```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
os.chdir('/content/drive/MyDrive')
```

```
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('Amazon_Unlocked_Mobile.csv')

# Now, you can work with your data using the DataFrame 'df'
```

```
df.head()
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	It works good but it goes slow sometimes but i...	0.0
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	Great phone to replace my lost phone. The only...	0.0

```
# Check for null values in the DataFrame
null_values = df.isnull().sum()

# Display the columns with null values and their counts
print("Columns with Null Values:")
print(null_values[null_values > 0])
```

```
Columns with Null Values:
Brand Name      65171
Price            5933
Reviews            62
Review Votes    12296
dtype: int64
```

```
# Get unique values for each column
unique_values = df.nunique()

# Display the unique values for each column
print("Unique Values for Each Column:")
print(unique_values)
```

```
Unique Values for Each Column:
Product Name      4410
Brand Name         384
Price             1754
Rating               5
Reviews         162491
Review Votes       241
dtype: int64
```

```
# Get a description of the data (e.g., mean, min, max, etc.) for numeric columns
data_description = df.describe()

# Display the data description
print("Description of the Data:")
print(data_description)
```

```
Description of the Data:
               Price         Rating   Review Votes
count  407907.000000  413840.000000  401544.000000
mean      226.867155       3.819578       1.507237
std       273.006259       1.548216       9.163853
min         1.730000       1.000000       0.000000
25%        79.990000       3.000000       0.000000
50%       144.710000       5.000000       0.000000
75%       269.990000       5.000000       1.000000
max      2598.000000       5.000000     645.000000
```

```
# Display information about the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 413840 entries, 0 to 413839
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Product Name  413840 non-null  object
 1   Brand Name    348669 non-null  object
 2   Price         407907 non-null  float64
 3   Rating        413840 non-null  int64
 4   Reviews       413778 non-null  object
 5   Review Votes  401544 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 18.9+ MB
```

```
# Get the column names of the DataFrame
column_names = df.columns

# Display the column names
print("Column Names:")
print(column_names)
```

```
Column Names:
Index(['Product Name', 'Brand Name', 'Price', 'Rating', 'Reviews',
       'Review Votes'],
      dtype='object')
```

```
# Drop records with null values in the "Reviews" column
df = df.dropna(subset=["Reviews"])
```

```
# Substitute null values in the "Review Votes" column with zero
df["Review Votes"].fillna(0, inplace=True)
```

```
<ipython-input-11-78e3f223c195>:2: 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["Review Votes"].fillna(0, inplace=True)
```

```
# Substitute null values in the "Price" column with the median value (144.71)
median_price = df["Price"].median()
df["Price"].fillna(median_price, inplace=True)
```

```
<ipython-input-12-ea18c92ae049>:3: 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["Price"].fillna(median_price, inplace=True)
```

```
# Remove duplicated records
df = df.drop_duplicates()
```

```
# Get unique items in the "Brand Name" column
unique_brands = df["Brand Name"].unique()

# Display the unique brand names
print("Unique Brand Names:")
print(unique_brands)
```

```
'BlueCosmo' 'GT Star' 'Boost Mobile' 'SANYO' 'sony' 'CablesOnline'
'Casio' 'CAT PHONES' 'CAT' 'Caterpillar' 'CATERPILLAR' 'CBSKY' 'CECTDIGI'
'CELLALLURE' 'Unnecto' 'CHSLING' 'CNPGD'
'Conquest S8 Pro 3GB Smartphone (Black)' 'Consumer Cellular' 'Junlan'
```

```
'Weil Technology (HK) Limited' '1PartsBuy' 'Lenovo Manufacturer' 'DK'
'LeEco' 'Letv' 'lg' 'LG Electronic' 'LG Electronics' 'Wireless One'
'LG Electronics MobileComm USA' 'LGIC' 'P710' 'Risio' 'GL' 'LGG'
'LifeProof' 'Sudroid' 'Lumen' 'Mango Natural' 'MANN ZUG S Rugged Phone'
'Marozi International Ltd' 'MEIZU' 'Meizu' 'M-HORSE' 'Microsoft'
'microsoft' 'Fosler Corporation' 'FOSLER COMPANY' 'Eson'
'Selected-Gadgets' 'Sony Ericsson' 'Mophie' 'mophie' 'ShopTech4Less'
'Motorola X 2nd gen XT1093' 'motorola' 'Moto X' 'MOTOROLA'
'Moxee Technologies' 'MOCREO' 'KINGWELL' 'Neoix' 'Sony'
'New Nextel Rugged Motorola I680 Cell Phone' 'Higoo' 'Sdeals' 'Sidekick'
'bbfone' 'Newsunshine' 'MICROSOFT' 'Grade - A' 'nokia' 'AT&T'
'Nuu Mobile' 'Ofeely' 'Ofeely company' 'OLA Products' 'OnePlus' 'Orbic'
'Oukitel' 'UMI' 'Otium' 'Defender' 'Ultrabox' 'OUKITEL' 'Padgene'
'Sprint' 'PALM' 'Pandaoo' 'pantech' 'PCD' 'Tuscan Leveling System'
'Pharos' 'PHAROS' 'Plum Mobile' 'Polaroid' 'Posh Mobile'
'AMM Global Enterprises' 'QJO' 'Que Products' 'Quick-get' 'RCA' 'Rim'
'REACH' 'RomaCostume' 'SENTEL' 'RugGear' 'Saliency' 'SAMSUNG' 'Samssung'
'Luxury Phone' 'Galaxy' 'Samsung Korea LTD' 'BlueSolar'
'samsung galaxy international inc' 'Global Services' 'samsung'
'PowerMoxie' 'Unknown' 'Samsung Galaxy' 'Samsybg Galaxy'
'Samsung international' 'Samsung International' 'TOP' 'Vostrostone'
'Samsung Korea' 'samsung galaxy' 'red' 'Sanyo' 'Sanyo, Katana'
'Seawolf Technologies' 'SGH-T199' 'Sharp' 'SIAM' 'Blackphone'
'Silent Circle' 'SKY Devices' 'Sky Devices' 'VEGA' 'GEEKERA' 'JoyGeek'
'Smart&Cool' 'KROO' 'IPRO' 'Yota'
'SHENZHEN SNOPOW OUTDOOR TECHNOLOGY CO.,LTD'
'SHENZHEN SNOPOW OUTDOOR TECHNOLOGY CO.LTD' 'Social'
'Unlocked Cell Phone' 'SOLE Mobile' 'Sonim Technologies' 'SonyEricsson'
'SONY ERICSSON' 'sony ericsson' 'Sony Ericsson Mobile' 'Sony/Ericsson'
'SONY' 'EL GUAPO' 'Sonim' 'ssiony' 'Stoga' 'TracFone' 'HJ Wireless'
'Blue' 'Samsung/Straight Talk' 'PAE' 'ling' 'LuMen' 'Supersonic' 'SVP'
'MTM Trading LLC' 'Turbosight LLD' 'Tell' 'The NoPhone' 'Thetford Marine'
'SyuanTech Co., Ltd' 'THL' 'ThL' 'Tivax' 'T-Mobile' 'Danger Inc.'
'LiteFuze' 'DreamsEden' 'ToShare Tech' 'TOTO' 'TSJYING' 'TUMI' 'iFcane'
'ifcane' 'Peek' 'Wmicro' 'UHAPPY' 'Purplelan' 'Tsing' 'WRTeam' 'Coolpad'
'Cubot' 'CUBOT' 'Concox' 'KIKAR' 'the Nokia' 'NUU Mobile' 'OceanCross'
'UTStarcom' 'Verizon' 'Vernee' 'verykool' 'VIP' 'Ut Starcom'
'Visual Land' 'Wogiz' 'q' 'Shenzhen Xin Sheng Shang Technology Co.,Ltd'
'Xiaomi' 'SZ Wave' 'ZTE(USA) Wireless' 'Dead Sea Secrets'
```

