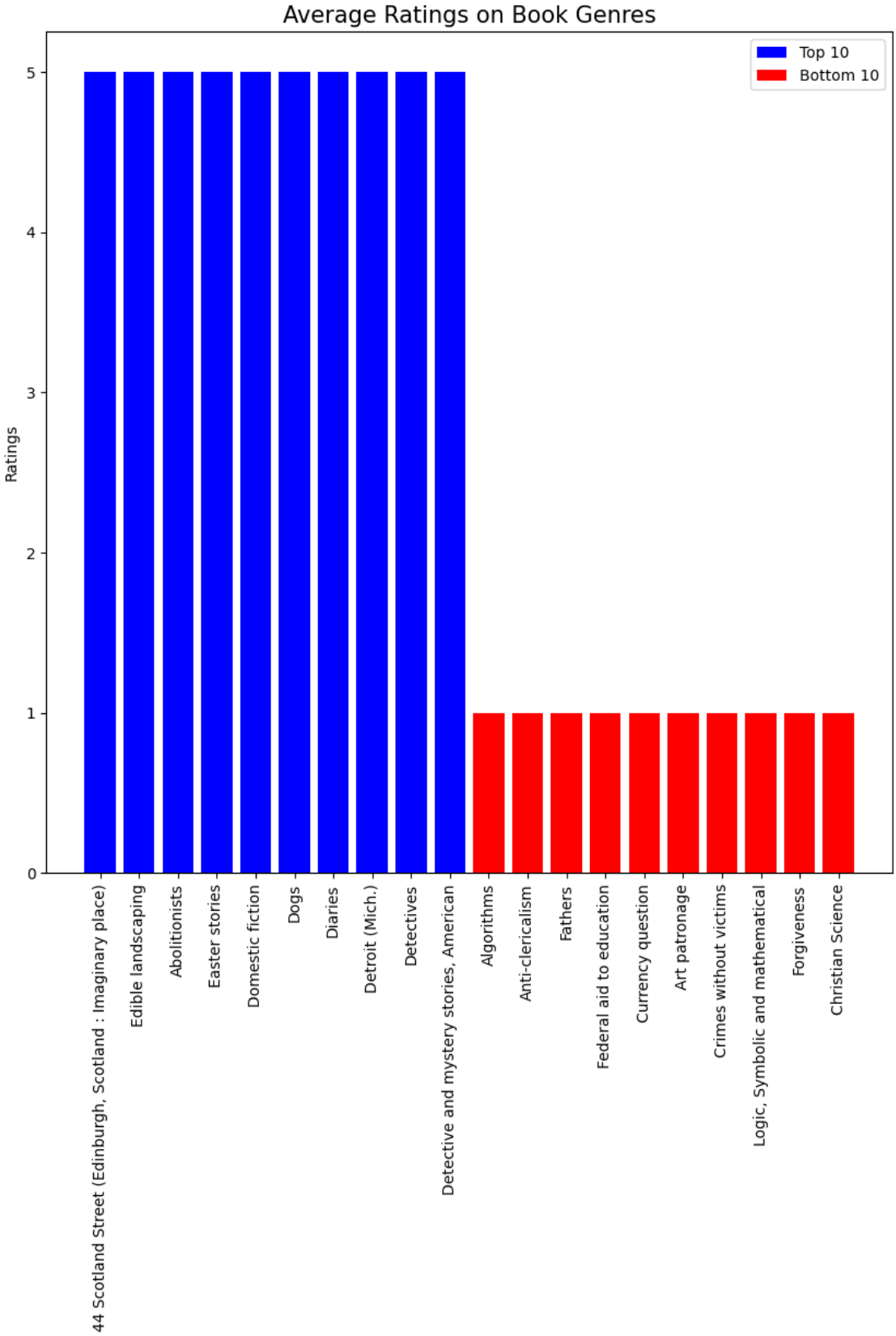
[22]:
```
plt.figure(figsize = (7, 7))
rating_counts = data[data['ratingsCount'] > 4000][['Title','ratingsCount']].
 ↪drop_duplicates()
plt.bar(rating_counts['Title'],rating_counts['ratingsCount'])
plt.title('Higest Rated Books with over 4000 ratings each Book', fontsize = 15)
plt.show()
```



2.5 In which Genres readers give Positive and Negative Ratings

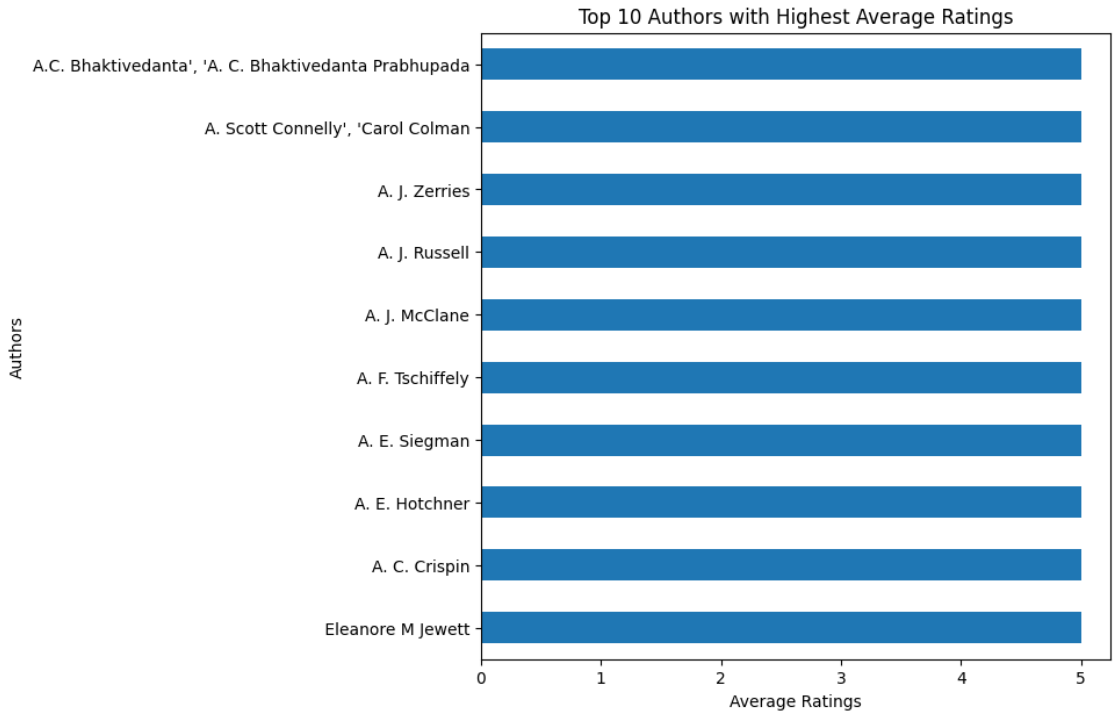
[23]:
```
# Convert 'review/score' column to numeric (if applicable)
data['review/score'] = pd.to_numeric(data['review/score'], errors='coerce')

# Filter out non-numeric values (if any)
numeric_data = data.dropna(subset=['review/score'])
```

```
# Group by 'categories' and compute the mean of 'review/score'
avg_cat_rating = numeric_data.groupby('categories')['review/score'].mean().
 ↪sort_values(ascending=False).head(10)
avg_cat_rating_d = numeric_data.groupby('categories')['review/score'].mean().
 ↪sort_values().head(10)

# Plot the bar chart for top and bottom categories
plt.figure(figsize=(10, 10))
plt.bar(avg_cat_rating.index, avg_cat_rating, color='blue', label='Top 10')
plt.bar(avg_cat_rating_d.index, avg_cat_rating_d, color='red', label='Bottom␣
 ↪10')
plt.title('Average Ratings on Book Genres', fontsize=15)
plt.xticks(rotation='vertical')
plt.ylabel('Ratings')
plt.legend()
plt.show()
```


Average Ratings on Book Genres
Top 10
Bottom 10
Ratings
0
1
2
3
4
5
44 Scotland Street (Edinburgh, Scotland : Imaginary place)
Edible landscaping
Abolitionists
Easter stories
Domestic fiction
Dogs
Diaries
Detroit (Mich.)
Detectives
Detective and mystery stories, American
Algorithms
Anti-clericalism
Fathers
Federal aid to education
Currency question
Art patronage
Crimes without victims
Logic, Symbolic and mathematical
Forgiveness
Christian Science

