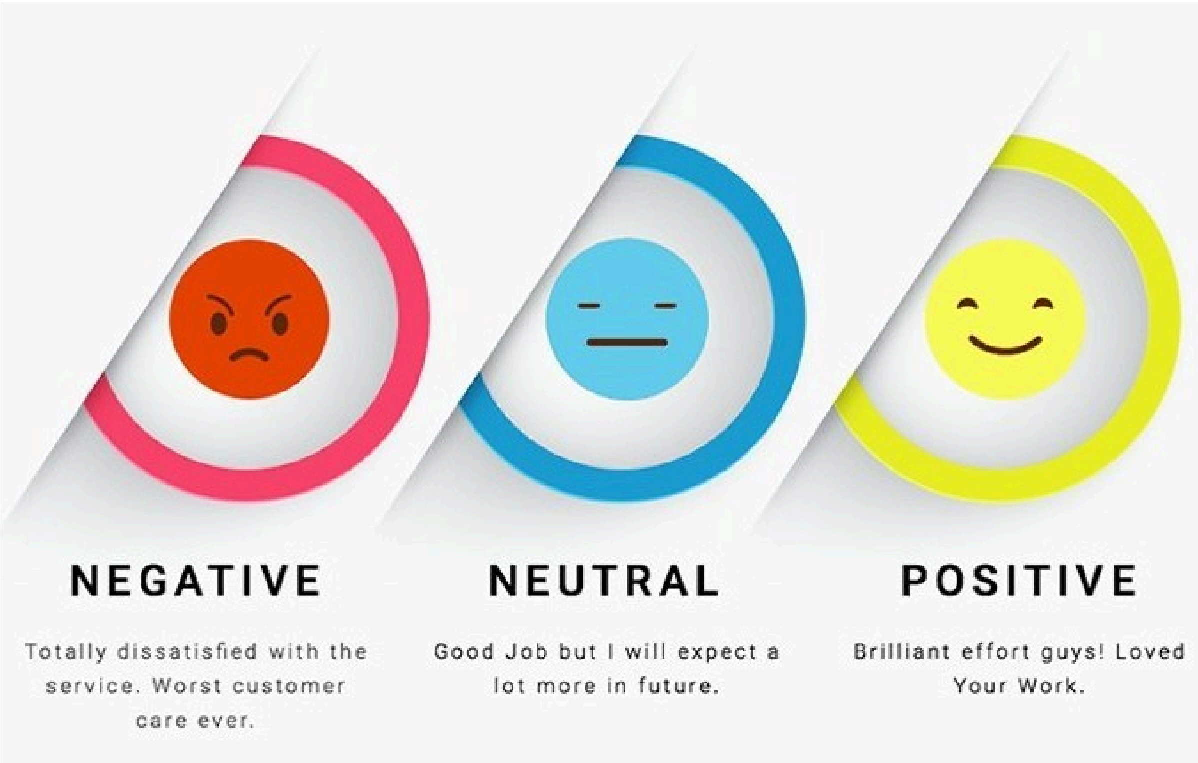
INTERNATIONAL SCHOOL OF AI AND DATA SCIENCE

Customer Sentiment Analysis

Table of Contents

1. Introduction

- **Sentiment is the emotion behind customer engagement.**
- When you monitor sentiment, you try to measure the **tone**, **context**, and **feeling** from customer actions.
- Whether a customer completes a purchase, leaves a review, or mentions your company socially, there is always an **emotional state** connected to their action.
- Customer sentiment can range anywhere from **pleased** or **loving** to **neutral** or **angry**, and no matter where your customers fall on the sentiment spectrum, it's imperative you understand not only what their emotional state is, but what's driving it.



2. Problem Statement

- **Analyzing customer sentiment** helps give **insight** into how **customers feel** about your brand.
- The more you listen to how your customers feel about recommending your company, giving you a rating, engaging with you on social channels, and giving you direct feedback, the more love everyone is sure to feel, and the deeper your relationships can be.

Business Scenario:

- A large electronics company **Green Electric** has been falling behind the competition in terms of providing a good customer service to their customers.
- Their previous marketing campaigns have also been hit or a miss, and they don't know for certain what their customers want.
- They want to **gain deeper audience insight**, **improve** their **customer engagement**, **provide improved customer service** to their customers and also **improve** the **success rate** of their future **marketing campaigns**.
- To achieve this, the management proposed **analyzing** the **sentiment** of different customers for different products.
- But, analyzing customer sentiment can be a **hectic** process if done **manually**, due to the sheer volume of data. So the company wants to **automate** the process of **Sentiment Analysis**.
- They have assigned their **Data Science team**, the task to automate the Sentiment Analysis of **future reviews**.

3. Importing Libraries

3.1 Installing Libraries

Note: After installing, you need to restart the runtime. Make sure not to execute the cell again after restarting the runtime.

```
!pip install -q pandas-profiling --upgrade
```

3.2 Importing Libraries

```
# For Numerical Python
import numpy as np

# For Panel Data Analysis
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)
pd.set_option('mode.chained_assignment', None)
pd.set_option('display.precision', 4)

#from pandas_profiling import ProfileReport

# For Data Visualization
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

# To Disable Warnings
import warnings
warnings.filterwarnings("ignore")


import re
import nltk


nltk.download('all')


from wordcloud import WordCloud


from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer


from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_repo
```

4. Data Loading and Dataset Description

- We are provided with a **customer review** data of different **Electronic products** sold on an E-commerce platform.
- This massive dataset of reviews will help us **build** a **Sentiment Analysis model** capable of classifying future reviews into their respective sentiment.
- The dataset contains information about the **marketplace**, **customers**, **products**, and also contains the review information including the entire **review text** written by the customer.
- Also provided in the dataset is the `star_rating`. It is the **1-5 star rating** of the review.

Records	Features	Dataset Size
30,93,869	15	1.61 GB

Column	Description
marketplace	2 letter country code of the marketplace where the review was written.
customer_id	Random identifier that can be used to aggregate reviews written by a single author.
review_id	The unique Product ID the review pertains to.
product_id	Sales for the given department in the given store.
product_parent	Random identifier that can be used to aggregate reviews for the same product.
product_title	Title of the product.
product_category	Broad product category that can be used to group reviews (also used to group the dataset into coherent par
star_rating	The 1-5 star rating of the review.
helpful_votes	Number of helpful votes.
total_votes	Number of total votes the review received.
vine	Review was written as part of the Vine program.
verified_purchase	The review is on a verified purchase.
review_headline	The title of the review.
review_body	The review text.
review_date	The date the review was written.

4.1 Data Loading

```
reviews_df = pd.read_csv('https://storage.googleapis.com/retail-analytics-data/re
reviews_df.head(3)
```

	marketplace	customer_id	review_id	product_id	product_parent	pro
0	US	41409413	R2MTG1GCZLR2DK	B00428R89M	112201306	RP Ext
1	US	49668221	R2HBOEM8LE9928	B000068O48	734576678	H 3 1/4"
2	US	12338275	R1P4RW1R9FDPEE	B000GGKOG8	614448099	Cl Ti

4.2 Dataset Description

```
reviews_df.describe()
```

	customer_id	product_parent	star_rating	helpful_votes	total_votes
count	3.0939e+06	3.0939e+06	3.0939e+06	3.0939e+06	3.0939e+06
mean	2.8789e+07	5.1020e+08	4.0355e+00	1.8598e+00	2.3711e+00
std	1.5431e+07	2.8683e+08	1.3874e+00	2.1328e+01	2.2487e+01
min	1.0005e+04	6.4780e+03	1.0000e+00	0.0000e+00	0.0000e+00
25%	1.5037e+07	2.6236e+08	3.0000e+00	0.0000e+00	0.0000e+00
50%	2.8063e+07	5.0855e+08	5.0000e+00	0.0000e+00	0.0000e+00
75%	4.3279e+07	7.6324e+08	5.0000e+00	1.0000e+00	1.0000e+00
max	5.3097e+07	1.0000e+09	5.0000e+00	1.2786e+04	1.2944e+04

Observations:

- Most of the columns in the dataset contain **textual data** and hence are not shown in the `describe` function output.
- **star_rating** column has a **mean** of **4** and a **median** of **5**.
  - This implies that it is **negative** (**left**) skewed.