```
# @title
# Define a mapping of old brand names to new corrected brand names
brand_name_mapping = {
"HTM": "HTC",
"Jethro": "Jethro",
"e passion": "ePassion",
"Cedar Tree Technologies": "Cedar Tree Technologies",
"Indigi": "Indigi",
"Phone Baby": "Phone Baby",
"OtterBox": "OtterBox",
"Lenovo": "Lenovo",
"Huawei": "Huawei",
"JUNING": "JUNING",
"Elephone": "Elephone",
"Plum": "Plum",
"VKworld": "VKworld",
"NOKIA": "Nokia",
"Ulefone": "Ulefone",
"Jiuhe": "Jiuhe",
"inDigi": "Indigi",
"Acer": "Acer",
"Aeku": "Aeku",
"AKUA": "AKUA",
"Alcatel": "Alcatel",
"TCL Mobile": "TCL",
"LG": "LG",
"amar": "Amar",
"Amazon": "Amazon",
"Odysseus": "Odysseus",
"Digital SNITCH": "Digital SNITCH",
"star": "Star",
"Android": "Android",
"Yezz": "Yezz",
"Yezz Wireless Ltd.": "Yezz",
"Apple": "Apple",
"Apple Computer": "Apple",
"Certified Refurbished": "Certified Refurbished",
"Amazon.com, LLC *** KEEP PORules ACTIVE ***": "Amazon",
"apple": "Apple",
"ARGOM TECH": "Argom Tech",
"Asus": "Asus",
"ASUS": "Asus",
"ASUS Computers": "Asus",
"asus": "Asus",
"BlackBerry": "BlackBerry",
"Motorola": "Motorola",
"Palm": "Palm",
"Pantech": "Pantech",
"ZTE": "ZTE",
"ATT": "AT&T",
"UnAssigned": "UnAssigned",
"Atoah": "Atoah",
"BlackBerry Storm 9530 Smartphone Unlocked GSM Wireless Handheld Device w/Camera Bluetooth 3.25\" Touchscreen LCD": "BlackBerry",
"WorryFree Gadgets": "WorryFree Gadgets",
"worryfree": "WorryFree",
"iDROID USA": "iDROID USA",
"Blackberry (Rim)": "BlackBerry",
"Research In Motion": "BlackBerry",
"BLACKBERRY": "BlackBerry",
"Black Berry": "BlackBerry",
"Blackberry": "BlackBerry",
"blackberry": "BlackBerry",
"RIM": "BlackBerry",
"Research in Motion": "BlackBerry",
"Blackberrry": "BlackBerry",
"Storm": "BlackBerry",
"Blackview": "Blackview",
"BLU": "BLU",
"Various": "Various",
"CT-Miami LLC": "CT-Miami",
"blu": "BLU",
"BLUBOO": "BLUBOO",
"Bluboo": "BLUBOO",
"Iridium": "Iridium",
"AeroAntenna": "AeroAntenna",
"BlueCosmo": "BlueCosmo",
"GT Star": "GT Star",
"Boost Mobile": "Boost Mobile",
"SANYO": "Sanyo",
"sony": "Sony",
"CablesOnline": "CablesOnline",
"Casio": "Casio",
"CAT PHONES": "CAT",
"CAT": "CAT",
"Caterpillar": "CAT",
"CATERPILLAR": "CAT",
"CBSKY": "CBSKY",
"CECTDIGI": "CECTDIGI",
"CELLALLURE": "CellAllure",
"Unnecto": "Unnecto",
"CHSLING": "CHSLING",
"CNPGD": "CNPGD",
"Conquest S8 Pro 3GB Smartphone (Black)": "Conquest",
"Consumer Cellular": "Consumer Cellular",
"Junlan": "Junlan",
"Quality Technology Industrial Co., ltd.": "Quality Technology",
"Crony": "Crony",
"Crown": "Crown",
"China": "China",
"HTC": "HTC",
"Dell Computers": "Dell",
"Docooler": "Docooler",
"DIKOO": "DIKOO",
"DOOGEE": "DOOGEE",
"Doogee": "DOOGEE",
"KVD": "KVD",
"Doro": "Doro",
"DTECH": "DTECH",
"Dupad Story": "Dupad Story",
"EasyN": "EasyN",
"ECOO": "ECOO",
"Emporia": "Emporia",
"FIGO": "FIGO",
"FiGo": "FIGO",
"Maxwest": "Maxwest",
"Futuretech": "Futuretech",
"Wool": "Wool",
"S7 active": "S7 active",
"Evergreen Flag & Garden": "Evergreen Flag & Garden",
"Generic": "Generic",
"Getnord": "Getnord",
"Glocalme®": "Glocalme",
"IPRO Group": "IPRO",
"Goldengulf": "Goldengulf",
"goldengulf": "Goldengulf",
"GOOGLE": "Google",
"Google Pixel": "Google",
```