2.6 Top 10 Authors with 5 star Ratings

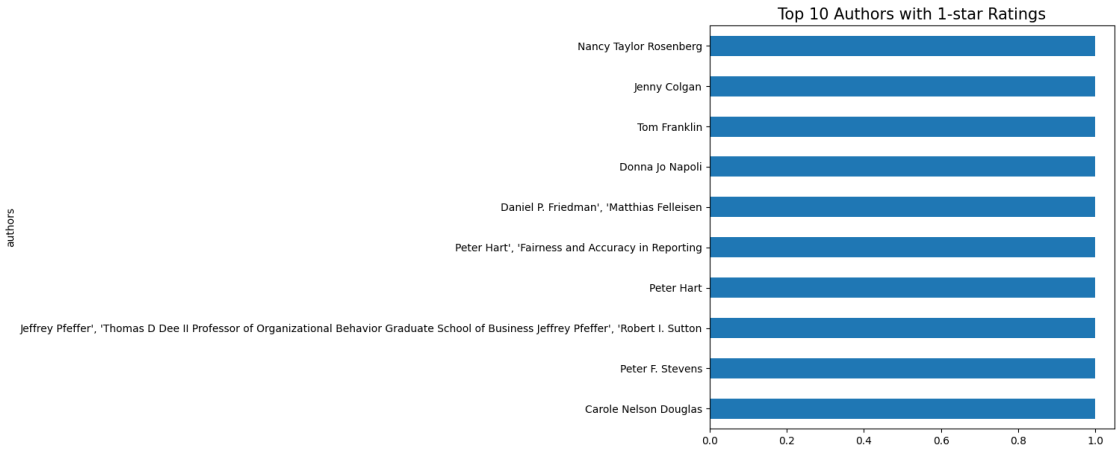
[24]:
```
# Convert 'review/score' column to numeric, coercing errors to NaN
data['review/score'] = pd.to_numeric(data['review/score'], errors='coerce')

# Group by 'authors', compute the mean of 'review/score', and plot the top 10␣
 ↪authors
top_authors = data.groupby('authors')['review/score'].mean().nlargest(10)
top_authors.plot(kind='barh', figsize=(7, 7))
plt.title('Top 10 Authors with Highest Average Ratings')
plt.xlabel('Average Ratings')
plt.ylabel('Authors')
plt.show()
```



2.7 Top 10 Authors with 1 star Ratings

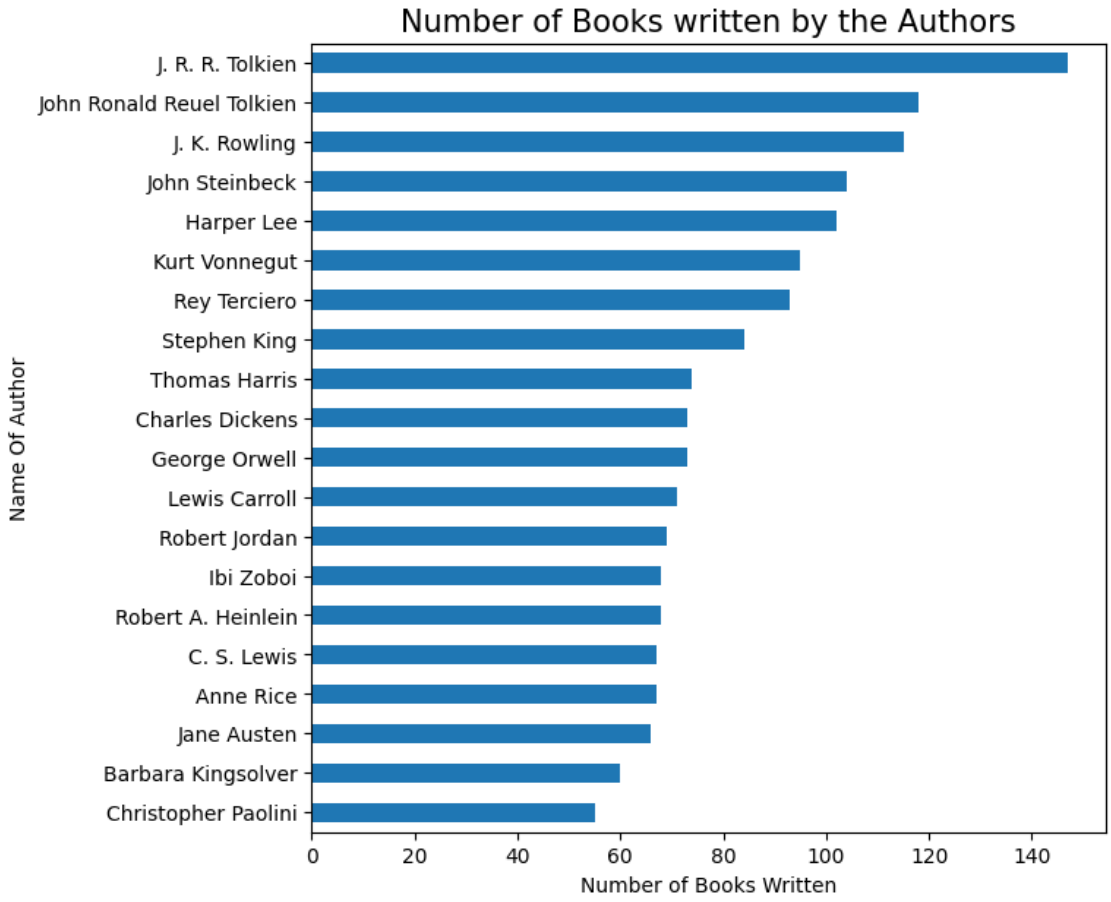
[25]:
```
# Grouping the data by authors and calculating the mean review score for each␣
 ↪author
average_scores_by_author = data.groupby('authors')['review/score'].mean()

# Sorting the authors based on their average review scores and selecting the␣
 ↪bottom 10 (lowest scores)
```

```
bottom_10_authors = average_scores_by_author.sort_values(ascending=True).
 ↪head(10)

# Creating a horizontal bar plot to show the top 10 authors with the lowest␣
 ↪average review scores
bottom_10_authors.plot(kind='barh', figsize=(7,7))

# Setting the title for the plot
plt.title('Top 10 Authors with 1-star Ratings', fontsize=15)

# Displaying the plot
plt.show()
```



2.8 Number of Books written by each Author

[26]:
```
data['authors'].value_counts().head(20).sort_values(ascending = True).
 ↪plot(kind='barh', figsize=(7,7))
plt.title('Number of Books written by the Authors', fontsize = 15)
plt.ylabel('Name Of Author')
plt.xlabel('Number of Books Written')
plt.show()
```



3 Sentiment Analysis

[27]:
```
pip install vaderSentiment
```

```
Requirement already satisfied: vaderSentiment in /opt/conda/lib/python3.10/site-
packages (3.3.2)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-
packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests->vaderSentiment) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from requests->vaderSentiment) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests->vaderSentiment)
(1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests->vaderSentiment)
(2023.11.17)
```

Note: you may need to restart the kernel to use updated packages.

[28]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
vader = SentimentIntensityAnalyzer()
```

[29]:
```
data.head(2)
```

[29]:
```
                    Title  review/score  \
1123055      The Assassin           5.0
1180512  The Book of Ruth           3.0

                                                review/text        authors  \
1123055  I have to say, this is one of the best suspens…  Matthew Quirk
1180512  It was an ok book. I read it last week, and I …  Jane Hamilton

        categories  ratingsCount  word_count
1123055    Fiction           2.0          99
1180512    Fiction          39.0          82
```

[30]:
```
# Lower casing the reviews
data['clean_reviews'] = data['review/text'].str.lower()
# Calculating Polarity score of reviews
data['score'] = data['clean_reviews'].apply(lambda review: vader.
 ↪polarity_scores(review))
# Extracting compound column
data['compound']  = data['score'].apply(lambda score_dict:␣
 ↪score_dict['compound'])
```

[31]:
```
# if compound value is more than 0.05 then it precive as positive
# if compound value is less than -0.0 then it precive as negative
# if compound value is equal to 0.0 then it precive as neutral

data['Sentiment'] = data['compound'].apply(lambda x: 'positive' if x >= 0.05␣
 ↪else 'negative' if x < -0.05 else 'neutral')
data.head()
```

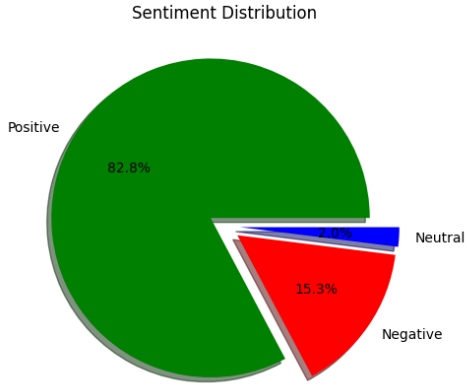
[31]:
```
                        Title  review/score  \
1123055           The Assassin           5.0
1180512       The Book of Ruth           3.0
28403           Fahrenheit 451           2.0
2280455             Odd Thomas           5.0
683668    The Dolphins of Pern           4.0