4.3 Pandas Profiling before Data Preprocessing

- Using **pandas-profiling** to quickly *analyse* our data.
- This will take some time.

```
profile = ProfileReport(reviews_df, progress_bar=False, minimal=True)
```

```
profile.to_file(output_file="Pre_Profiling_Report.html")
print('Pre-Profiling Accomplished!')
```

Observations:

- There are **15 variables** and **3093869 observations** in the dataset.
- There are **123 missing** cells (**less than 0.1%** of all cells) in the data.
- Of all the 16 variables **8** are **categorical**, **5** are **numerical** and **2** are **boolean**.
- `review_id` column has a high cardinality with **3093869** distinct values.
- `customer_id` column has a high cardinality with **2154357** distinct values.
- `product_id` column has a high cardinality with **185852** distinct values.
- `product_title` column has a high cardinality with **167933** distinct values.
- `product_parent` column has a high cardinality with **166244** distinct values.
- `review_headline` column has a high cardinality with **1637220** distinct values.
- `review_body` column has a high cardinality with **2897248** distinct values.
    - This column will provide us the **data** required to **build** our Sentiment Analysis Mode.
- `review_date` column has a high cardinality with **5904** distinct values.
- `marketplace` column contains only a **single** value throughout the dataset i. e. **US**
- `product_category` column contains only a **single** value throughout the dataset i. e. **Electronics**
- `vine` and `verified_purchase` columns both have two distinct values: **Y**, and **N**
- `star_rating` is the target variable.
    - It contains **5** different **rating values**.
    - They are **1**, **2**, **3**, **4**, and **5**.
    - **5** is the **most common** value in the `star_rating` column.

5. Data Pre-Processing

5.1 Assign String DataType to Review Body

- Currently, the `review_body` column doesn't have a uniform datatype for all the observations in the data.
- As a result, it will give **errors** when subjected to string operations.
- So we will assign a **string** datatype to the entire column to prevent such errors.

```
reviews_df['review_body'] = reviews_df['review_body'].astype(str)
```

- Now, the `review_body` column will have a **string** datatype for each observation in the data.

5.2 Cleaning the Reviews

- Here, we will **clean** the review data by:
  - **Changing** the **case** of each word to **lowercase**.
  - **Fixing** certain words like *i'm to i am*, *he's to he is*, *she's to she is*, etc.
  - **Removing** all the **punctuation marks** from each review.
  - **Removing** any additional white space from each review.
- Then, we will **create** a new column in the dataset `clean_reviews`, that will contain all the cleaned reviews.

- Creating a function `clean_text` that will help us in **cleaning** the **reviews** and **saving** them.

```
# Creating a helper function to clean the text.
def clean_text(text):

    text = text.lower().strip()
    text = re.sub(r"i'm", "i am", text)
    text = re.sub(r"he's", "he is", text)
    text = re.sub(r"she's", "she is", text)
    text = re.sub(r"that's", "that is", text)
    text = re.sub(r"what's", "what is", text)
    text = re.sub(r"where's", "where is", text)
    text = re.sub(r"how's", "how is", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"can't", "cannot", text)
    text = re.sub(r"n't", " not", text)
    text = re.sub(r"<br>", " ", text)

    #Complex Expression
    text = re.sub(r"([-?.!,/\"])", r" \1 ", text)
    text = re.sub(r"[-()\"#/@;:<>{}`+=~|.!?,']", "", text)
    text = re.sub(r"[ ]+", " ", text)
    text = text.rstrip().strip()

    return text
```

Complicated Regex

- This regex will **add** a **space before** and **after** any of **-?.!,/"**
- Here, if any of these characters is found in the text, it will become a capture group.
    - Anything in **()** is a **capture group** in **r"([-?.!,/"])"**.
    - It is then denoted by **\1**, and is **replaced** by that character with a space before and after it.
- This will help us **separate words** from these **characters**, and as a result will provide a cleaner text.

```
re.sub(r"([-!?.,/\"])", r" \1 ", 'Hi!')
```

```
'Hi ! '
```

- Here, we can see a **space** is **added** between **Hi** and **!**

- This regex will **remove** any of these **characters -()"#/@;:<>{}`+=~|.!?,'** from the text.

```
re.sub(r"[-()\"#/@;:<>{}`+=~|.!?,']", "", 'Hi ! ')
```

```
'Hi '
```

- **!** is **removed** from **Hi !**

- Using this regex, **more than one space** is **replaced** by a **single space**.
- Here, **+** denotes **more than one**.
- If there is more than a single space in the text at once, only then it will be replaced by a single space.

```
re.sub(r"[ ]+", " ", 'Hi   how are you')
```

```
'Hi how are you'
```

- **3 spaces** in the above text is **replaced** by a **single space**.

Creating Clean Reviews

- Creating a **corpus** of cleaned reviews using the `clean_text` function on **review_body** column of `reviews_df`.
- This will take some time.

```
reviews_df['clean_reviews'] = reviews_df['review_body'].apply(lambda x:clean_text
reviews_df.head()
```

	marketplace	customer_id	review_id	product_id	product_parent	pro
0	US	41409413	R2MTG1GCZLR2DK	B00428R89M	112201306	RP Ext
1	US	49668221	R2HBOEM8LE9928	B000068O48	734576678	H 3 1/4"
2	US	12338275	R1P4RW1R9FDPEE	B000GGKOG8	614448099	Ch Ti
3	US	38487968	R1EBPM82ENI67M	B000NU4OTA	72265257	L c C C Ge
4	US	23732619	R372S58V6D11AT	B00JOQIO6S	308169188	

- We can see `clean_reviews` column contain all the **cleaned reviews** in the dataset.

5.3 Calculating Polarity and Subjectivity of Reviews

- **Polarity** is a float value within the range **[-1.0 to 1.0]**.
  - Here, **0** indicates **neutral**,
  - **+1** indicates a **very positive** sentiment, and
  - **-1** represents a **very negative** sentiment.
- **Subjectivity** is a float value within the range **[0.0 to 1.0]**.
  - Here, **0.0** is **very objective**, and

- **1.0** is **very subjective**.
- **Subjective** sentence **express** some *personal feelings, views, beliefs, opinions, allegations, desires, beliefs, suspicions, and speculations*.
- **Objective** sentences are **factual**.