```
  "GOSO": "GOSO",
  "Inmarsat": "Inmarsat",
  "GreatCall": "GreatCall",
  "Jitterbug": "Jitterbug",
  "GT STAR": "GT STAR",
  "GX": "GX",
  "H2O": "H2O",
  "china": "China",
  "Victor": "Victor",
  "shenzhen jinwanyi company": "Shenzhen Jinwanyi",
  "HOMTOM": "HOMTOM",
  "JHM": "JHM",
  "HOMTOM HT7": "HOMTOM",
  "HP Handheld": "HP",
  "VoiceStream": "VoiceStream",
  "HP": "HP",
  "HTC America": "HTC",
  "htc": "HTC",
  "Huadoo": "Huadoo",
  "HUAWEI": "Huawei",
  "XJKJ": "XJKJ",
  "LightInTheBox": "LightInTheBox",
  "winwinzonece": "winwinzonece",
  "HyRich": "HyRich",
  "Star": "Star",
  "px phone": "px phone",
  "indigi": "Indigi",
  "Indigi®": "Indigi",
  "iNew": "iNew",
  "Spicy World": "Spicy World",
  "iVAPO": "iVAPO",
  "IPRO GROUP": "IPRO",
  "Ipro": "IPRO",
  "iRULU": "iRULU",
  "JANIZZ": "JANIZZ",
  "Jersa": "Jersa",
  "Trait-Tech": "Trait-Tech",
  "JIAKE": "JIAKE",
  "GrandElectronics": "GrandElectronics",
  "Jiayu": "Jiayu",
  "John's": "John's",
  "John's Phone": "John's Phone",
  "Eachbid": "Eachbid",
  "Kata": "Kata",
  "katito": "Katito",
  "KINGZONE": "KINGZONE",
  "Kivors": "Kivors",
  "Kocaso": "Kocaso",
  "KOCASO": "Kocaso",
  "KOOTION": "KOOTION",
  "Kovee Tech": "Kovee Tech",
  "Kyocera": "Kyocera",
  "LandRum": "LandRum",
  "LBER": "LBER",
  "Leagoo": "Leagoo",
  "OTEDA Industrial Co., Limited": "OTEDA Industrial Co., Limited",
  "JINHAIHUAHUI": "JINHAIHUAHUI",
  "WEIL TECHNOLOGY (HK) Limited": "WEIL TECHNOLOGY (HK) Limited",
  "iPartsBuy": "iPartsBuy",
  "Lenovo Manufacturer": "Lenovo",
  "DK": "DK",
  "LeEco": "LeEco",
  "Letv": "LeEco",
  "lg": "LG",
  "LG Electronic": "LG",
  "LG Electronics": "LG",
  "Wireless One": "Wireless One",
  "LG Electronics MobileComm USA": "LG",
  "LGIC": "LGIC",
  "P710": "P710",
  "Risio": "Risio",
  "GL": "GL",
  "LGG": "LGG",
  "LifeProof": "LifeProof",
  "Sudroid": "Sudroid",
  "Lumen": "Lumen",
  "Mango Natural": "Mango Natural",
  "MANN ZUG S Rugged Phone": "MANN ZUG",
  "Marozi International Ltd": "Marozi International Ltd",
  "MEIZU": "Meizu",
  "Meizu": "Meizu",
  "M-HORSE": "M-HORSE",
  "Microsoft": "Microsoft",
  "microsoft": "Microsoft",
  "Fosler Corporation": "Fosler Corporation",
  "FOSLER COMPANY": "Fosler Corporation",
  "Eson": "Eson",
  "Selected-Gadgets": "Selected-Gadgets",
  "Sony Ericsson": "Sony Ericsson",
  "Mophie": "Mophie",
  "mophie": "Mophie",
  "ShopTech4Less": "ShopTech4Less",
  "Motorola X 2nd gen XT1093": "Motorola",
  "motorola": "Motorola",
  "Moto X": "Motorola",
  "MOTOROLA": "Motorola",
  "Moxee Technologies": "Moxee Technologies",
  "MOCREO": "MOCREO",
  "KINGWELL": "KINGWELL",
  "Neoix": "Neoix",
  "Sony": "Sony",
  "New Nextel Rugged Motorola I680 Cell Phone": "Motorola",
  "Higoo": "Higoo",
  "Sdeals": "Sdeals",
  "Sidekick": "Sidekick",
  "bbfone": "bbfone",
  "Newsunshine": "Newsunshine",
  "MICROSOFT": "Microsoft",
  "Grade - A": "Grade - A",
  "nokia": "Nokia",
  "AT&T": "AT&T",
  "Nuu Mobile": "Nuu Mobile",
  "Ofeely": "Ofeely",
  "Ofeely company": "Ofeely",
  "OLA Products": "OLA Products",
  "OnePlus": "OnePlus",
  "Orbic": "Orbic",
  "Oukitel": "Oukitel",
  "UMI": "UMI",
  "Otium": "Otium",
  "Defender": "Defender",
  "Ultrabox": "Ultrabox",
  "OUKITEL": "OUKITEL",
  "Padgene": "Padgene",
  "Sprint": "Sprint",
  "PALM": "Palm",
  "Pandaoo": "Pandaoo",
  "pantech": "Pantech",
  "PCD": "PCD",
  "Tuscan Leveling System": "Tuscan Leveling System",
  "Pharos": "Pharos",
  "PHAROS": "Pharos",
  "Plum Mobile": "Plum Mobile",
  "Polaroid": "Polaroid",
  "Posh Mobile": "Posh Mobile",
  "AMM Global Enterprises": "AMM Global Enterprises",
  "QJO": "QJO",
  "Que Products": "Que Products",
  "Quick-get": "Quick-get",
  "RCA": "RCA",
  "Rim": "Rim",
  "REACH": "REACH",
  "RomaCostume": "RomaCostume",
  "SENTEL": "SENTEL",
  "RugGear": "RugGear",
  "Saliency": "Saliency",
  "SAMSUNG": "Samsung",
  "Samssung": "Samsung",
  "Luxury Phone": "Luxury Phone",
  "Galaxy": "Samsung",
  "Samsung Korea LTD": "Samsung",
  "BlueSolar": "BlueSolar",
  "samsung galaxy international inc": "Samsung",
  "samsung": "Samsung",
  "PowerMoxie": "PowerMoxie",
  "Unknown": "Unknown",
  "Samsung Galaxy": "Samsung",
  "Samsybg Galaxy": "Samsung",
  "Samsung international": "Samsung",
  "Samsung International": "Samsung",
  "TOP": "TOP",
  "Vostrostone": "Vostrostone",
  "Samsung Korea": "Samsung",
```