                                                 review/text         authors  \
1123055  I have to say, this is one of the best suspens…   Matthew Quirk
1180512  It was an ok book. I read it last week, and I …   Jane Hamilton
28403    If you look at Farenheit 451 from the perspect…    Ray Bradbury
```

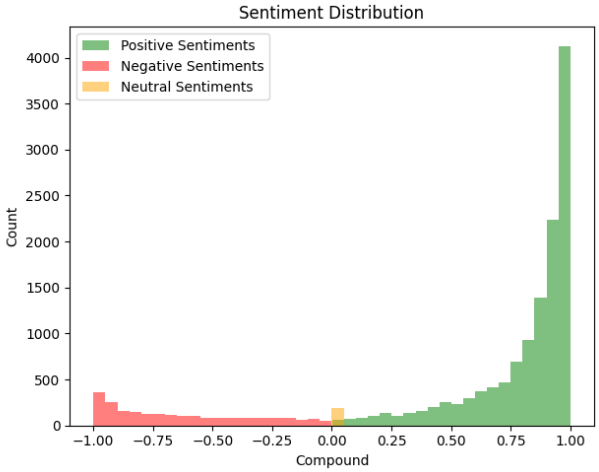
```
2280455  This was my first exposure to Dean Koontz and …     Dean Koontz
683668   i really liked the book. but my only problem i…  Anne McCaffrey

           categories  ratingsCount  word_count  \
1123055       Fiction           2.0          99
1180512       Fiction          39.0          82
28403    Book burning           1.0         214
2280455       Fiction         168.0          93
683668        Fiction          12.0          43

                                                clean_reviews  \
1123055  i have to say, this is one of the best suspens…
1180512  it was an ok book. i read it last week, and i …
28403    if you look at farenheit 451 from the perspect…
2280455  this was my first exposure to dean koontz and …
683668   i really liked the book. but my only problem i…

                                                     score  compound Sentiment
1123055  {'neg': 0.012, 'neu': 0.812, 'pos': 0.175, 'co…    0.9609  positive
1180512  {'neg': 0.139, 'neu': 0.804, 'pos': 0.057, 'co…   -0.7554  negative
28403    {'neg': 0.151, 'neu': 0.76, 'pos': 0.088, 'com…   -0.9468  negative
2280455  {'neg': 0.042, 'neu': 0.803, 'pos': 0.155, 'co…    0.8977  positive
683668   {'neg': 0.122, 'neu': 0.656, 'pos': 0.222, 'co…    0.7758  positive
```

3.1 Distribution of Negative, Neutral and Positive Sentiment in whole corpus

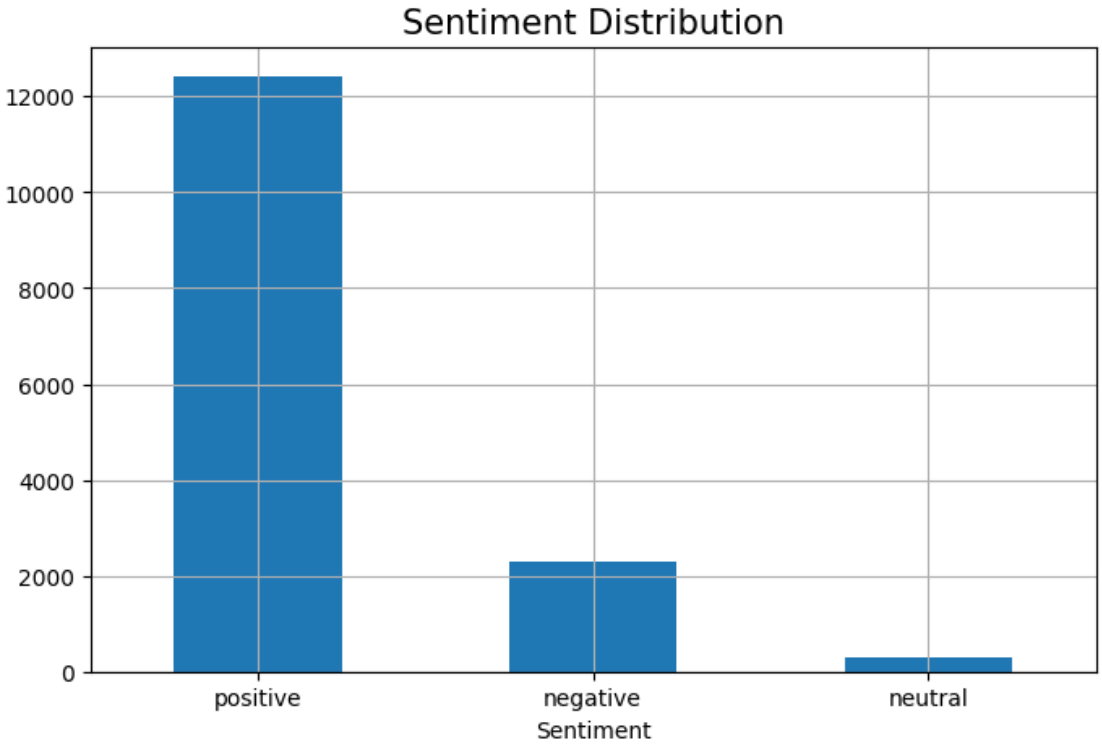
[32]:
```python
import matplotlib.pyplot as plt

# Create a figure and subplots with 1 row and 2 columns
plt.figure(figsize=(12, 5))

# Plot 1 - Pie chart for sentiment distribution
plt.subplot(1, 2, 1)
labels = ['Positive', 'Negative', 'Neutral']
sizes = data['Sentiment'].value_counts()
colors = ['green', 'red', 'blue']
explode = (0.1, 0.1, 0.1)

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.
  ↪1f%%', shadow=True)
plt.title('Sentiment Distribution')

# Plot 2 - Histogram for sentiment distribution
plt.subplot(1, 2, 2)
positive = data[data['compound'] > 0]['compound']
negative = data[data['compound'] < 0]['compound']
neutral = data[data['compound'] == 0]['compound']
```

```
sentiments = [positive, negative, neutral]
colors = ['green', 'red', 'orange']
labels = ['Positive Sentiments', 'Negative Sentiments', 'Neutral Sentiments']

for sentiment, color, label in zip(sentiments, colors, labels):
    plt.hist(sentiment, bins=20, color=color, alpha=0.5, label=label)

plt.title('Sentiment Distribution')
plt.xlabel('Compound')
plt.ylabel('Count')
plt.legend()