- We will use `textblob` library's **TextBlob** class to find the **polarity** and **subjectivity** values for each review.

```
for review in reviews_df['clean_reviews'][10:15]:
    blob = TextBlob(review)
    print(review, ': ', blob.sentiment, '\n')
```

```
alll good :  Sentiment(polarity=0.7, subjectivity=0.6000000000000001)

love clock radio & cd player easy to operate :  Sentiment(polarity=0.46666666

breaks very easily and takes a while to load music :  Sentiment(polarity=0.56

excellent gain in radio frequency reception over the stock antenna that came 

everything i expected for a great price :  Sentiment(polarity=0.350000000000000
```

- Here, we can see the *polarity* and *subjectivity* values for the **first 5 reviews** in the dataset.
- **First 4** reviews have a **neutral sentiment** according to the polarity values equal to **0**.
- **Fifth** review has a slightly **positive sentiment** with a polarity of **0.0875**

Calculating the Polarity and Subjectivity of Cleaned Reviews

- Using TextBlob to **calculate** the **polarity** and **subjectivity** values for each cleaned review.
- These values are then **appended** to `polarity` and `subjectivity` lists.
- This will take some time.

```
%%time
polarity = []
subjectivity = []

for review in reviews_df['clean_reviews']:
    blob = TextBlob(review)
    polarity.append(blob.sentiment.polarity)
    subjectivity.append(blob.sentiment.subjectivity)
```

```
CPU times: user 38min 52s, sys: 9.81 s, total: 39min 2s
Wall time: 39min 3s
```

- **Adding** the polarity and subjectivity values to `polarity` and `subjectivity` columns in the dataset.

```
reviews_df['polarity'] = polarity
reviews_df['subjectivity'] = subjectivity
```

```
reviews_df.head(3)
```

	marketplace	customer_id	review_id	product_id	product_parent	pr
0	US	41409413	R2MTG1GCZLR2DK	B00428R89M	112201306	RI Ex
1	US	49668221	R2HBOEM8LE9928	B000068O48	734576678	H 1/4
2	US	12338275	R1P4RW1R9FDPEE	B000GGKOG8	614448099	C T
3	US	38487968	R1EBPM82ENI67M	B000NU4OTA	72265257	(G
4	US	23732619	R372S58V6D11AT	B00JOQIO6S	308169188	
5	US	21257820	R1A4514XOYI1PD	B008NCD2LG	976385982	BS J
6	US	3084991	R20D9EHB7N20V6	B00007FGUF	670878953	Gc
7	US	8153674	R1WUTD8MVSROJU	B00M9V2RMM	508452933	CC

- We can see that the **polarity** and **subjectivity** values for each cleaned review in the `polarity` and `subjectivity` columns respectively.

9 US 41463864 R904DQPBCEM7A B00NSTA0E4 458130381 (

6. Exploratory Data Analysis

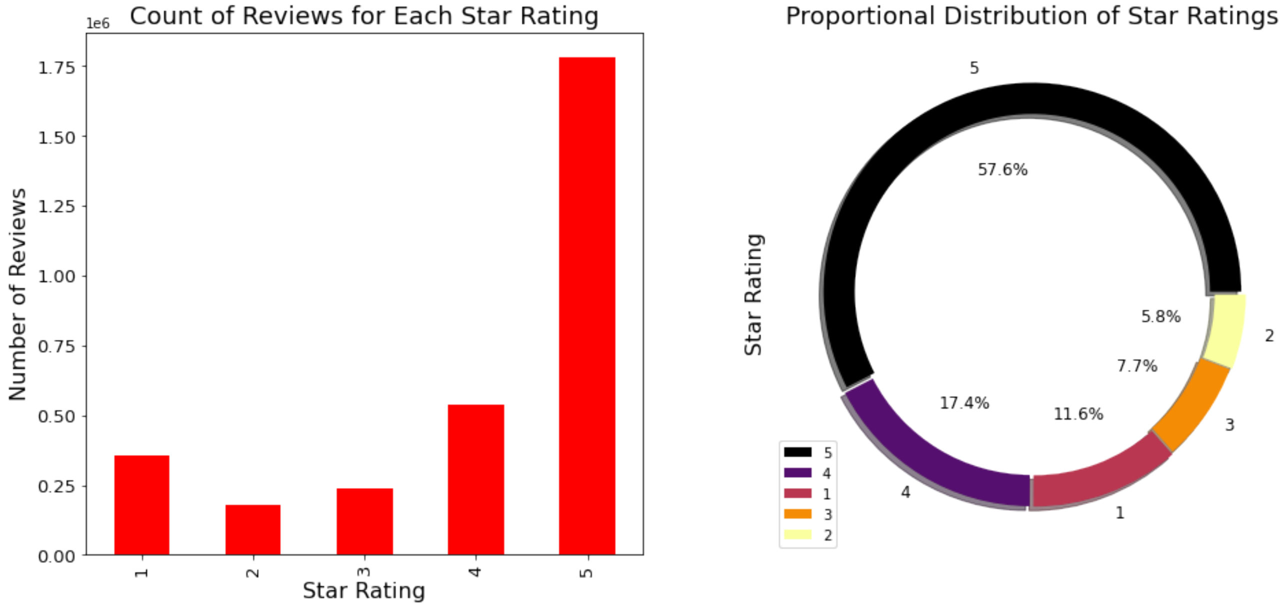
Question 1: How are the Star Ratings distributed for the Reviews?

```
reviews_df['star_rating'].value_counts()
```

```
5    1781161
4     536821
1     358120
3     238587
2     179180
Name: star_rating, dtype: int64
```

```
# Plotting the Count and Proportional Distribution of reviews based on star ratin
plt.figure(figsize=(16, 7))
plt.subplot(1, 2, 1)

# Plotting the count of reviews for each star rating
reviews_df['star_rating'].value_counts().sort_index().plot(kind='bar', color='Red
plt.xlabel('Star Rating', fontsize=16)
plt.ylabel('Number of Reviews', fontsize=16)
plt.title('Count of Reviews for Each Star Rating', fontsize=18)

plt.subplot(1, 2, 2)

# Plotting the proportional distribution of star ratings
reviews_df['star_rating'].value_counts().plot(kind='pie', autopct='%1.1f%%', exp
                                               fontsize=12, wedgeprops=dict(width=
                                               shadow=True, startangle=0, cmap='in
plt.ylabel('Star Rating', fontsize=16)
plt.title('Proportional Distribution of Star Ratings', fontsize=18)
```

```
Text(0.5, 1.0, 'Proportional Distribution of Star Ratings')
```



Observations:

- We can observe that most of the reviews have a **5 star rating** with a **57.6%** share of all the reviews.
- It is followed by 4 stars, then 1 star, 3 stars, and then 2 stars.
- **2 star rating** has the **least share** of the reviews with just **5.8%** reviews having this rating.
- This tells us that most of the people **usually** *leave a 5 star review* for the product they have bought and **less frequently** will *leave a low star review*.

Question 2: How are the Reviews distributed into Sentiments based on Polarity?

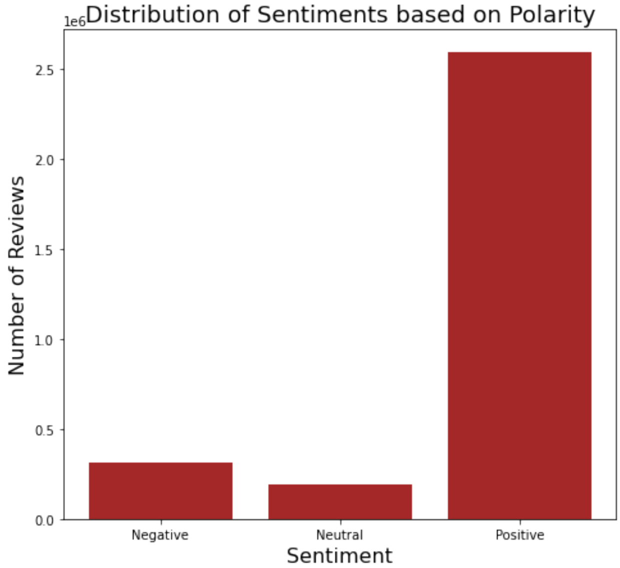
- Reviews having:
  - a **negative polarity** will have a **negative sentiment**,
  - **zero** polarity will have a **neutral sentiment**, and
  - **positive** polarity will have a **positive sentiment**.