```
    "samsung galaxy": "Samsung",
    "red": "Red",
    "Sanyo": "Sanyo",
    "Sanyo, Katana": "Sanyo",
    "Seawolf Technologies": "Seawolf Technologies",
    "SGH-T199": "SGH-T199",
    "Sharp": "Sharp",
    "SIAM": "SIAM",
    "Blackphone": "Blackphone",
    "Silent Circle": "Silent Circle",
    "SKY Devices": "SKY Devices",
    "Sky Devices": "SKY Devices",
    "VEGA": "VEGA",
    "GEEKERA": "GEEKERA",
    "JoyGeek": "JoyGeek",
    "Smart&Cool": "Smart&Cool",
    "KROO": "KROO",
    "IPRO": "IPRO",
    "Yota": "Yota",
    "SHENZHEN SNOPOW OUTDOOR TECHNOLOGY CO.,LTD": "SHENZHEN SNOPOW",
    "SHENZHEN SNOPOW OUTDOOR TECHNOLOGY CO.LTD": "SHENZHEN SNOPOW",
    "Social": "Social",
    "Unlocked Cell Phone": "Unlocked Cell Phone",
    "SOLE Mobile": "SOLE Mobile",
    "Sonim Technologies": "Sonim",
    "SonyEricsson": "Sony Ericsson",
    "SONY ERICSSON": "Sony Ericsson",
    "sony ericsson": "Sony Ericsson",
    "Sony Ericsson Mobile": "Sony Ericsson",
    "Sony/Ericsson": "Sony Ericsson",
    "SONY": "Sony",
    "EL GUAPO": "EL GUAPO",
    "Sonim": "Sonim",
    "ssiony": "Sony",
    "Stoga": "Stoga",
    "TracFone": "TracFone",
    "HJ Wireless": "HJ Wireless",
    "Blue": "Blue",
    "Samsung/Straight Talk": "Samsung/Straight Talk",
    "PAE": "PAE",
    "ling": "Ling",
    "LuMen": "LuMen",
    "Supersonic": "Supersonic",
    "SVP": "SVP",
    "MTM Trading LLC": "MTM Trading LLC",
    "Turbosight LLD": "Turbosight LLD",
    "Tell": "Tell",
    "The NoPhone": "The NoPhone",
    "Thetford Marine": "Thetford Marine",
    "SyuanTech Co., Ltd": "SyuanTech",
    "THL": "THL",
    "ThL": "THL",
    "Tivax": "Tivax",
    "T-Mobile": "T-Mobile",
    "Danger Inc.": "Danger Inc.",
    "LiteFuze": "LiteFuze",
    "DreamsEden": "DreamsEden",
    "ToShare Tech": "ToShare Tech",
    "TOTO": "TOTO",
    "TSJYING": "TSJYING",
    "TUMI": "TUMI",
    "iFcane": "iFcane",
    "ifcane": "iFcane",
    "Peek": "Peek",
    "Wmicro": "Wmicro",
    "UHAPPY": "UHAPPY",
    "Purplelan": "Purplelan",
    "Tsing": "Tsing",
    "WRTeam": "WRTeam",
    "Coolpad": "Coolpad",
    "Cubot": "Cubot",
    "CUBOT": "Cubot",
    "Concox": "Concox",
    "KIKAR": "KIKAR",
    "the Nokia": "Nokia",
    "NUU Mobile": "NUU Mobile",
    "OceanCross": "OceanCross",
    "UTStarcom": "UTStarcom",
    "Verizon": "Verizon",
    "Vernee": "Vernee",
    "verykool": "verykool",
    "VIP": "VIP",
    "Ut Starcom": "UTStarcom",
    "Visual Land": "Visual Land",
    "Wogiz": "Wogiz",
    "q": "Q",
    "Shenzhen Xin Sheng Shang Technology Co.,Ltd": "Shenzhen Xin Sheng Shang",
    "Xiaomi": "Xiaomi",
    "SZ Wave": "SZ Wave",
    "ZTE(USA) Wireless": "ZTE",
    "Dead Sea Secrets": "Dead Sea Secrets",
    "ZTE Corporation": "ZTE",
    "Zte": "ZTE"
    # Add more mappings as needed
}

# Use the replace method to apply the mapping to the 'Brand Name' column
df['Brand Name'] = df['Brand Name'].replace(brand_name_mapping)