plt.tight_layout()
plt.show()
```





[33]:
```
data['Sentiment'].value_counts().plot(kind = 'bar', figsize = (8,5))
plt.xticks(rotation = 'horizontal')
plt.title('Sentiment Distribution',fontsize = 15)
plt.grid()
plt.show()
```



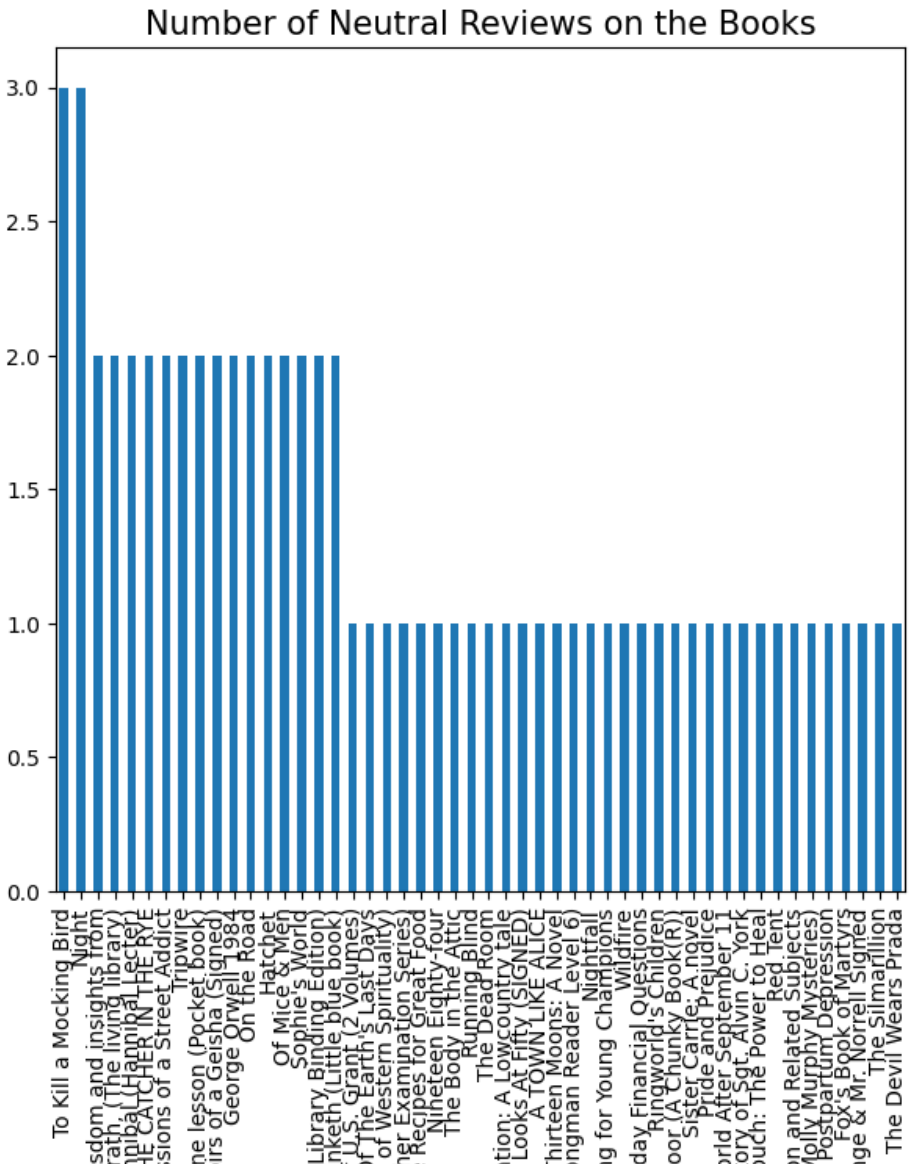
3.2 Most number of Positive Reviews on the Books

[34]:
```
data[data['Sentiment'] == 'positive']['Title'].value_counts().head(50).
 ↪plot(kind = 'bar', figsize = (7,7))
plt.title('Number of Positive Reviews on the Books',fontsize = 15)
plt.xticks(rotation = 90)
plt.show()
```

Number of Positive Reviews on the Books
0
10
20
30
40
50
60
Harry Potter and The Sorcerer's Stone
The Giver
The Hobbit There and Back Again
The Hobbit
Harry Potter and the Chamber of Secrets
Memoirs of a Geisha (Signed)
Holes
The Hobbit or There and Back Again
Rich Dad, Poor Dad
The Catcher in the Rye [Audiobook] [Cd] [Unabridged] (Audio CD)
ERAGON: INHERITANCE, BOOK ONE
The Lord of the Rings Trilogy 3 Volumes
Harry Potter & the Prisoner of Azkaban
The Lord of the Rings (3 Volume Set)
Left Behind - A Novel Of The Earth's Last Days
To Kill a Mockingbird
Hannibal (Hannibal Lecter)
Pride & Prejudice (Classic Library)
The Lord of the Rings, The Two Towers, The Return of the King, I_II_III)
The Lord Of The Rings THREE VOLUME BOXED SET (The Fellowship Of The Ring, The Two Towers, The Return Of The King, The Two towers)
THE CATCHER IN THE RYE
Pride & Prejudice (Penguin Classics)
Who Moved My Cheese? An Amazing Way to Deal with Change in Your Work and in Your Life
Redeeming Love
NEW EARTH
The Lord of the Rings - Boxed Set
Five People You Meet in Heaven
To Kill a Mocking Bird
The poisonwood Bible a Novel
Nineteen Eighty-four
Gulliver's Travels
Outsiders
Of Mice & Men
The Red Tent (Bestselling Backlist)
Pride & Prejudice (New Windmill)
The Stand. The Complete and Uncut Edition
Cold Mountain A Novel
I know this much is true
The Outsiders
George Orwell 1984
To Kill A Mockingbird
Pride and Prejudice
Of Mice And Men
Jane Eyre (Signet classics)
Wuthering Heights (Signet classics)
The Five Love Languages: The Secret to Love that Lasts
The Clan of the Cave Bear
Path of Daggers (Wheel of Time 08)
The Other Boleyn Girl
Love You Forever
Angela's Ashes (Turtleback School & Library Binding Edition)
Title

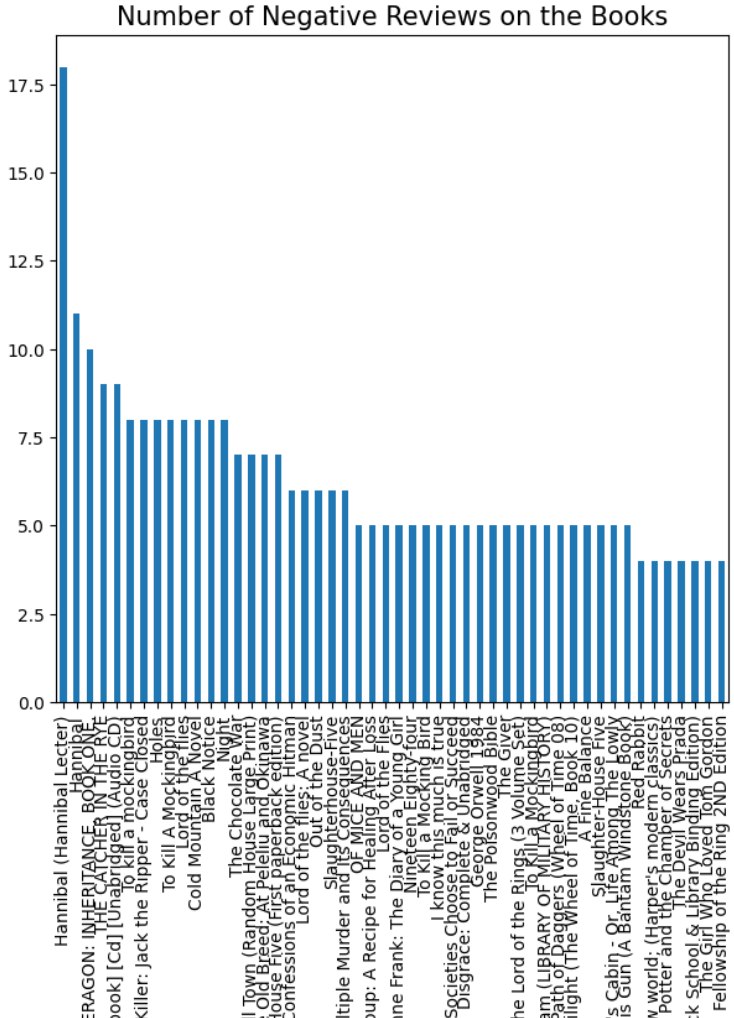
3.3 Most number of Neutral Reviews on the Books

[35]:
```
data[data['Sentiment'] == 'neutral']['Title'].value_counts().head(50).plot(kind␣
  ↪= 'bar', figsize = (7,7))
plt.title('Number of Neutral Reviews on the Books',fontsize = 15)
plt.xticks(rotation = 90)
plt.show()
```


Number of Neutral Reviews on the Books
3.0
2.5
2.0
1.5
1.0
0.5
0.0
To Kill a Mocking Bird
Night
The 7 habits of highly effective people: Wisdom and insights from
The grapes of wrath (The living library)
Hannibal (Hannibal Lecter)
THE CATCHER IN THE RYE
Confessions of a Street Addict
Tripwire
Economics in one lesson (Pocket book)
Memoirs of a Geisha (Signed)
George Orwell 1984
On the Road
Hatchet
Of Mice & Men
Sophie's World
Angela's Ashes (Turtleback School & Library Binding Edition)
As a man thinketh (Little blue book)
Personal Memoirs of U.S. Grant (2 Volumes)
Left Behind : A Novel Of The Earth's Last Days
Teresa of Avila: The Interior Castle (The Classics of Western Spirituality)
PRAXIS/CST Music Education (National Teacher Examination Series)
How To Cook Everything: Simple Recipes for Great Food
Nineteen Eighty-four
The Body in the Attic
Running Blind
The Dead Room
Plantation: A Lowcountry tale
A Pirate Looks At Fifty (SIGNED)
A TOWN LIKE ALICE
Thirteen Moons: A Novel
Penguin Readers Level 6: Misery (Penguin Longman Reader Level 6)
Nightfall
Total Training for Young Champions
Wildfire
The Money Answer Book: Quick Answers to Everyday Financial Questions
Ringworld's Children
Open the Barn Door (A Chunky Book(R))
Sister Carrie, A novel
Pride and Prejudice
Longitudes and Attitudes: Exploring the World After September 11
Sgt. York: His Life, Legend & Legacy: The Remarkable Untold Story of Sgt. Alvin C. York
Quantum Touch: The Power to Heal
Red Tent
Why I Am Not a Christian: And Other Essays on Religion and Related Subjects
For the Love of Mike (Molly Murphy Mysteries)
Down Came the Rain: My Journey Through Postpartum Depression
Fox's Book of Martyrs
Jonathan Strange & Mr. Norrell Signed
The Silmarillion
The Devil Wears Prada
Title

3.4 Most number of Negative Reviews on the Books

[36]:
```
data[data['Sentiment'] == 'negative']['Title'].value_counts().head(50).
 ↪plot(kind = 'bar', figsize = (7,7))
plt.title('Number of Negative Reviews on the Books',fontsize = 15)
plt.xticks(rotation = 90)
plt.show()
```


Number of Negative Reviews on the Books
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Hannibal (Hannibal Lecter)
Hannibal
ERAGON: INHERITANCE, BOOK ONE
The Catcher in the Rye [Audiobook] [Cd] [Unabridged] (Audio CD)
To kill a mockingbird
Portrait of a Killer: Jack the Ripper - Case Closed
Holes
To Kill A Mockingbird
Lord of the Flies
Cold Mountain: A Novel
Black Notice
Night
The Chocolate War
The Innocent Man: Murder and Injustice in a Small Town (Random House Large Print)
With the Old Breed: At Peleliu and Okinawa
Slaughter-House Five (First paperback edition)
Confessions of an Economic Hitman
Lord of the flies: A novel
Out of the Dust
Slaughterhouse-Five
In Cold Blood: A True Account of a Multiple Murder and Its Consequences
OF MICE AND MEN
Tear Soup: A Recipe for Healing After Loss
Lord of the Flies
Anne Frank: The Diary of a Young Girl
Nineteen Eighty-four
To Kill a Mocking Bird
I know this much is true
Collapse: How Societies Choose to Fail or Succeed
Disgrace: Complete & Unabridged
George Orwell 1984
The Poisonwood Bible
The Giver
The Lord of the Rings (3 Volume Set)
WE WERE SOLDIERS ONCE . . . AND YOUNG : IA DRANG The Battle That Changed the War in Vietnam (LIBRARY OF MILITARY HISTORY)
Path of Daggers (Wheel of Time 08)
Crossroads of Twilight (The Wheel of Time, Book 10)
A Fine Balance
Slaughter-House Five
Uncle Tom's Cabin - Or, Life Among The Lowly
Johnny Got His Gun (A Bantam Windstone Book)
Red Rabbit
Brave new world: (Harper's modern classics)
Harry Potter and the Chamber of Secrets
The Devil Wears Prada
Angela's Ashes (Turtleback School & Library Binding Edition)
The Girl Who Loved Tom Gordon
Fellowship of the Ring 2ND Edition
Title

3.5 Dataframe of Reviews vs Sentiments

[37]:
```
reviews_df = data[['clean_reviews', 'Sentiment']]
```

[38]:
```
reviews_df.head()
```

[38]:

	clean_reviews	Sentiment
1123055	i have to say, this is one of the best suspens…	positive
1180512	it was an ok book. i read it last week, and i …	negative
28403	if you look at farenheit 451 from the perspect…	negative
2280455	this was my first exposure to dean koontz and …	positive
683668	i really liked the book. but my only problem i…	positive

3.6 Dividing Data into x & y

[39]:
```
x = reviews_df.drop(['Sentiment'], axis=1)
y = reviews_df['Sentiment']
```

[40]:
```
x.shape
```

[40]:
```
(15000, 1)
```

[41]:
```
y.shape
```

[41]:
```
(15000,)
```

4 text - preprocessing

[42]:
```
import numpy as np # basic Libraries
import pandas as pd
import seaborn as sns
import nltk
```

[43]:
```
!pip install -U nltk
```