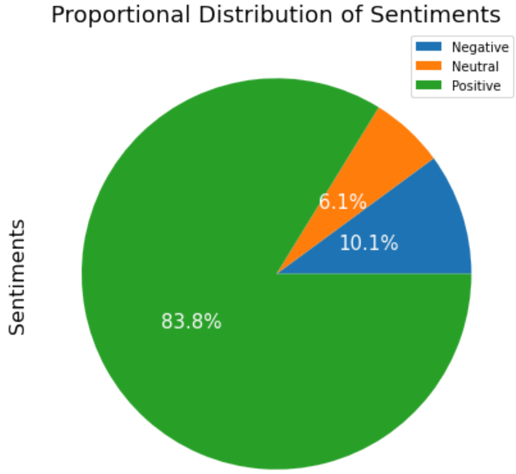
```
print('Reviews with Negative Sentiment based on Polarity:', len(reviews_df[review
print('Reviews with Neutral Sentiment based on Polarity:', len(reviews_df[reviews
print('Reviews with Positive Sentiment based on Polarity:', len(reviews_df[review
```

```
Reviews with Negative Sentiment based on Polarity: 311344
Reviews with Neutral Sentiment based on Polarity: 190167
Reviews with Positive Sentiment based on Polarity: 2592358
```

```
# Plotting the Count and Proportional Distribution of reviews based on sentiment
plt.figure(figsize=(17, 7))
plt.subplot(1, 2, 1)

# Plotting the count of reviews for each sentiment
plt.bar(['Negative', 'Neutral', 'Positive'], [len(reviews_df[reviews_df['polarity
                                              len(reviews_df[reviews_df['polarity
plt.xlabel('Sentiment', fontsize=16)
plt.ylabel('Number of Reviews', fontsize=16)
plt.title('Distribution of Sentiments based on Polarity', fontsize=18)

plt.subplot(1, 2, 2)

# Plotting the proportional distribution of sentiments
plt.pie(x=[len(reviews_df[reviews_df['polarity'] < 0]), len(reviews_df[reviews_df
           len(reviews_df[reviews_df['polarity'] > 0])],
        labels=['Negative', 'Neutral', 'Positive'], autopct='%1.1f%%', pctdistanc
        textprops={'fontsize':15, 'color':'white'})
plt.ylabel('Sentiments', fontsize=16)
plt.title('Proportional Distribution of Sentiments', fontsize=18)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fb14835ac18>
```





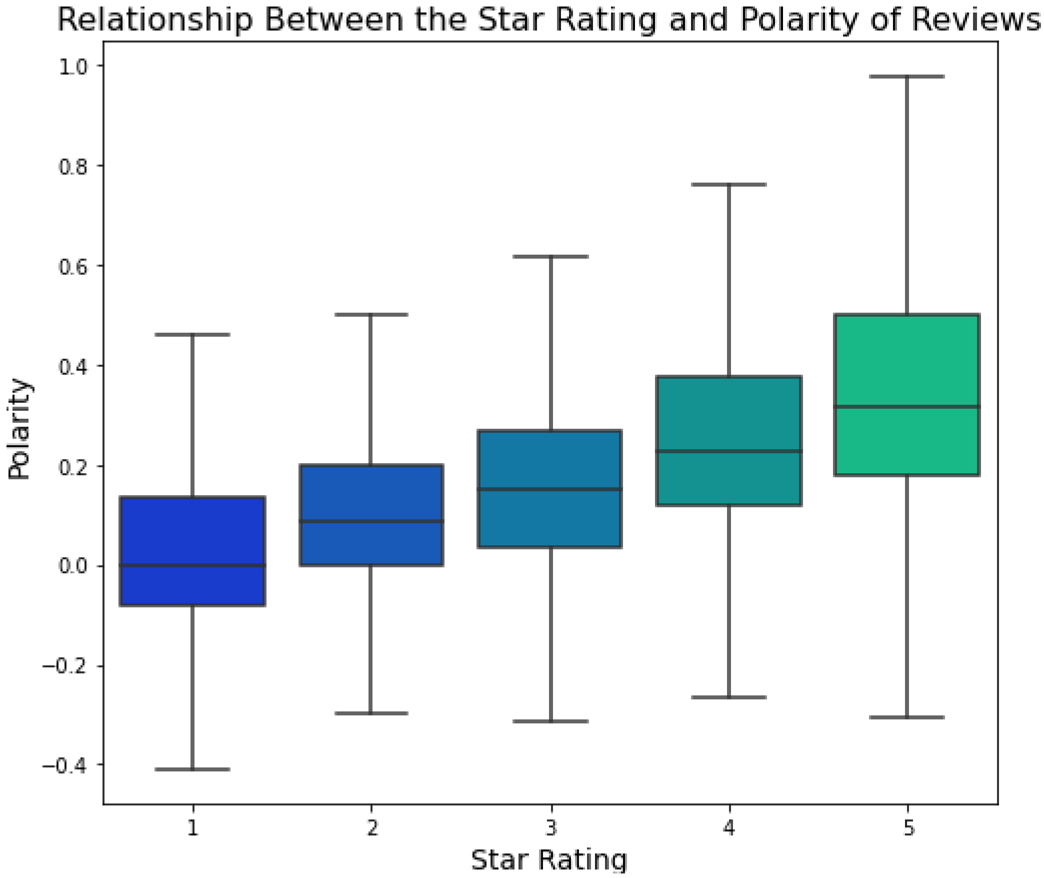
Observations:

- Based on the polarity values, **83.8% reviews** have a **positive sentiment**.
- Only **10.1%** reviews have a **negative** sentiment and **6.1%** reviews have a **neutral** sentiment.

Question 3: What is the Relationship between the Star Ratings and the Polarity of the Reviews?

```
plt.figure(figsize=(8, 7))
sns.boxplot(data=reviews_df, x='star_rating', y='polarity', palette='winter', wid
plt.xlabel('Star Rating', fontsize=14)
plt.ylabel('Polarity', fontsize=14)
plt.title('Relationship Between the Star Rating and Polarity of Reviews', fontsiz
```

```
Text(0.5, 1.0, 'Relationship Between the Star Rating and Polarity of Reviews')
```



Observations:

- From the boxplot, we can infer that as the **star rating increases** the **highest polarity** value is also **increasing**.
- This **doesn't prove** that there is any *positive correlation* between the two features.
- Also, *even at 4 and 5 star ratings* the box plot is showing reviews with **negative polarity** o negative **sentiment**.
    - And there is **positive polarity** at *1 and 2 star ratings*.
- We need to **investigate** this in-depth.

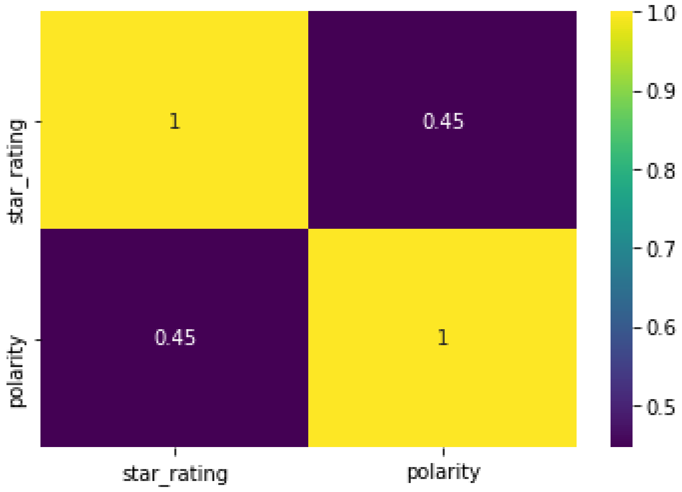
Question 4: Is there a Positive Correlation between the Star Ratings and the Polarity of Reviews?

```
reviews_df[['star_rating', 'polarity']].corr()
```

	star_rating	polarity
star_rating	1.0000	0.4467
polarity	0.4467	1.0000

```
plt.figure(figsize=(6, 4))
sns.heatmap(reviews_df[['star_rating', 'polarity']].corr(), annot=True, cmap='vir
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb1481d4e48>
```



Observations:

- `star_rating` and `polarity` have a **positive correlation** of **0.45**
- This **isn't a strong positive correlation**, but is a positive correlation nonetheless.