# Check the unique values in the 'Brand Name' column after the changes
df1 = df['Brand Name'].unique()
print("Unique Brand Names:")
print(df1)
```

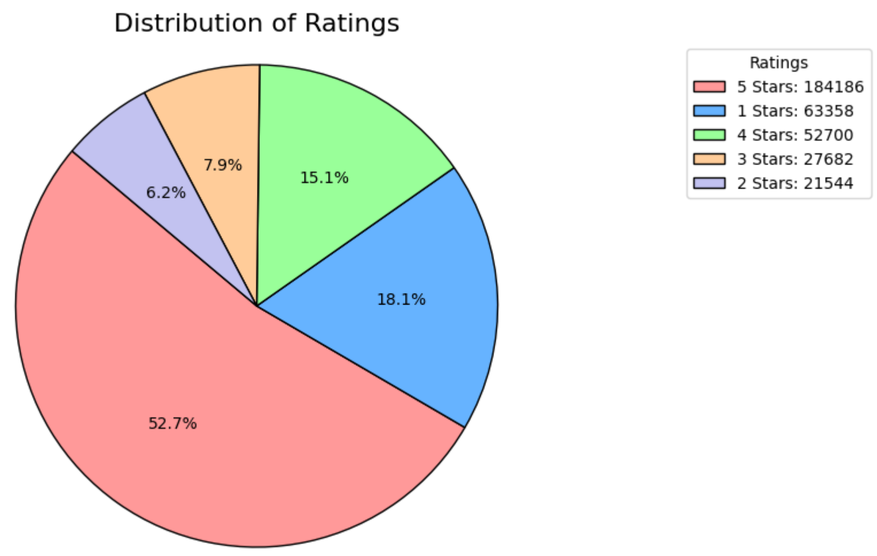
```
Unique Brand Names:
['Samsung' 'Nokia' nan 'HTC' 'Jethro' 'ePassion' 'Cedar Tree Technologies'
 'Indigi' 'Phone Baby' 'OtterBox' 'Lenovo' 'Huawei' 'JUNING' 'Elephone'
 'Plum' 'VKworld' 'Ulefone' 'Jiuhe' 'Acer' 'Aeku' 'AKUA' 'Alcatel' 'TCL'
 'LG' 'Amar' 'Amazon' 'Odysseus' 'Digital SNITCH' 'Star' 'Android' 'Yezz'
 'Apple' 'Certified Refurbished' 'Argom Tech' 'Asus' 'BlackBerry'
 'Motorola' 'Palm' 'Pantech' 'ZTE' 'AT&T' 'UnAssigned' 'Atoah'
 'WorryFree Gadgets' 'WorryFree' 'iDROID USA' 'Blackview' 'BLU' 'Various'
 'CT-Miami' 'BLUBOO' 'Iridium' 'AeroAntenna' 'BlueCosmo' 'GT Star'
 'Boost Mobile' 'Sanyo' 'Sony' 'CablesOnline' 'Casio' 'CAT' 'CBSKY'
 'CECTDIGI' 'CellAllure' 'Unnecto' 'CHSLING' 'CNPGD' 'Conquest'
 'Consumer Cellular' 'Junlan' 'Quality Technology' 'Crony' 'Crown' 'China'
 'Dell' 'Docooler' 'DIKOO' 'DOOGEE' 'KVD' 'Doro' 'DTECH' 'Dupad Story'
 'EasyN' 'ECOO' 'Emporia' 'FIGO' 'Maxwest' 'Futuretech' 'Wool' 'S7 active'
 'Evergreen Flag & Garden' 'Generic' 'Getnord' 'Glocalme' 'IPRO'
 'Goldengulf' 'Google' 'GOSO' 'Inmarsat' 'GreatCall' 'Jitterbug' 'GT STAR'
 'GX' 'H2O' 'Victor' 'Shenzhen Jinwanyi' 'HOMTOM' 'JHM' 'HP' 'VoiceStream'
 'Huadoo' 'XJKJ' 'LightInTheBox' 'winwinzonece' 'HyRich' 'px phone' 'iNew'
 'Spicy World' 'iVAPO' 'iRULU' 'JANIZZ' 'Jersa' 'Trait-Tech' 'JIAKE'
 'GrandElectronics' 'Jiayu' "John's" "John's Phone" 'Eachbid' 'Kata'
 'Katito' 'KINGZONE' 'Kivors' 'Kocaso' 'KOOTION' 'Kovee Tech' 'Kyocera'
 'LandRum' 'LBER' 'Leagoo' 'OTEDA Industrial Co., Limited' 'JINHAIHUAHUI'
 'WEIL TECHNOLOGY (HK) Limited' 'iPartsBuy' 'DK' 'LeEco' 'Wireless One'
 'LGIC' 'P710' 'Risio' 'GL' 'LGG' 'LifeProof' 'Sudroid' 'Lumen'
 'Mango Natural' 'MANN ZUG' 'Marozi International Ltd' 'Meizu' 'M-HORSE'
 'Microsoft' 'Fosler Corporation' 'Eson' 'Selected-Gadgets'
 'Sony Ericsson' 'Mophie' 'ShopTech4Less' 'Moxee Technologies' 'MOCREO'
 'KINGWELL' 'Neoix' 'Higoo' 'Sdeals' 'Sidekick' 'bbfone' 'Newsunshine'
 'Grade - A' 'Nuu Mobile' 'Ofeely' 'OLA Products' 'OnePlus' 'Orbic'
 'Oukitel' 'UMI' 'Otium' 'Defender' 'Ultrabox' 'OUKITEL' 'Padgene'
 'Sprint' 'Pandaoo' 'PCD' 'Tuscan Leveling System' 'Pharos' 'Plum Mobile'
 'Polaroid' 'Posh Mobile' 'AMM Global Enterprises' 'QJO' 'Que Products'
 'Quick-get' 'RCA' 'Rim' 'REACH' 'RomaCostume' 'SENTEL' 'RugGear'
 'Saliency' 'Luxury Phone' 'BlueSolar' 'Global Services' 'PowerMoxie'
 'Unknown' 'TOP' 'Vostrostone' 'Red' 'Seawolf Technologies' 'SGH-T199'
 'Sharp' 'SIAM' 'Blackphone' 'Silent Circle' 'SKY Devices' 'VEGA'
 'GEEKERA' 'JoyGeek' 'Smart&Cool' 'KROO' 'Yota' 'SHENZHEN SNOPOW' 'Social'
 'Unlocked Cell Phone' 'SOLE Mobile' 'Sonim' 'EL GUAPO' 'Stoga' 'TracFone'
 'HJ Wireless' 'Blue' 'Samsung/Straight Talk' 'PAE' 'Ling' 'LuMen'
 'Supersonic' 'SVP' 'MTM Trading LLC' 'Turbosight LLD' 'Tell'
 'The NoPhone' 'Thetford Marine' 'SyuanTech' 'THL' 'Tivax' 'T-Mobile'
 'Danger Inc.' 'LiteFuze' 'DreamsEden' 'ToShare Tech' 'TOTO' 'TSJYING'
 'TUMI' 'iFcane' 'Peek' 'Wmicro' 'UHAPPY' 'Purplelan' 'Tsing' 'WRTeam'
 'Coolpad' 'Cubot' 'Concox' 'KIKAR' 'NUU Mobile' 'OceanCross' 'UTStarcom'
 'Verizon' 'Vernee' 'verykool' 'VIP' 'Visual Land' 'Wogiz' 'Q'
 'Shenzhen Xin Sheng Shang' 'Xiaomi' 'SZ Wave' 'Dead Sea Secrets']
```