```
Requirement already satisfied: nltk in /opt/conda/lib/python3.10/site-packages
(3.2.4)
Collecting nltk
  Obtaining dependency information for nltk from https://files.pythonhosted.org/
packages/a6/0a/0d20d2c0f16be91b9fa32a77b76c60f9baf6eba419e5ef5deca17af9c582/nltk
-3.8.1-py3-none-any.whl.metadata
  Downloading nltk-3.8.1-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: click in /opt/conda/lib/python3.10/site-packages
```

```
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /opt/conda/lib/python3.10/site-packages
(from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/opt/conda/lib/python3.10/site-packages (from nltk) (2023.8.8)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.10/site-packages
(from nltk) (4.66.1)
Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
                         1.5/1.5 MB
20.1 MB/s eta 0:00:0000:0100:01
Installing collected packages: nltk
  Attempting uninstall: nltk
    Found existing installation: nltk 3.2.4
    Uninstalling nltk-3.2.4:
      Successfully uninstalled nltk-3.2.4
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.
preprocessing 0.1.13 requires nltk==3.2.4, but you have nltk 3.8.1 which is
incompatible.
Successfully installed nltk-3.8.1
```

[44]:
```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /usr/share/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /usr/share/nltk_data…
```

[44]: True

[45]:
```
import re # regular expression module
stemmer = nltk.SnowballStemmer("english") # for stemming
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english')) # for stopword
```

[46]:
```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
import re, string, unicodedata
```

```
from string import punctuation
```

[47]:
```
def hapus_url(text):
    return re.sub(r'http\S+','', text)

def remove_special_characters(text, remove_digits=True):
    pattern=r'[^a-zA-Z0-9\s]'
    text=re.sub(pattern,'',text)
    return text

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

def final_clean(text):
    final_text= []
    for i in text.split():
        if i.strip().lower() not in stopword and i.strip().lower().isalpha():
            final_text.append(i.strip().lower())
    return " ".join(final_text)
```

[48]:
```
def clean(text):
    text = hapus_url(text)
    text = remove_special_characters(text, remove_digits=True)
    text = lemmi(text)
    text = final_clean(text)
    return text
```

[49]:
```
import nltk
import subprocess

# Download and unzip wordnet
try:
    nltk.data.find('wordnet.zip')
except:
    nltk.download('wordnet', download_dir='/kaggle/working/')
    command = "unzip /kaggle/working/corpora/wordnet.zip -d /kaggle/working/
 ↪corpora"
    subprocess.run(command.split())
    nltk.data.path.append('/kaggle/working/')

# Now you can import the NLTK resources as usual
from nltk.corpus import wordnet
```

```
[nltk_data] Downloading package wordnet to /kaggle/working/…
Archive:  /kaggle/working/corpora/wordnet.zip
```

```
   creating: /kaggle/working/corpora/wordnet/
  inflating: /kaggle/working/corpora/wordnet/lexnames
  inflating: /kaggle/working/corpora/wordnet/data.verb
  inflating: /kaggle/working/corpora/wordnet/index.adv
  inflating: /kaggle/working/corpora/wordnet/adv.exc
  inflating: /kaggle/working/corpora/wordnet/index.verb
  inflating: /kaggle/working/corpora/wordnet/cntlist.rev
  inflating: /kaggle/working/corpora/wordnet/data.adj
  inflating: /kaggle/working/corpora/wordnet/index.adj
  inflating: /kaggle/working/corpora/wordnet/LICENSE
  inflating: /kaggle/working/corpora/wordnet/citation.bib
  inflating: /kaggle/working/corpora/wordnet/noun.exc
  inflating: /kaggle/working/corpora/wordnet/verb.exc
  inflating: /kaggle/working/corpora/wordnet/README
  inflating: /kaggle/working/corpora/wordnet/index.sense
  inflating: /kaggle/working/corpora/wordnet/data.noun
  inflating: /kaggle/working/corpora/wordnet/data.adv
  inflating: /kaggle/working/corpora/wordnet/index.noun
  inflating: /kaggle/working/corpora/wordnet/adj.exc
```

[50]:
```
x["clean_reviews"] = x["clean_reviews"].apply(clean)
```

[51]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.
↪25,random_state=0)
```