Question 6: Why the Boxplot was showing Negative Polarity for High Star Rated Reviews?

- Let's see some examples where both these cases are overlapping i. e. **high star rating** but **low polarity**.

```
for review in reviews_df[(reviews_df['star_rating'] == 5) & (reviews_df['polari
    print(review, '\n')
```

```
this thing frees you from the oppression of horribly cramped power strips

these are sturdy and got me through the worst winter of nyc that i can rememb

this is terrible doesnt get any channels

they work well and you do not get any nasty signal interference with them

worked well even though the cable was bent against the wall mounted tv terrib
```

Observations:

- These reviews contain some words like **horrible**, **worst**, **terrible**, **nasty** that are usually linked with a **negative emotion**.
  - As a result, the **polarity** values are **negative**.

    - But these words are used in order to **praise** the product.
- The **star ratings** are **high** because the customers actually are **satisfied** with the product and showing a **positive sentiment** in this context.

Question 7: Why the Boxplot was showing Positive Polarity for Low Star Rated Reviews?

- Let's see some examples where both these cases are overlapping i. e. **low star rating** but **high polarity**.

```
for review in reviews_df[(reviews_df['star_rating'] == 1) & (reviews_df['polarity
    print(review, '\n')
```

```
not the best sounding amplifier

it was not the correct unit for the speakers even though i put in the model nu

did not work seller is awesome on service

not the company but the product worked flawlessly for two weeks brick since th

not the greatest
```

Observations:

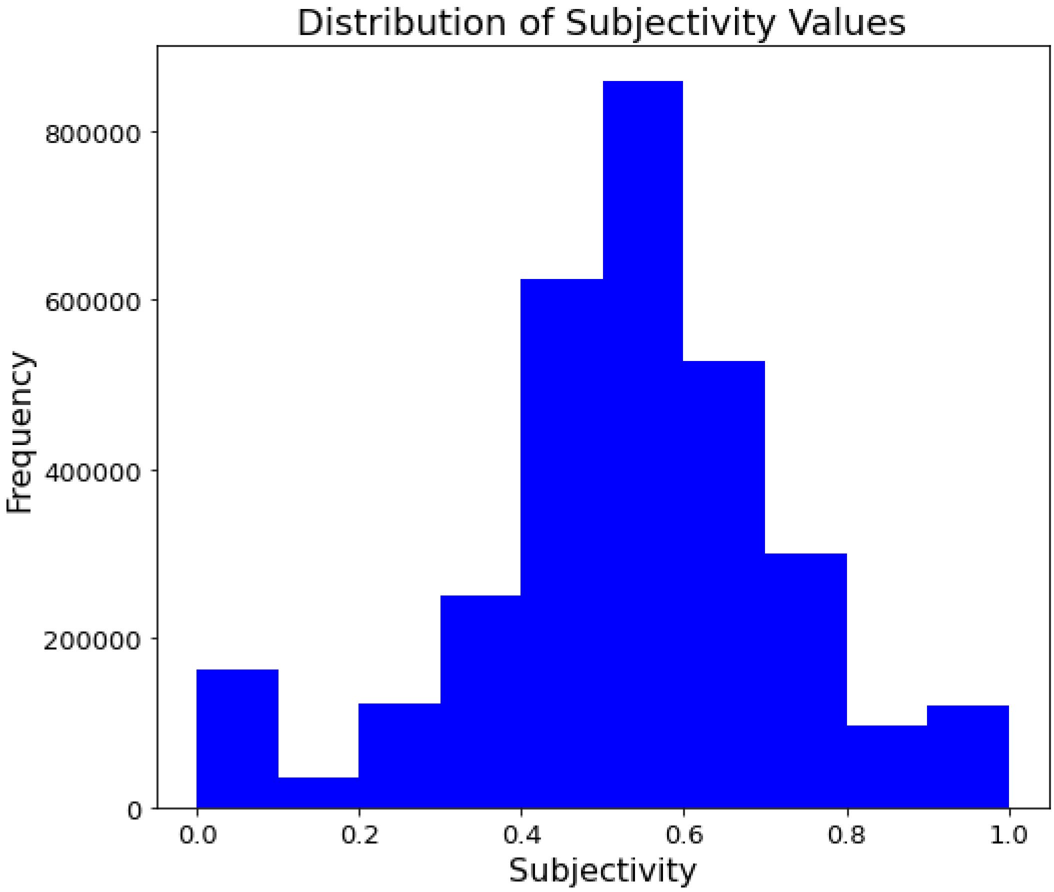
- All these reviews contains words like **best**, **perfectly**, **awesome**, **flawlessly**, **greatest**, that are used to express **positive emotions**.
    - As a result, the **polarity** values are **positive**.
    - But, there is also a **negation** in most of these reviews, due to which these words are actually expressing **dissatisfaction**.
- The **star ratings** for these reviews are **low** because the customers actually are **complaining** about their products and showing **negative sentiment**.

- This *star rating* and *polarity* **issue** needs to be **resolved** before building the sentiment analysis model.
- We need to consider both these features while applying a sentiment to our reviews.

Question 8: How are the Subjectivity values distributed for the Reviews?

```
plt.figure(figsize=(8, 7))
reviews_df['subjectivity'].plot(kind='hist', color='blue', fontsize=13)
plt.xlabel('Subjectivity', fontsize=16)
plt.ylabel('Frequency', fontsize=16)
plt.title('Distribution of Subjectivity Values', fontsize=18)
```

```
Text(0.5, 1.0, 'Distribution of Subjectivity Values')
```



Observations:

- **Subjectivity** values follow a **normal distribution**.
- **More than 80%** values have a *subjectivity* **higher than 0.4**.
- The reviews need to be subjective in order to **build** a **robust** sentiment analysis model.
    - Because *objective reviews* will **downgrade** the **performance** of the model.

Question 9: What should be the Threshold of Subjectivity for the Reviews?

- We need to **set** a **threshold** for subjectivity that will allow us to **remove reviews** from the dataset having subjectivity lower than the threshold value.
- We will experiment with **3** different `subjectivity` values: **0.1**, **0.2**, and **0.3**
- There are a total of **3093869** reviews in the dataset.

Reviews having Subjectivity value of 0.1

```
for review in reviews_df[reviews_df['subjectivity'] == 0.1].sample(5, random_stat
    print(review, '\n')
```

```
maxell once again proved that everything it makes is inferior quality do not l

this meter did the job it was intended to do but we really should have purcha

worth ever $

i plugged it in and all worked as advertised my samsung blue ray player now ta

appears to be a grade above hardware store product
```

- The reviews are quite **objective** at this point.

```
print('Proportion of Reviews left after Subjectivity Threshold set to 0.1:', (len
```

```
Proportion of Reviews left after Subjectivity Threshold set to 0.1: 94.714449
```

Reviews having Subjectivity value of 0.2

```
for review in reviews_df[reviews_df['subjectivity'] == 0.2].sample(5, random_stat
    print(review, '\n')
```

```
this hdmi cable worked well with hd digital as well as blu ray dvd is the fit

thanks for the comunication the saler respond to have a contact in all time i

this battery will not hold a charge it is useless

this product works really well in my truck no bulky at all and it never gets

i bought it for the auto reverse feature and used it as a source in transferr
```