```
df.isnull().sum()
```

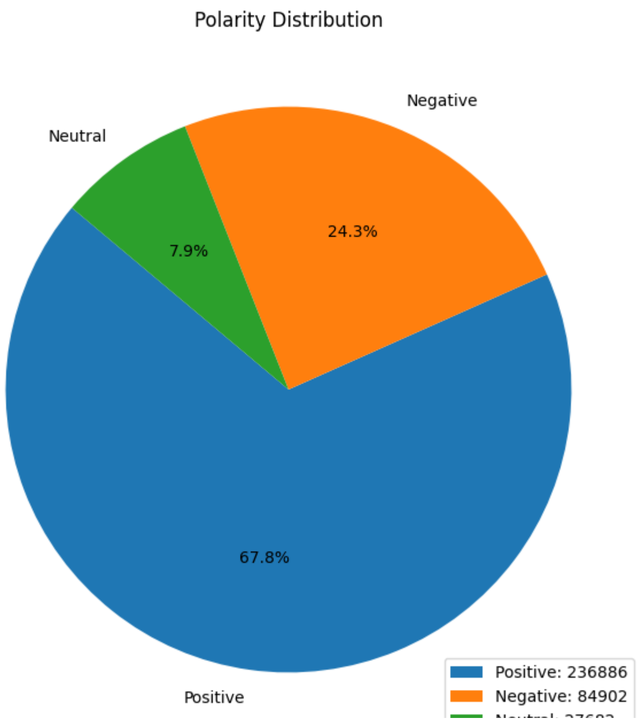
```
Product Name        0
Brand Name      54684
Price               0
Rating              0
Reviews             0
Review Votes        0
dtype: int64
```

```
import matplotlib.pyplot as plt
```

```
# Group the data by rating and count the number of occurrences
rating_counts = df['Rating'].value_counts()

# Define custom colors for the pie chart
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0']

# Create a pie chart
plt.figure(figsize=(10, 6))
plt.pie(rating_counts, labels=None, autopct='%1.1f%%', startangle=140, colors=colors, wedgeprops={'edgecolor': 'black'})
plt.title('Distribution of Ratings', fontsize=16)

# Display the total count of each rating as labels
rating_labels = [f'{rating} Stars: {count}' for rating, count in rating_counts.items()]
plt.legend(rating_labels, title="Ratings", loc='upper right', bbox_to_anchor=(1.2, 1))

# Display the pie chart
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```




```
# Function to categorize ratings
def categorize_rating(rating):
    if rating in [4, 5]:
        return "Positive"
    elif rating in [1, 2]:
        return "Negative"
    elif rating == 3:
        return "Neutral"

# Apply the categorize_rating function to create a new column 'Sentiment'
df['Sentiment'] = df['Rating'].apply(categorize_rating)

# Count the number of each sentiment category
sentiment_counts = df['Sentiment'].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Polarity Distribution')

# Add a legend with total counts
legend_labels = [f'{sentiment}: {count}' for sentiment, count in sentiment_counts.items()]
plt.legend(legend_labels, loc='lower right')
plt.show()
```




```
# Count the occurrences of each unique item in the 'Brand Name' column
brand_counts = df['Brand Name'].value_counts().head()

# Display the counts
print(brand_counts)
```

```
Samsung       58439
BLU           51780
Apple         50077
LG            22131
BlackBerry    14973
Name: Brand Name, dtype: int64
```

```
# Convert all text data to lowercase
df = df.applymap(lambda x: x.lower() if isinstance(x, str) else x)
```

```
df.head()
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
import spacy

# Download NLTK stopwords data if not already downloaded
nltk.download('stopwords')
nltk.download('punkt')

# Load spaCy's English language model
nlp = spacy.load("en_core_web_sm")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
def tokenize(text):
    tokens = word_tokenize(text.lower())
    return tokens

def remove_stopwords(tokens):
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word not in stop_words]
    return filtered_tokens

def remove_punctuation(tokens):
    table = str.maketrans('', '', string.punctuation)
    clean_tokens = [word.translate(table) for word in tokens]
    return clean_tokens

def lemmatize(tokens):
    doc = nlp(" ".join(tokens))
    lemmatized_tokens = [token.lemma_ for token in doc]
    return lemmatized_tokens
```

```
df['Tokenized'] = df['Reviews'].apply(tokenize)
df['Without_Stopwords'] = df['Tokenized'].apply(remove_stopwords)
df['Without_Punctuation'] = df['Without_Stopwords'].apply(remove_punctuation)
df['Lemmatized'] = df['Without_Punctuation'].apply(lemmatize)
df.head(10)
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stopwords	Without_Punctuation	Lemmatized
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive	[i, feel, so, lucky, to, have, found, this, us...	[feel, lucky, found, used, (, phone, us, &, us...	[feel, lucky, found, used, , phone, us, , used...	[feel, lucky, find, use, , phone, we, , use,...
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive	[nice, phone, „, nice, up, grade, from, my, pa...	[nice, phone, „, nice, grade, pantach, revue, ...	[nice, phone, , nice, grade, pantach, revue, ,...	[nice, phone, , nice, grade, pantach, revue, ...
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive	[very, pleased]	[pleased]	[pleased]	[please]
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive	[it, works, good, but, it, goes, slow, sometim...	[works, good, goes, slow, sometimes, good, pho...	[works, good, goes, slow, sometimes, good, pho...	[work, good, go, slow, sometimes, good, phone,...
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive	[great, phone, to, replace, my, lost, phone, ...	[great, phone, replace, lost, phone, ., thing,...	[great, phone, replace, lost, phone, , thing, ...	[great, phone, replace, lose, phone, , thing,...
5	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	1	i already had a phone with problems... i know ...	1.0	negative	[i, already, had, a, phone, with, problems, ...	[already, phone, problems, ..., know, stated, ...	[already, phone, problems, , know, stated, use...	[already, phone, problem, , know, state, use,...
6	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	2	the charging port was loose. i got that solder...	0.0	negative	[the, charging, port, was, loose, ., i, got, t...	[charging, port, loose, ., got, soldered, ., n...	[charging, port, loose, , got, soldered, , nee...	[charge, port, loose, , got, solder, , need,...
7	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	2	phone looks good but wouldn't stay charged, ha...	0.0	negative	[phone, looks, good, but, would, n't, stay, ch...	[phone, looks, good, would, n't, stay, charged...	[phone, looks, good, would, nt, stay, charged,...	[phone, look, good, would, not, stay, charge, ...
8	"clear clean esn" sprint				i originally was using the			[i, originally, was, using,	[originally, using, samsung, s2,	[originally, using, samsung, s2,	[originally, use, samsung,

```
# Define the file path where you want to save the CSV in Google Drive
file_path = "/content/drive/MyDrive/your_file.csv"

# Save the DataFrame as a CSV file
df.to_csv(file_path, index=False)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Split the dataset into training (60%) and temp (40%)
train_df, temp_df = train_test_split(df, test_size=0.4, random_state=42)