[52]:
```
x_train
```

[52]:

	clean_reviews
2183966	read lot series far never disappointed book he…
797289	sorry book beautifully laid everything recipe …
430948	bought handy little pocket book giving one eve…
642689	detailed account alaska move slowly beautifull…
1614249	personal favorite harry potter bookswhy new ch…
…	…
2628863	copy book ha page possibly best page ive ever …
1024005	excellent book nearly paint scenery mind book …
2863514	rockwell another long line haunted personality…
2338317	lighthouse published landmark modernist novelw…
1884915	wa younger trusted history teacher textbook ga…

[11250 rows x 1 columns]

[53]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[54]:
```
tfidf_vectorizer = TfidfVectorizer()  # You can adjust the max_features␣
↪parameter
```

```
x_train_tfidf = tfidf_vectorizer.fit_transform(x_train['clean_reviews'])

# Transform the 'clean_reviews' column on the testing data
x_test_tfidf = tfidf_vectorizer.transform(x_test['clean_reviews'])
```

[55]:
```
x_train_tfidf_df = pd.DataFrame(x_train_tfidf.toarray())
x_test_tfidf_df =  pd.DataFrame(x_test_tfidf.toarray())
```

4.1 Balancing Data

4.1.1 Using Smote for Balancing Data

[56]:
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
x_train_bal, y_train_bal = oversample.fit_resample(x_train_tfidf, y_train)
y_train_bal.value_counts()
```

[56]:
```
Sentiment
positive    9293
negative    9293
neutral     9293
Name: count, dtype: int64
```

[57]:
```
x_test_bal, y_test_bal = oversample.fit_resample(x_test_tfidf, y_test)
y_test_bal.value_counts()
```

[57]:
```
Sentiment
positive    3123
negative    3123
neutral     3123
Name: count, dtype: int64
```

4.2 Model Training

4.3 1) Logistic Regression

4.3.1 Applying Logistic Regression on Imbalanced Data

[58]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

[59]:
```
log = LogisticRegression()
log.fit(x_train_tfidf_df, y_train)
```

[59]:
```
LogisticRegression()
```

[60]:
```
perd_log=log.predict(x_test_tfidf_df)
```

[61]:
```python
print(classification_report(perd_log, y_test))
```

```
              precision    recall  f1-score   support

    negative       0.24      0.85      0.38       158
     neutral       0.00      0.00      0.00         0
    positive       0.99      0.86      0.92      3592

    accuracy                           0.86      3750
   macro avg       0.41      0.57      0.43      3750
weighted avg       0.96      0.86      0.90      3750

/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

4.3.2 Applying Logistic Regression on Balanced Data

[62]:
```python
log_bal = LogisticRegression()
log_bal.fit(x_train_bal, y_train_bal)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458:
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
  n_iter_i = _check_optimize_result(
```

[62]:
```
LogisticRegression()
```

[63]:
```python
perd_log_bal =log_bal.predict(x_test_bal)
```

[64]:
```
print(classification_report(perd_log_bal, y_test_bal))
```

```
              precision    recall  f1-score   support

    negative       0.76      0.63      0.69      3722
     neutral       0.11      0.89      0.19       368
    positive       0.92      0.54      0.68      5279

    accuracy                           0.59      9369
   macro avg       0.59      0.69      0.52      9369
weighted avg       0.82      0.59      0.66      9369
```

[65]:
```
from sklearn.metrics import confusion_matrix
```

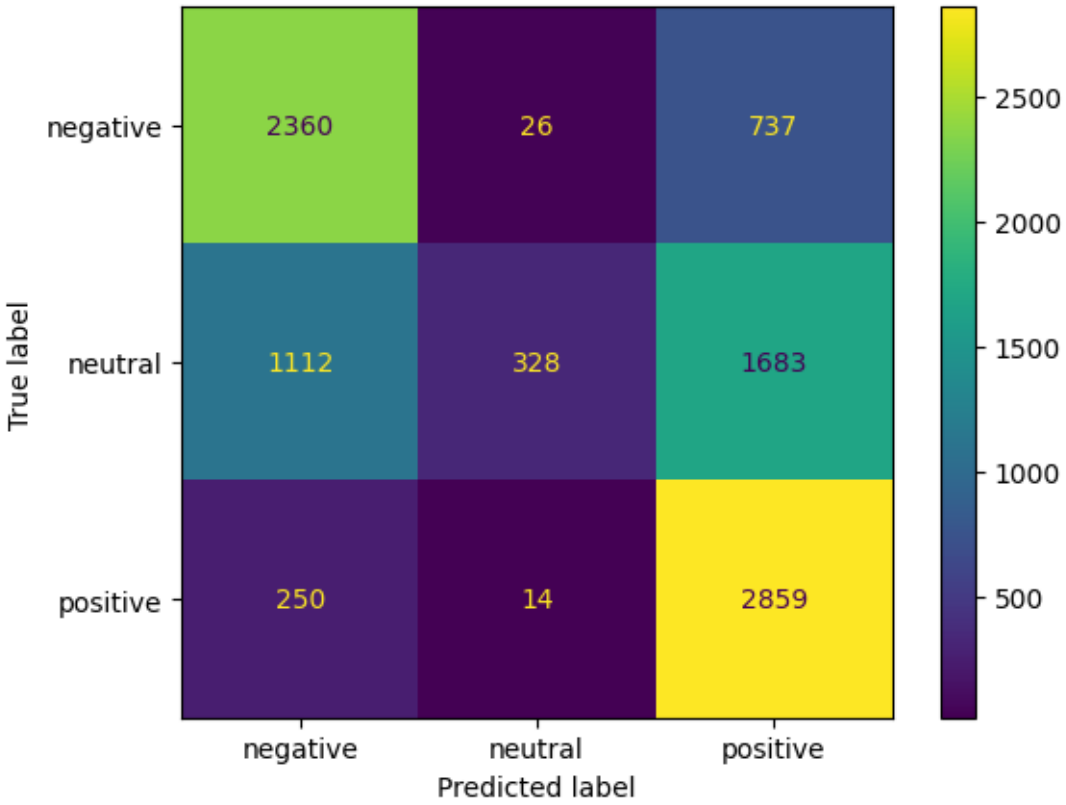
[66]:
```
confusion_matrix(y_test_bal,perd_log_bal)
```

[66]:
```
array([[2360,   26,  737],
       [1112,  328, 1683],
       [ 250,   14, 2859]])
```

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

[68]:
```
cm = confusion_matrix(y_test_bal, perd_log_bal, labels=log.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=log.classes_)
disp.plot()
```

[68]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x78641a142c50>
```



4.4 2) Decision Tree

4.4.1 Decision tree on imbalanced Data

[69]:
```
from sklearn.tree import DecisionTreeClassifier
```

[70]:
```
classifier= DecisionTreeClassifier(criterion='entropy', random_state=45)
classifier.fit(x_train_tfidf_df, y_train)
```

[70]:
```
DecisionTreeClassifier(criterion='entropy', random_state=45)
```

[71]:
```
pred_tre = classifier.predict(x_test_tfidf_df)
```

[72]:
```
print(classification_report(pred_tre, y_test))
```

```
              precision    recall  f1-score   support

    negative       0.36      0.35      0.35       559
     neutral       0.15      0.17      0.16        71
    positive       0.87      0.87      0.87      3120