- The **ojectivity** of reviews has **decreased** and sentiment analysis might be easy for human eye.
- But, it will still be **difficult** for a *machine* to **anlyze** the *sentiment* of these reviews at this point.

```
print('Proportion of Reviews left after Subjectivity Threshold set to 0.2:', (len
```

```
Proportion of Reviews left after Subjectivity Threshold set to 0.2: 93.613045
```

Reviews having Subjectivity value of 0.3

```
for review in reviews_df[reviews_df['subjectivity'] == 0.3].sample(5, random_stat
    print(review, '\n')
```

```
it does its job one cannot complain mike the owner was very curtious in answe

works very well

the best i have ever owned

maxell are the best

works well with a low price
```

- The reviews seem to be much **clearer** and **subjective** now.

```
print('Proportion of Reviews left after Subjectivity Threshold set to 0.3:', (len
```

```
Proportion of Reviews left after Subjectivity Threshold set to 0.3: 90.458548
```

Observations:

- Out of the 3 subjectivity values, we get the **best results** when *subjectivity threshold* is set to **0.3**
- The reviews are **subjective** and it is **easy** to **analyze** their *sentiment* as well.
- Also, we will still have about **90%** of the reviews from the dataset, when we set the subjectivity threshold to 0.3

Question 10: What are the Most Common Words in Positive Reviews?

- For **positive** reviews, we are setting `star_rating` to **5** and `polarity` to **1**.

```
wordcloud = WordCloud(width=2500, height=2000, max_words=50,
                      background_color='White'
                      ).generate(str(reviews_df[(reviews_df['star_rating'] == 5)
                                 (reviews_df['polarity'] == 1)].sample(10000, ran


plt.figure(figsize=(10, 10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 2499.5, 1999.5, -0.5)
```

Observations:

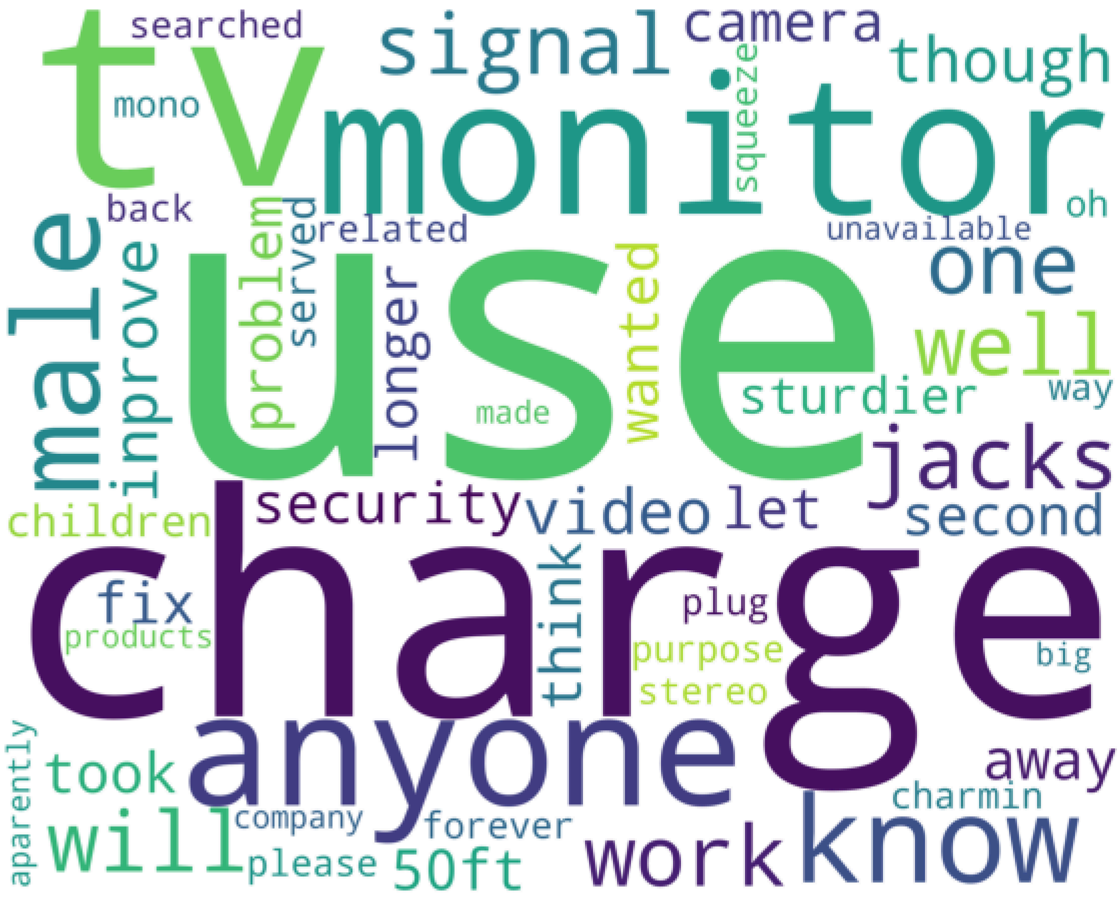
- The most common words in the positive reviews are **perfect**, **awesome**, **perfectly**, **excellent**, etc.
- These words are usually used to indicate a **positive sentiment**.

Question 11: What are the Most Common Words in Neutral Reviews?

- For **neutral** reviews, we are setting `star_rating` to **3** and `polarity` to **0**.

```
wordcloud = WordCloud(width=2500, height=2000, max_words=50,
                      background_color='White'
                      ).generate(str(reviews_df[(reviews_df['star_rating'] == 3)
                                 (reviews_df['polarity'] == 0)].sample(10000, ran

plt.figure(figsize=(10, 10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 2499.5, 1999.5, -0.5)
```



Observations:

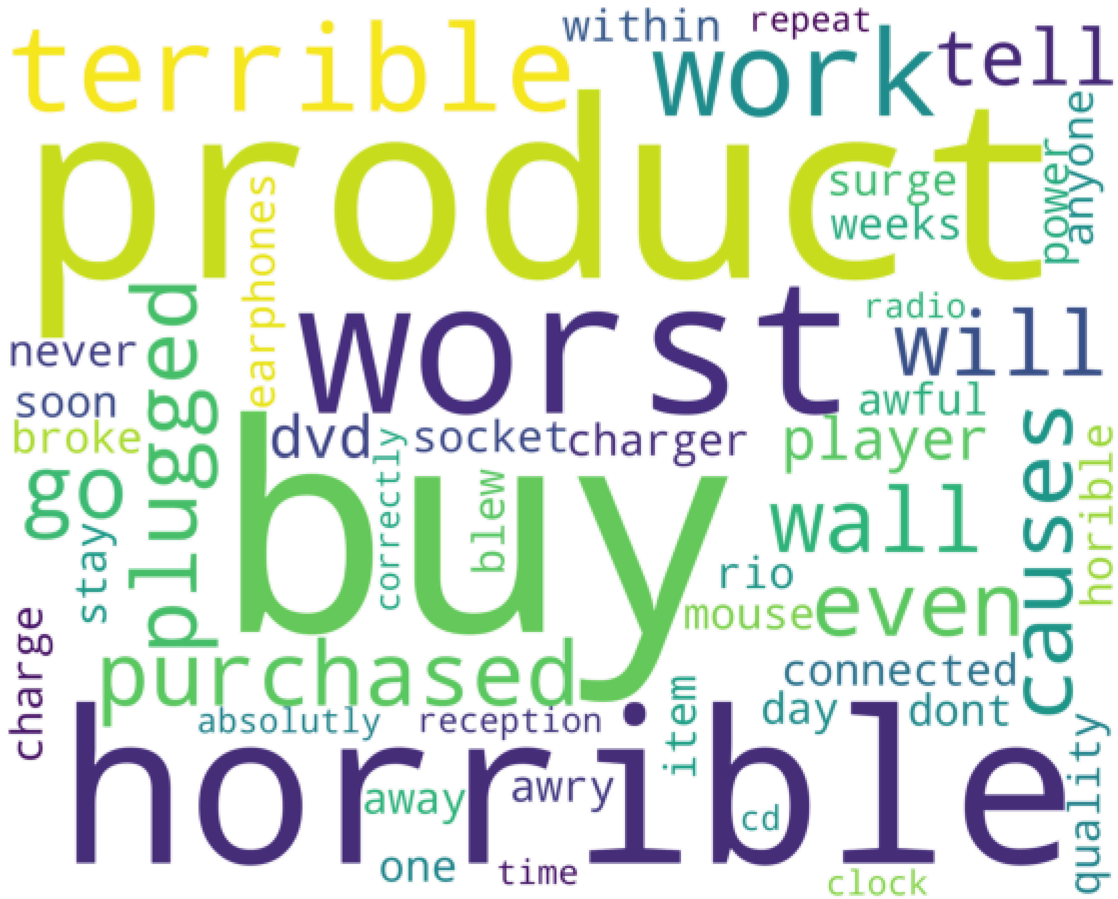
- The *most common words for neutral reviews* are **charge**, **use**, **monitor**, **anyone**, etc.
- These words **don't express** a **clear sentiment** and hence are appropriate for neutral reviews.

