SENTIMENT ANALYSIS USING LOGISTIC REGRESSION

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

In [2]:
```
df = pd.read_csv('Reviews.csv')
df.head(10)
```

Out[2]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food
1	2	B001E3GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised
2	3	B000LQOCH0	ABELMWX0XAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all
3	4	B000LAOQIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wasp"	0	0	5	1350777600	Great taffy
5	6	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	0	0	4	1342051200	Nice Taffy
6	7	B006K2ZZ7K	A15PZKVAYXKRU1	David C. Sullivan	0	0	5	1340150400	Great just as good as the expensive brands!
7	8	B006K2ZZ7K	A3RGQVEQN31IQ	Pamela G. Williams	0	0	5	1336003200	Wonderful tasty taffy
8	9	B000E7L2R4	A1MZYO9TZK0BB0	R. James	1	1	5	1322006400	Yay Barley
9	10	B001711APVA	A21BT40VZCCYT4	Carol A. Reed	0	0	5	1351209600	Healthy Dog Food

REMOVING ALL THE NULL VALUES

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
Id                      0
ProductId               0
UserId                  0
ProfileName            16
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                27
Text                    0
dtype: int64
```

In [4]:
```
df = df.dropna()
```

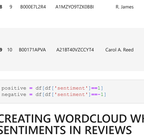
In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Id                      0
ProductId               0
UserId                  0
ProfileName             0
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                 0
Text                    0
dtype: int64
```

CREATING WORDCLOUD TO FIND MOST FREQUENTLY USED WORDS

In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
color = sns.color_palette()
%matplotlib inline
```

In [7]:
```
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud,STOPWORDS
stopwords = set(STOPWORDS)
stopwords.update(["br", "book"])
text = " " + join(review for review in df.Text)
wordcloud = WordCloud(stopwords=stopwords).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.savefig('wordcloud01.png')
plt.axis("off")
plt.show()
```



CLASSIFYING THE REVIEWS INTO "POS" AND "NEG" BASED ON "SCORE" & CREATING "SENTIMENT" COLUMN

In [8]:
```
df = df[df['Score'] != 3]
df['sentiment'] = df['Score'].apply(lambda rating : +1 if rating > 3 else -1)
df.head(10)
```

Out[8]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food
1	2	B001E3GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised
2	3	B000LQOCH0	ABELMWX0XAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all
3	4	B000LAOQIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wasp"	0	0	5	1350777600	Great taffy
5	6	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	0	0	4	1342051200	Nice Taffy
6	7	B006K2ZZ7K	A15PZKVAYXKRU1	David C. Sullivan	0	0	5	1340150400	Great just as good as the expensive brands!
7	8	B006K2ZZ7K	A3RGQVEQN31IQ	Pamela G. Williams	0	0	5	1336003200	Wonderful tasty taffy
8	9	B000E7L2R4	A1MZYO9TZK0BB0	R. James	1	1	5	1322006400	Yay Barley
9	10	B001711APVA	A21BT40VZCCYT4	Carol A. Reed	0	0	5	1351209600	Healthy Dog Food

In [9]:
```
positive = df[df['sentiment']==1]
negative = df[df['sentiment']==-1]
```

CREATING WORDCLOUD WHICH CONTAINS "POSITIVE" SENTIMENTS IN REVIEWS

In [10]:
```
stopwords = set(STOPWORDS)
stopwords.update(["br", "book"],"good","great")
pos = " " + join(review for review in positive.Summary)
wordcloud_pos = WordCloud(stopwords=stopwords).generate(pos)
plt.imshow(wordcloud_pos, interpolation='bilinear')
plt.axis("off")
plt.savefig("wordcloud02.png")
plt.show()
```

CREATING WORDCLOUD WHICH CONTAINS "NEGATIVE" SENTIMENTS IN REVIEWS

In [11]:
```
neg = " " + join(review for review in negative.Summary)
wordcloud_neg = WordCloud(stopwords=stopwords).generate(neg)
plt.imshow(wordcloud_neg, interpolation='bilinear')
plt.axis("off")
plt.savefig("wordcloud03.png")
plt.show()
```

In [12]:
```
df['sentiment'] = df['sentiment'].replace{-1 : 'negative'}
df['sentiment'] = df['sentiment'].replace{1 : 'positive'}
```

REMOVING ALL THE PUNCTUATION IN "SUMMARY" COLUMN TO MAKE USE OF IT FOR THE ALGORITHM

In [13]:
```
def remove_punctuation(text):
    final = "".joinfu for u in text if u not in ("?", ".", ",", ",", ":", ";", "(",")")
    return final
df['Text'] = df['Text'].apply(remove_punctuation)
df = df.dropna(subset=['Summary'])
df['Summary'] = df['Summary'].apply(remove_punctuation)
```

CREATING NEW DATAFRAME BY SPLITTING THE "SUMMARY" AND "SENTIMENT" COLUMNS IN ORIGINAL DATASET

In [14]:
```
df_updated = df[['Summary','sentiment']]
df_updated.head()
```

Out[14]:

	Summary	sentiment
0	Good Quality Dog Food	positive
1	Not as Advertised	negative
2	Delight says it all	positive
3	Cough Medicine	negative
4	Great taffy	positive

SPLITTING OUR DATASET INTO "TRAIN DATA" AND "TEST DATA" USING "RANDOM" SPLIT METHOD

In [15]:
```
import numpy as np
index = df.index
df['random_number'] = np.random.randn(len(index))
train = df[df['random_number'] <= 0.8]
test = df[df['random_number'] > 0.8]
```

CREATING "BAG OF WORDS" MODEL WHICH CONVERTS TEXT INTO BAG OF WORDS FOR THE ALGORITHM

In [16]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(token_pattern='\b\w+\b')
train_matrix = vectorizer.fit_transform(train['Summary'])
test_matrix = vectorizer.transform(test['Summary'])
```

BUILDING THE LOGISTIC REGRESSION ALGORITHM FOR OUR DATASET

In [17]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver='lbfgs', max_iter=1000)
```

In [18]:
```
X_train = train_matrix
X_test = test_matrix
y_train = train['sentiment']
y_test = test['sentiment']
lr.fit(X_train,y_train)
```

Out[18]:
```
LogisticRegression(max_iter=1000)
```

In [19]:
```
predictions = lr.predict(X_test)
```

TESTING OUR LOGISTIC REGRESSION MODEL TO FIND "ACCURACY", "PRECISION", "RECALL"

In [20]:
```
from sklearn.metrics import confusion_matrix,classification_report
new = np.asarray(y_test)
confusion_matrix(predictions,y_test)
```

Out[20]:
```
array([[11476,  2394],
       [ 5796, 91767]], dtype=int64)
```

In [21]:
```
print(classification_report(predictions,y_test))
```

Out[21]:
```
              precision    recall  f1-score   support

    negative       0.66      0.83      0.74     13870
    positive       0.97      0.94      0.96     97563

    accuracy                           0.93    111433
   macro avg       0.82      0.88      0.85    111433
weighted avg       0.94      0.93      0.93    111433
```