    accuracy                           0.78      3750
```

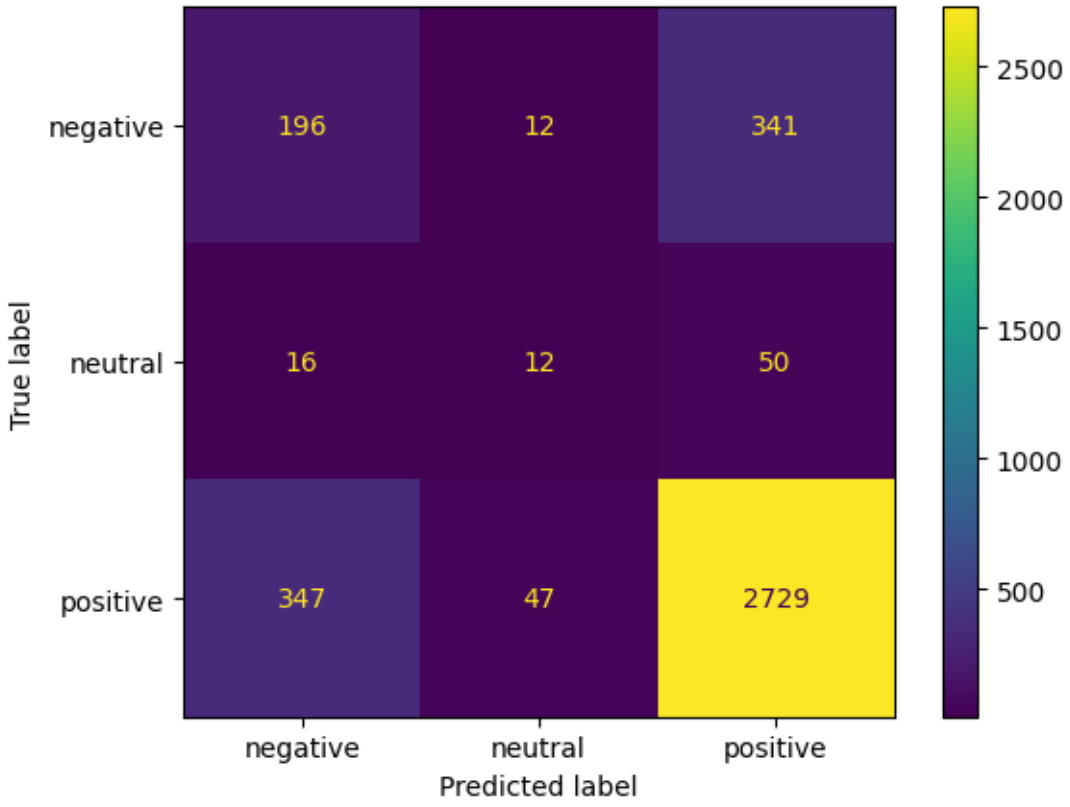
```
   macro avg       0.46      0.46      0.46      3750
weighted avg       0.78      0.78      0.78      3750
```

[73]:
```
confusion_matrix(y_test,pred_tre)
```

[73]:
```
array([[ 196,   12,  341],
       [  16,   12,   50],
       [ 347,   47, 2729]])
```

[74]:
```
cm = confusion_matrix(y_test, pred_tre, labels=classifier.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classifier.
 ↪classes_)
disp.plot()
```

[74]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x786418769bd0>
```



4.4.2 Decision tree on balanced data

[75]:
```
classifier_bal= DecisionTreeClassifier(criterion='entropy', random_state=45)
classifier_bal.fit(x_train_bal, y_train_bal)
```

[75]:
```
DecisionTreeClassifier(criterion='entropy', random_state=45)
```

[76]:
```
pred_tre_bal = classifier_bal.predict(x_test_bal)
```

[77]:
```
print(classification_report(pred_tre_bal, y_test_bal))
```

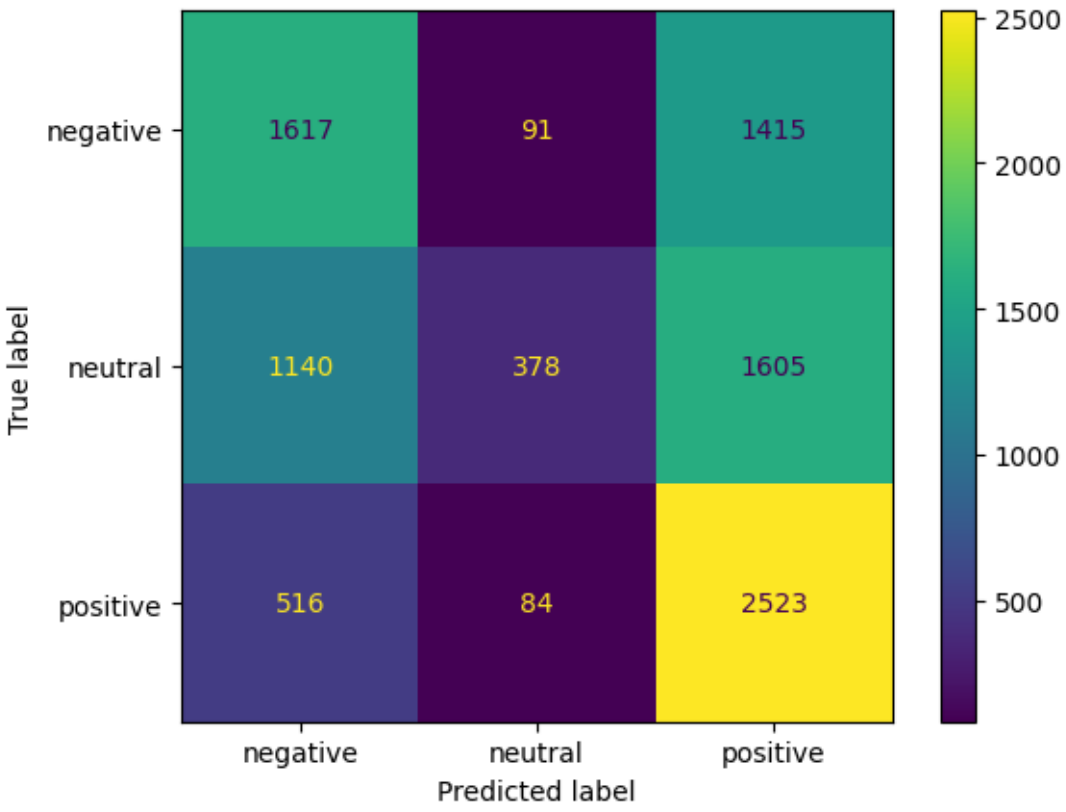
```
              precision    recall  f1-score   support

    negative       0.52      0.49      0.51      3273
     neutral       0.12      0.68      0.21       553
    positive       0.81      0.46      0.58      5543

    accuracy                           0.48      9369
   macro avg       0.48      0.54      0.43      9369
weighted avg       0.67      0.48      0.53      9369
```

[78]:
```
confusion_matrix(y_test_bal,pred_tre_bal)
```

[78]:
```
array([[1617,   91, 1415],
       [1140,  378, 1605],
       [ 516,   84, 2523]])
```

[79]:
```
cm = confusion_matrix(y_test_bal, pred_tre_bal, labels=classifier.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classifier.
 ↪classes_)
disp.plot()
```

[79]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x786418768f40>
```



4.5 3) Naive Bayes

4.5.1 Naive bayes on imbalanced Data

[80]:
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
```

[81]:
```
nb.fit(x_train_tfidf_df, y_train)
```

[81]:
```
GaussianNB()
```

[82]:
```
pred_nb = nb.predict(x_test_tfidf_df)
```

[83]:
```
print(classification_report(pred_nb, y_test))
```

```
              precision    recall  f1-score   support

    negative       0.29      0.14      0.18      1160
     neutral       0.06      0.10      0.08        49
    positive       0.68      0.84      0.75      2541
```

```
    accuracy                           0.61      3750
   macro avg       0.34      0.36      0.34      3750
weighted avg       0.55      0.61      0.57      3750
```

4.5.2 Naive Bayes on Balanced Data

[84]:
```
import numpy as np

# Assuming x_train_bal is your sparse matrix
x_train_bal_dense = x_train_bal.toarray()

# Then proceed with fitting the model using the dense array
nb_bal = GaussianNB()
nb_bal.fit(x_train_bal_dense, y_train_bal)
```

[84]:
```
GaussianNB()
```

[85]:
```
pred_nb_bal = nb_bal.predict(x_test_bal.toarray())
```

[86]:
```
print(classification_report(pred_nb_bal, y_test_bal))
```

```
              precision    recall  f1-score   support

    negative       0.19      0.19      0.19      3130
     neutral       0.01      0.48      0.03        91
    positive       0.68      0.35      0.46      6148

    accuracy                           0.30      9369
   macro avg       0.30      0.34      0.23      9369
weighted avg       0.51      0.30      0.37      9369
```

4.6 4) RandomForest

4.6.1 RandomForest on Imbalanced Data

[87]:
```
from sklearn.ensemble import RandomForestClassifier
```

[88]:
```
x_train_tfidf_df.shape
```

[88]:
```
(11250, 62831)
```

[89]:
```
x_test_tfidf_df.shape
```

[89]:
```
(3750, 62831)
```

[90]:
```
ran=RandomForestClassifier(n_estimators=100)
ran.fit(x_train_tfidf_df,y_train)
```