Question 12: What are the Most Common Words in Negative Reviews?

- For **negative** reviews, we are setting `star_rating` to **1** and `polarity` to **-1**.

```
wordcloud = WordCloud(width=2500, height=2000, max_words=50,
                      background_color='White'
                      ).generate(str(reviews_df[(reviews_df['star_rating'] == 1)
                                 (reviews_df['polarity'] == -1)]['clean_reviews']


plt.figure(figsize=(10, 10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 2499.5, 1999.5, -0.5)
```



Observations:

- The most common words for negative reviews are **horrible**, **worst**, **terrible**, **awful**, **broke**, etc.
- These words **clearly express** a **negative emotion** and are appropriate for negative reviews.

7. Post Data Processing & Analysis

- After completing the analysis on the data, we can move on towards fitting our Machine Learning models with our data.
- But, our dataset still contains a lot of **redundant columns** in our data which won't help the model in making predictions.
- Also, we need to **remove samples** having **subjectivity lower than** the *subjectivity threshold* value of **0.3**
- And, we need to create a `sentiment` column containing the **labels** for our machine learning model.
- In this section, we will **remove** all the redundant columns, **drop samples** that doesn't satisfy our selection criteria, and then **create** a `sentiment` column.

- We will also be **splitting** the data into two subsets for training and testing purposes.

7.1 Removing Redundant Columns

- We will **remove** every **redundant column** from our dataset.
- We will create a **new dataframe** containing only the **essential features** and use this dataframe down the line.

```
# Removing redundant columns from reviews_df
essential_df = reviews_df[['star_rating', 'clean_reviews', 'polarity', 'subjectiv
essential_df.head()
```

	star_rating	clean_reviews	polarity	subjectivity
0	5	as described	0.0000	0.000
1	5	it works as advertising	0.0000	0.000
2	5	works pissa	0.0000	0.000
3	1	did not work at all	0.0000	0.000
4	5	works well bass is somewhat lacking but is present overall pleased with the item	0.0875	0.375

- The **new dataframe** only contains the **essential** features: `star_rating`, `clean_reviews`, `polarity`, `subjectivity`.

7.2 Removing Samples Having Subjectivity Less Than 0.3

- Here, we will **remove** the **samples** having **subjectivity lower than** the subjectivity threshold value of **0.3**

```
# Checking the current minimum value of subjectivity
essential_df['subjectivity'].min()
```

```
0.0
```

```
essential_df = essential_df[essential_df['subjectivity'] >= 0.3]
```

```
# Checking the minimum value of subjectivity after removing samples
essential_df['subjectivity'].min()
```

```
0.3
```

- The **minimun value** of **subjectivity** has increased from 0 to **0.3**
- We have successfully **removed** every **sample** having *subjectivity less than 0.3*

7.3 Creating Sentiment Column

- Now, we will create a `sentiment` column which will provide the labels for our training samples used in the Machine Learning models.
- We will use both the `star_rating` and `polarity` values to **divide** our reviews into different **sentiments**.

- For **positive** reviews, we are using a `star_rating` **higher than 3** and a `polarity` value **greater than or equal to 0.5**.
- We have to use such a *high value of polarity* due to the **large volume** of *positive reviews*.
- We need to **reduce** the **size** of our **dataset** for training, otherwise our **session will crash**, due to *shortage of RAM*.

```
essential_df[(essential_df['star_rating'] > 3) & (essential_df['polarity'] >= 0.5
```

	star_rating	clean_reviews	polarity	subjectivity
6	5	wish i could give this product more than five stars lifesaver	0.500	0.5000
7	5	works great	0.800	0.7500
8	4	great sound and compact battery life seems good happy with this product	0.675	0.68
10	5	alll good	0.700	0.6000

```
positive_df = essential_df[(essential_df['star_rating'] > 3) & (essential_df['pol

positive_df['sentiment'] = 'positive'

positive_df.head()
```

	star_rating	clean_reviews	polarity	subjectivity	sentiment
6	5	wish i could give this product more than five stars lifesaver	0.500	0.5000	positive
7	5	works great	0.800	0.7500	positive
8	4	great sound and compact battery life seems good happy with this product	0.675	0.6875	positive
10	5	alll good	0.700	0.6000	positive

- For **neutral** reviews, we are using a `star_rating` **equal to 3** and a `polarity` value **between -0.1 and 0.1**.

```
essential_df[(essential_df['star_rating'] == 3) & (essential_df['polarity'] >= -0
```

	star_rating	clean_reviews	polarity	subjectivity
120	3	these do not hold as long of a charge as the green capped compeditors even though the mw hr is advertised to be higher however these do work as batteries and will hold a wouldecent charge however the charge is shorter lived than other batteries of the same type i have tried i do use these often also these did fry one of my usb ports so use only in outlet usb ports	-0.0208	0.4500
147	3	i bought this to replace a lost neoprene case for my qc20i earbuds this case is just ok br br it seems very cheaply made and it is kind of hard to use because the zipper is a little stiff if it was more expensive i would have returned it but there is really no point i would get a buck or two back after return shipping	0.0827	0.5252
		product is just okay volume level is not very good i am used to going louder charge only		

```
neutral_df = essential_df[(essential_df['star_rating'] == 3) & (essential_df['pol

neutral_df['sentiment'] = 'neutral'

neutral_df.head()
```

	star_rating	clean_reviews	polarity	subjectivity	sentiment
120	3	these do not hold as long of a charge as the green capped competitors even though the mw hr is advertised to be higher however these do work as batteries and will hold a wouldecent charge however the charge is shorter lived than other batteries of the same type i have tried i do use these often also these did fry one of my usb ports so use only in outlet usb ports	-0.0208	0.4500	neutral
147	3	i bought this to replace a lost neoprene case for my qc20i earbuds this case is just ok br br it seems very cheaply made and it is kind of hard to use because the zipper is a little stiff if it was more expensive i would have returned it but there is really no point i would get a buck or two back after return shipping	0.0827	0.5252	neutral