# Split the temp_df into validation (50%) and testing (50%)
validation_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42)
```

```
data2 = df.copy()
```

```
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the 'sentiment' column
data2['sentiment_encoded'] = label_encoder.fit_transform(data2['Sentiment'])

# Display the DataFrame with the encoded sentiment column
data2
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stopwords	Without_Punctuation	Lemmatized	sentiment_encoded
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive	[i, feel, so, lucky, to, have, found, this, us...	[feel, lucky, found, used, (, phone, us, &, us...	[feel, lucky, found, used, , phone, us, , used...	[feel, lucky, find, use, , phone, we, , use,...	2
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive	[nice, phone, „, nice, up, grade, from, my, pa...	[nice, phone, „, nice, grade, pantach, revue, ...	[nice, phone, , nice, grade, pantach, revue, ,...	[nice, phone, , nice, grade, pantach, revue, ...	2
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive	[very, pleased]	[pleased]	[pleased]	[please]	2
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive	[it, works, good, but, it, goes, slow, sometim...	[works, good, goes, slow, sometimes, good, pho...	[works, good, goes, slow, sometimes, good, pho...	[work, good, go, slow, sometimes, good, phone,...	2
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive	[great, phone, to, replace, my, lost, phone, ...	[great, phone, replace, lost, phone, ., thing,...	[great, phone, replace, lost, phone, , thing, ...	[great, phone, replace, lose, phone, , thing,...	2
...	...	...	...	...	...	...	...	...	...	...	...	...
413825	samsung convoy u640 phone for verizon wireless...	samsung	79.95	5	great phone. large keys, best flip phone i hav...	0.0	positive	[great, phone, ., large, keys, „, best, flip, ...	[great, phone, ., large, keys, „, best, flip, ...	[great, phone, , large, keys, , best, flip, ph...	[great, phone, , large, key, , good, flip, p...	2
413826	samsung convoy u640 phone for verizon wireless...	samsung	79.95	5	pros...works great, very durable, easy to navi...	0.0	positive	[pros, ..., works, great, „, very, durable, „,...	[pros, ..., works, great, „, durable, „, easy,...	[pros, , works, great, , durable, , easy, navi...	[pro, , work, great, , durable, , easy, nav...	2
	samsung convoy				just as described			[just, as, described,			[describe, perfect	

```
import numpy as np
print("Samp1es per class:{}".format(np.bincount(data2['sentiment_encoded'])))
```

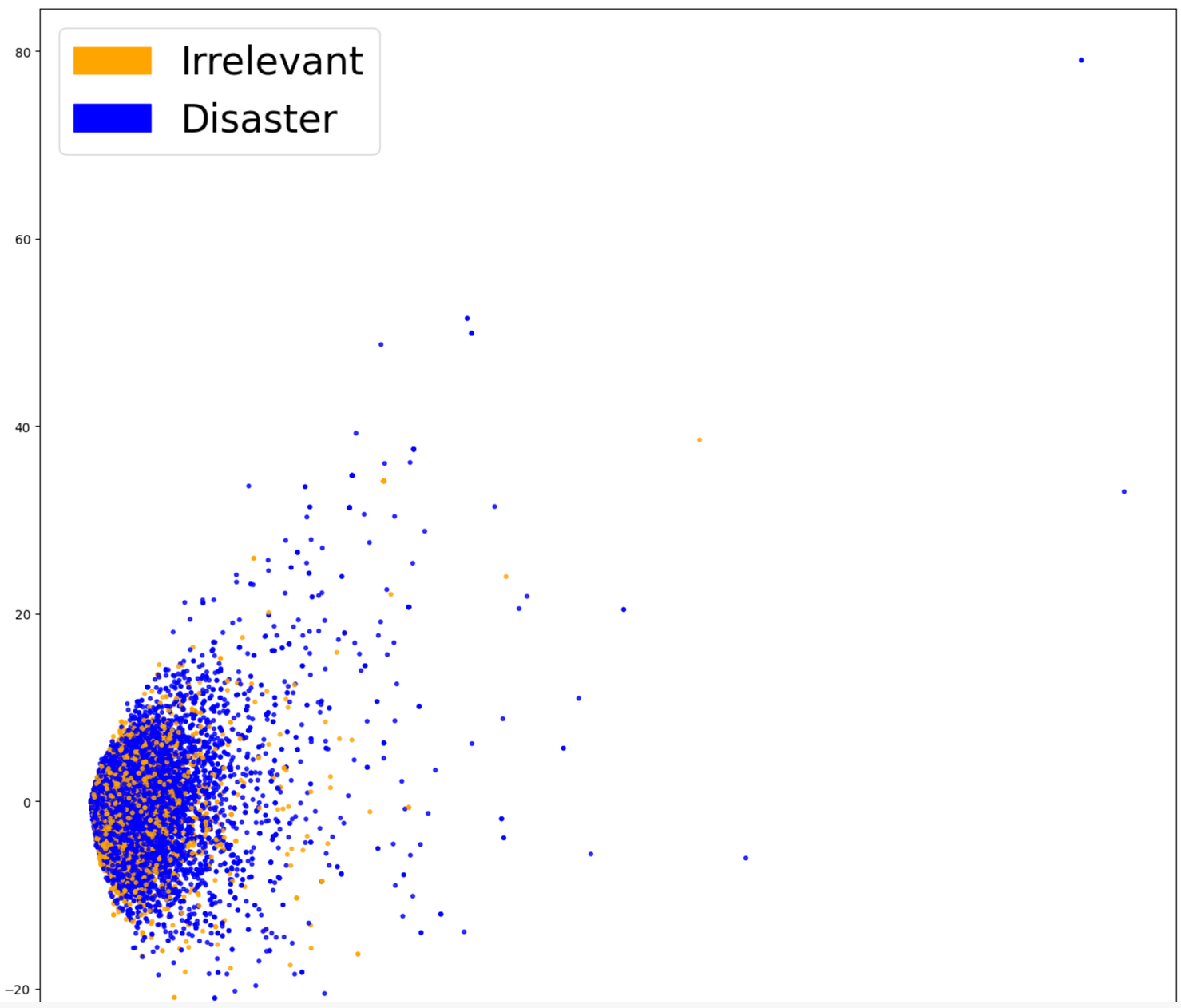
```
Samp1es per class:[ 84902  27682 236886]
```

```
# Inspecting our dataset a little more
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical

all_words = [word for Tokenized in data2["Tokenized"] for word in Tokenized]
sentence_lengths = [len(Tokenized) for Tokenized in data2["Tokenized"]]
VOCAB = sorted(list(set(all_words)))
print("%s words total, with a vocabulary size of %s" % (len(all_words), len(VOCAB)))
print("Max sentence length is %s" % max(sentence_lengths))
```

```
16743427 words total, with a vocabulary size of 124228
Max sentence length is 6201
```

```
# Bag of Words Counts
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

def cv(data):
    count_vectorizer = CountVectorizer()

    emb = count_vectorizer.fit_transform(data)

    return emb, count_vectorizer

list_corpus = data2["Reviews"].tolist()
list_labels = data2["sentiment_encoded"].tolist()

X_train, X_test, y_train, y_test = train_test_split(list_corpus, list_labels, test_size=0.2,
                                                    random_state=40)

X_train_counts, count_vectorizer = cv(X_train)
X_test_counts = count_vectorizer.transform(X_test)
```

```
from sklearn.decomposition import PCA, TruncatedSVD
import matplotlib
import matplotlib.patches as mpatches


def plot_LSA(test_data, test_labels, savepath="PCA_demo.csv", plot=True):
        lsa = TruncatedSVD(n_components=2)
        lsa.fit(test_data)
        lsa_scores = lsa.transform(test_data)
        color_mapper = {label:idx for idx,label in enumerate(set(test_labels))}
        color_column = [color_mapper[label] for label in test_labels]
        colors = ['orange','blue','blue']
        if plot:
            plt.scatter(lsa_scores[:,0], lsa_scores[:,1], s=8, alpha=.8, c=test_labels, cmap=matplotlib.colors.ListedColormap(colors))
            red_patch = mpatches.Patch(color='orange', label='Irrelevant')
            green_patch = mpatches.Patch(color='blue', label='Disaster')
            plt.legend(handles=[red_patch, green_patch], prop={'size': 5})


fig = plt.figure(figsize=(8,8))
plot_LSA(X_train_counts, y_train)
plt.show()
```




```
# Fitting a classifier
# Starting with a logistic regression is a good idea. It is simple, often gets the job done, and is easy to interpret.

from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(C=30.0, class_weight='balanced', solver='newton-cg',
                         multi_class='multinomial', n_jobs=-1, random_state=40)
clf.fit(X_train_counts, y_train)

y_predicted_counts = clf.predict(X_test_counts)
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report

def get_metrics(y_test, y_predicted):
    # true positives / (true positives+false positives)
    precision = precision_score(y_test, y_predicted, pos_label=None,
                                average='weighted')
    # true positives / (true positives + false negatives)
    recall = recall_score(y_test, y_predicted, pos_label=None,
                          average='weighted')

    # harmonic mean of precision and recall
    f1 = f1_score(y_test, y_predicted, pos_label=None, average='weighted')

    # true positives + true negatives/ total
    accuracy = accuracy_score(y_test, y_predicted)
    return accuracy, precision, recall, f1

accuracy, precision, recall, f1 = get_metrics(y_test, y_predicted_counts)
print("accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accuracy, precision, recall, f1))
```

```
accuracy = 0.872, precision = 0.895, recall = 0.872, f1 = 0.881
```

```
a = {'Metric': ['Accuracy', 'Precision', 'Recall', 'F1'],
        'Score': [accuracy, precision, recall, f1]}

result_cl = pd.DataFrame(a)
result_cl
```