[90]:
```
RandomForestClassifier()
```

[91]:
```
perd_ran=ran.predict(x_test_tfidf_df)
```

[92]:
```
perd_ran.shape
```

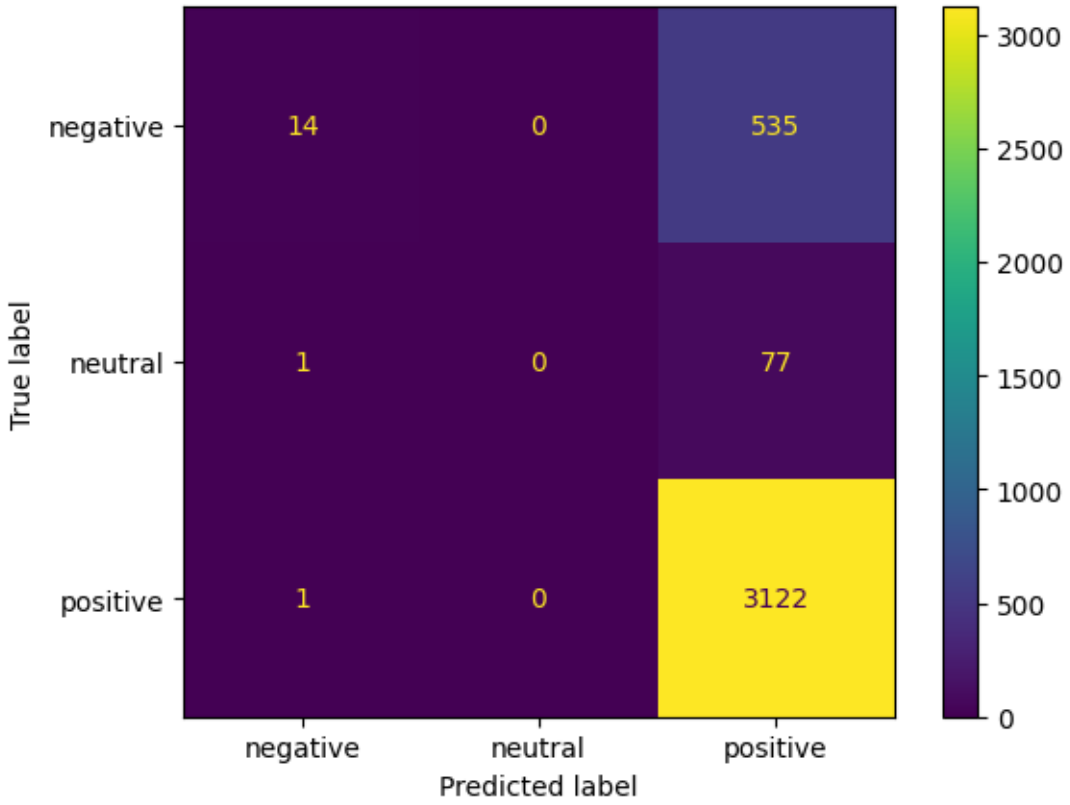
[92]:
```
(3750,)
```

[93]:
```
confusion_matrix(y_test,perd_ran)
```

[93]:
```
array([[  14,    0,  535],
       [   1,    0,   77],
       [   1,    0, 3122]])
```

[94]:
```
cm = confusion_matrix(y_test, perd_ran, labels=ran.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=ran.classes_)
disp.plot()
```

[94]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x786418604e50>
```



[95]:
```
y_test.shape , perd_ran.shape
```

[95]:
```
((3750,), (3750,))
```

[96]:
```
print(classification_report(perd_ran, y_test))
```

```
              precision    recall  f1-score   support

    negative       0.03      0.88      0.05        16
     neutral       0.00      0.00      0.00         0
    positive       1.00      0.84      0.91      3734

    accuracy                           0.84      3750
   macro avg       0.34      0.57      0.32      3750
weighted avg       1.00      0.84      0.91      3750

/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

4.6.2 RandomForest on balanced Data

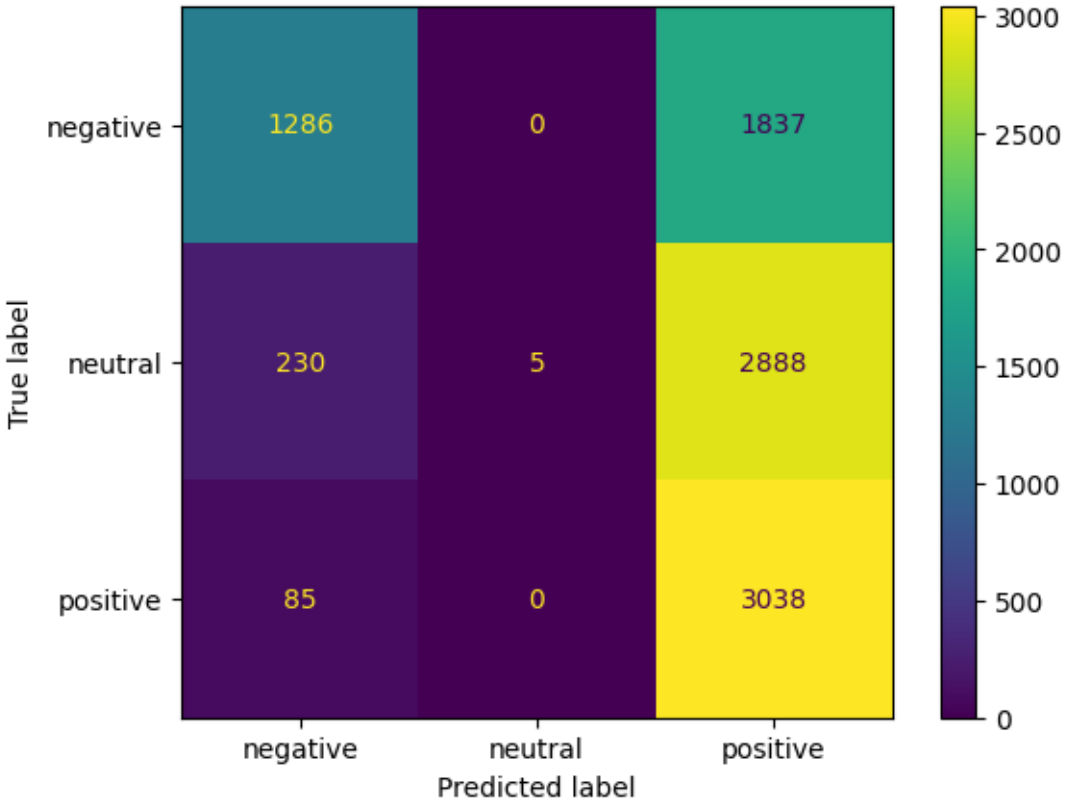
[97]:
```
ran_bal=RandomForestClassifier(n_estimators=100)
ran_bal.fit(x_train_bal,y_train_bal)
perd_ran_bal=ran_bal.predict(x_test_bal)
```

[98]:
```
confusion_matrix(y_test_bal,perd_ran_bal)
```

[98]:
```
array([[1286,    0, 1837],
       [ 230,    5, 2888],
       [  85,    0, 3038]])
```

[99]:
```
cm = confusion_matrix(y_test_bal, perd_ran_bal, labels=ran.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=ran.classes_)
disp.plot()
```

[99]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7864186054e0>
```



[100]:
```
print(classification_report(perd_ran_bal, y_test_bal))
```

```
              precision    recall  f1-score   support

    negative       0.41      0.80      0.54      1601
     neutral       0.00      1.00      0.00         5
    positive       0.97      0.39      0.56      7763

    accuracy                           0.46      9369
   macro avg       0.46      0.73      0.37      9369
weighted avg       0.88      0.46      0.56      9369
```

4.6.3 Saving the best model

[101]:
```
import pickle
```

[102]:
```
pickle.dump(log_bal, open('log_bal.pkl','wb'))
```

[103]:
```
with open('log_bal','wb') as f:
  pickle.dump(log_bal,f)
```

[104]:
```
with open('log_bal','rb') as f:
  mp=pickle.load(f)
```

[105]:
```
def predict(text):
  input_text = [text]
  text_to_vec = tfidf_vectorizer.transform(input_text).toarray()
  prediction = mp.predict(text_to_vec)
  return prediction
```

[106]:
```
User_input = input("Enter your text:")
output = predict(User_input)
print(output)
```

```
Enter your text: Hello i like the book
```

```
['positive']
```

[110]:
```
User_input = input("Enter your text:")
output = predict(User_input)
print(output)
```

```
Enter your text: Bad book
```

```
['negative']
```

[111]:
```
User_input = input("Enter your text:")
output = predict(User_input)
print(output)
```

```
Enter your text: The book was neither good nor bad. It was just okay.
```

```
['negative']
```