- For **negative** reviews, we are using a `star_rating` **less than 3** and a `polarity` value **less than 0**.

```
essential_df[(essential_df['star_rating'] < 3) & (essential_df['polarity'] < 0)].
```

	star_rating	clean_reviews	polarity	subjectivity
49	2	horrible	-1.0000	1.0000
75	1	day 20 november 2014 acquired the jbl portable load indoor outdoor bluetooth speaker black and hardly used it and 6 16 15 she has stopped working just take full charge when left more than 24 hours yet their working time is no more than 2hs the few times i used was always loaded with accessories that accompanied it i live in brasilia df brazil not with the product warranty or aq even there in the usa she when turned on provides a noise as if in short circuit and after a while for the noise only is working connected to the charger br it is expensive and that left me high and dry br disappointed	-0.0719	0.4677

```
negative_df = essential_df[(essential_df['star_rating'] < 3) & (essential_df['pol

negative_df['sentiment'] = 'negative'

negative_df.head()
```

	star_rating	clean_reviews	polarity	subjectivity	sentiment
49	2	horrible	-1.0000	1.0000	negative
75	1	day 20 november 2014 acquired the jbl portable load indoor outdoor bluetooth speaker black and hardly used it and 6 16 15 she has stopped working just take full charge when left more than 24 hours yet their working time is no more than 2hs the few times i used was always loaded with accessories that accompanied it i live in brasilia df brazil not with the product warranty or aq even there in the usa she when turned on provides a noise as if in short circuit and	-0.0719	0.4677	negative

- **Joining** all **3** dataframes to create our final dataset.

```
sentiment_df = pd.concat([positive_df, neutral_df, negative_df], ignore_index=Tru
sentiment_df.head()
```

	star_rating	clean_reviews	polarity	subjectivity	sentiment
0	5	wish i could give this product more than five stars lifesaver	0.500	0.5000	positive
1	5	works great	0.800	0.7500	positive
2	4	great sound and compact battery life seems good happy with this product	0.675	0.6875	positive
3	5	alll good	0.700	0.6000	positi

```
sentiment_df.tail()
```

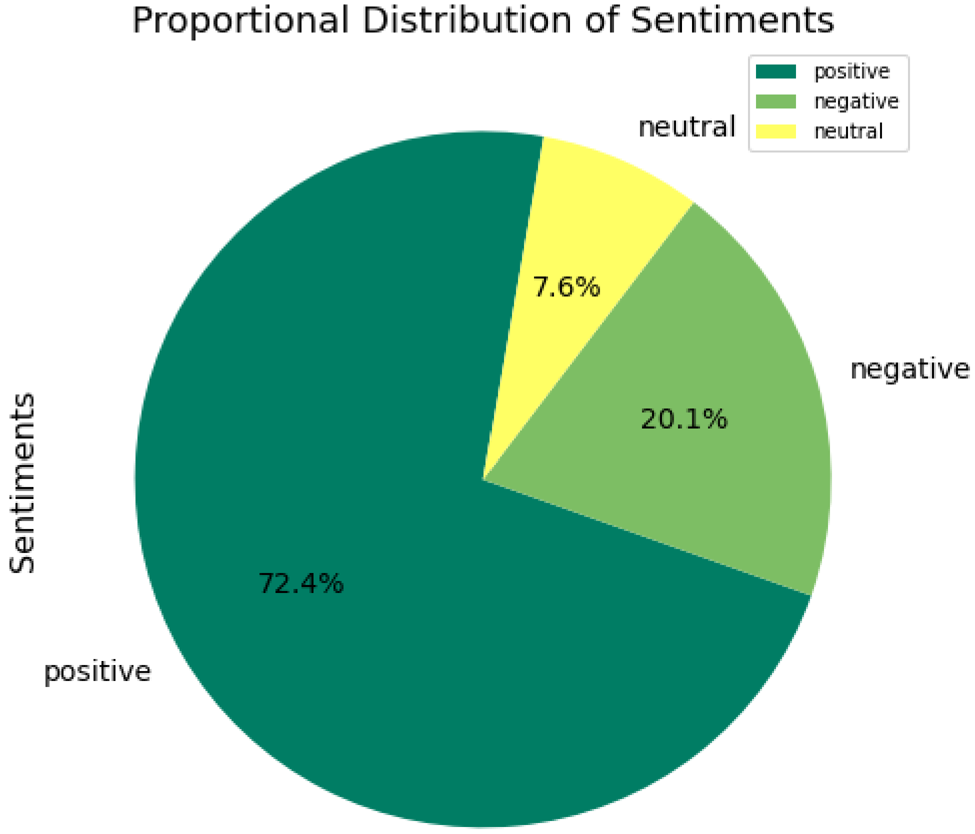
	star_rating	clean_reviews	polarity	subjectivity	sentiment
762084	2	this antenna when installed outside provided only slight improvement over &quotrabbit ear&quot antenna for local station reception	-0.0417	0.3042	negative
762085	1	although the concept is good diamond has done a poor job of designing the product as well as a poor job of supporting it the volume is too low even at its highest setting the hold switch is poorly designed so that you cannot tell if its engaged the clip holding the battery in is easily broken and the add on 32meg external memory did	-0.1170	0.4230	negative

- The final dataset contains the **selected reviews** along with their respective **sentiments**.

Proportional Distribution of Sentiments

```
# Plotting the proportional distribution of sentiments
sentiment_df['sentiment'].value_counts().plot(kind='pie', autopct='%1.1f%%', star
                                               fontsize=14, legend=True, cmap='summe
plt.ylabel('Sentiments', fontsize=16)
plt.title('Proportional Distribution of Sentiments', fontsize=18)
```

```
Text(0.5, 1.0, 'Proportional Distribution of Sentiments')
```



Observations:

- Our final dataset contains about **72.4% positive** reviews, **20.1% negative** reviews, and **7.6% neutral** reviews.

Saving the Final Dataset as a CSV File

- We will **save** our final dataset as a **csv file** in the local system.
- This will allow us to **resume** from this point onwards if we want to make changes to the model building process.

```
# Saving the dataframe to a csv file.
sentiment_df.to_csv('review_data.csv', index=False, encoding='utf-8')
```

```
# To load the saved csv file into a dataframe
sentiment_df = pd.read_csv('review_data.csv')
sentiment_df.head()
```

7.4 Data Splitting

- Now, we will **split** the dataset into **Train** and **Test** subsets.
- We will use **80%** data for **training** and the remaining **20%** data for **testing** our models.
- First, we will **separate** the **reviews** and their respective sentiment **labels** from the data.

```
# Separating the Reviews from the dataset
X = sentiment_df['clean_reviews'].values
X[:5]
```

```
array(['wish i could give this product more than five stars lifesaver',
       'works great',
       'great sound and compact battery life seems good happy with this
product',
       'alll good',
       'excellent gain in radio frequency reception over the stock antenna
that came with the radio'],
      dtype=object)
```

```
# Separating the labels
y = sentiment_df['sentiment'].values
y[:5]
```

```
array(['positive', 'positive', 'positive', 'positive', 'positive'],
      dtype=object)
```

- After **separating** the *reviews* and *labels*, we will **split** the data into **train** and **test sets**.

```
# Using scikit-learn's train_test_split function to split the dataset into train
# 80% of the data will be in the train set and 20% in the test set, as specified
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

```
# Checking the shapes of the training and test sets.
print('Training Data Shape:', X_train.shape, y_train.shape)
print('Testing Data Shape:', X_test.shape, y_test.shape)
```

```
Training Data Shape: (609671,) (609671,)
Testing Data Shape: (152418,) (152418,)
```

- The data has been **divided** into training and test sets.

8. Model Development & Evaluation

- In this section, we will be **building** our Machine Learning models and fitting them with the training data.

8.1 Building Machine Learning Model

- Building a `tokenizer` function that will **split** each review into a **list of tokens**.
- A **token** is a **single word** in this case, and the review will be splitted on a **single white space**.

```
def tokenizer(text):
    return text.split()
```

- Building a **TFIDF Vectorizer**.
- In this, first we create a **vocabulary** of **unique tokens** from the entire set of **documents** (i. e. **reviews**).