	Metric	Score
0	Accuracy	0.871534
1	Precision	0.894847
2	Recall	0.871534
3	F1	0.880541

```
from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest Classifier
clf_rf = RandomForestClassifier(n_estimators=100, random_state=40, class_weight='balanced', n_jobs=-1)

# Fit the classifier to your training data
clf_rf.fit(X_train_counts, y_train)

# Predict on the test data
y_predicted_rf = clf_rf.predict(X_test_counts)

# Calculate metrics
accuracy_rf, precision_rf, recall_rf, f1_rf = get_metrics(y_test, y_predicted_rf)

print("Random Forest - accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accuracy_rf, precision_rf, recall_rf, f1_rf))
```

```
Random Forest - accuracy = 0.930, precision = 0.928, recall = 0.930, f1 = 0.926
```

```
_rf = {'Metric': ['Accuracy', 'Precision', 'Recall', 'F1'],
        'Score': [accuracy_rf, precision_rf, recall_rf, f1_rf]}

result_rf = pd.DataFrame(_rf)
result_rf
```

```
from sklearn.naive_bayes import MultinomialNB

# Create a Naive Bayes Classifier
clf_nb = MultinomialNB()
```

```
# Fit the classifier to your training data
clf_nb.fit(X_train_counts, y_train)

# Predict on the test data
y_predicted_nb = clf_nb.predict(X_test_counts)

# Calculate metrics
accuracy_nb, precision_nb, recall_nb, f1_nb = get_metrics(y_test, y_predicted_nb)

print("Naive Bayes - accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accuracy_nb, precision_nb, recall_nb, f1_nb))
```

```
_nb = {'Metric': ['Accuracy', 'Precision', 'Recall', 'F1'],
        'Score': [accuracy_nb, precision_nb, recall_nb, f1_nb]}

result_nb = pd.DataFrame(_nb)
result_nb
```

```
# Bi-RNN Classifier
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Bidirectional, LSTM, Dense, Embedding

# Create a Bi-RNN model
model_rnn = Sequential()
model_rnn.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_sequence_length))
model_rnn.add(Bidirectional(LSTM(units=64, return_sequences=True)))
model_rnn.add(Dense(num_classes, activation='softmax'))

# Compile the model
model_rnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model_rnn.fit(X_train_padded, y_train_encoded, epochs=num_epochs, batch_size=batch_size, validation_split=0.1)

# Evaluate the model on test data
accuracy_rnn, precision_rnn, recall_rnn, f1_rnn = model_rnn.evaluate(X_test_padded, y_test_encoded)

print("Bi-RNN - accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accuracy_rnn, precision_rnn, recall_rnn, f1_rnn))
```

```
_rnn = {'Metric': ['Accuracy', 'Precision', 'Recall', 'F1'],
        'Score': [accuracy_rnn, precision_rnn, recall_rnn, f1_rnn]}

result_rnn = pd.DataFrame(_rnn)
result_rnn
```

```

```