AMAZON FINE FOOD REVIEWS DATA INSIGHTS

A project report submitted to Prof. Guillaume Faddoul of
San Francisco State University
In partial fulfillment of
the requirements for
ISYS 812 Python for Data Analysis

Master of Science

In

Business Analytics

by

Shailesh Krishna

San Francisco, California

December 2019

Abstract

The report contains the data analysis done on the Amazon Fine Food Reviews dataset as part of ISYS 812 Python for Data Analysis project. This dataset consists of reviews of fine foods from amazon. The report is divided into different sections describing the various phases of data preparation and text analysis done by us to gather useful insights.

Table of Contents

Introduction

Dataset Description

A. Name of the dataset: Amazon Fine Food Reviews [1]

B. Description: This dataset consists of reviews of fine food from amazon. The dataset has information for a period of ten years up to October 2012. Reviews include product and user data, ratings, and a plain text review. It also includes reviews from all other Amazon categories [1].

C. Dataset Information:

1. Total number of rows and columns
    a. Rows ~ 568K
    b. Columns: 10
2. Column description:
    a. IdRow: Id
    b. ProductId: Unique identifier for the product
    c. UserId: Unique identifier for the user
    d. ProfileName: Profile name of the user
    e. HelpfulnessNumerator: Number of users who found the review helpful
    f. HelpfulnessDenominator: Number of users who indicated whether they found the review helpful or not.
    g. Score: Rating between 1 and 5
    h. Time: Timestamp for the review
    i. Summary: Summary of the review

j. Text: Text of the review

3. Missing values: The below table lists the number of missing values per column.

Column	Number of missing values
Id	0
ProductId	0
UserId	0
ProfileName	16
HelpfulnessNumerator	0
HelpfulnessDenominator	0
Score	0
Time	0
Summary	27
Text	0

Motivation

Most of the e-commerce companies provide us with an option to write a rating/review regarding the purchased product or a service used by us. These ratings/reviews help companies to optimize strategies to get more customers. On the other hand, they also help the customers to make a more informed decision regarding the purchase of a product or service.

Driving Question

Amazon is undoubtedly the biggest online marketplace at present. As per some estimates it sells more than 10 million products on its platform. 9 out of 10 consumers would check prices on Amazon before making their final purchase. In such a scenario online customer reviews prove to be more helpful than information provided by the vendors. Consumers can judge the quality of any product by reading numerous reviews by other consumers. This was the driving question behind our analysis. In addition to that we wanted to perform some basic exploratory analysis to understand the reviews better as well as analyze the constituents of a positive review.

Dataset Cleaning Process

Below mentioned are the details about the data cleaning process.

A. Timestamp Conversion

1. The timestamp present in the dataset is the Unix timestamp**,** which is not readable. Hence, we converted it into a more readable format.
2. Also, we added three columns - day, month and year in the dataset for further analysis.

```
# Function convert time
def convert_time(time):
    """
    Function to convert unixtime to readable date format.
    """
    return datetime.utcfromtimestamp(time).strftime('%Y-%m-%d')

reviews_dataset['ConvertedTime'] = reviews_dataset['Time'].apply(convert_time)

# displaying top records
reviews_dataset[['Time','ConvertedTime']].head()
```

8]:

	Time	ConvertedTime
0	1303862400	2011-04-27
1	1346976000	2012-09-07
2	1219017600	2008-08-18
3	1307923200	2011-06-13
4	1350777600	2012-10-21

In [19]:

```
# Adding columns year, month and day to the dataset.
reviews_dataset[['Year','Month','Day']] = reviews_dataset.ConvertedTime.str.split('-',expand=True)
reviews_dataset.head()
```

ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	ConvertedTime	Year	Month	Day
001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better.	2011-04-27	2011	04	27

B. Drop missing values

1. We calculated the count of values in each column. Based on the calculation, we saw that the column ‘ProfileName’ is missing 16 values and ‘Summary’ is missing 27 values.

2. A total of 43 missing values in the entire dataset of 568454 records is very miniscule. Hence, we dropped the missing values from the dataset.
3. This dataset without the missing values will be used for further analysis.

```
# Calculating total count for each column
print(reviews_dataset.count())
```

```
Id                      568454
ProductId               568454
UserId                  568454
ProfileName             568438
HelpfulnessNumerator    568454
HelpfulnessDenominator  568454
Score                   568454
Time                    568454
Summary                 568427
Text                    568454
dtype: int64
```

```
# Checking for null values in the dataset.
print("Are there null values in the reviews dataset?: {}".format(reviews_dataset.isnull().values.any()))
```

```
Are there null values in the reviews dataset?: True
```

```
# Counting missing values.
total_missing = total_rows - reviews_dataset.count()
print(total_missing)
```

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

```
# Dropping missing values from the dataset.
reviews_dataset = reviews_dataset.dropna(how="any")
```

```
# Verifying the counts
print("The number of missing values in the reviews dataset are: {}".format(np.count_nonzero(reviews_dataset.isnull())))
```

```
The number of missing values in the reviews dataset are: 0
```

C. Duplicate rows

1. We checked for the duplicate rows in the data using pandas “duplicated()” method. There were no duplicate rows in our dataset.

```
# Checking duplicate rows
duplicate_rows = reviews_dataset[reviews_dataset.duplicated()]
print("The total number of duplicate rows in the reviews dataset are: {}".format(duplicate_rows.shape[0]))
duplicate_rows.head()
```

```
The total number of duplicate rows in the reviews dataset are: 0
```

2]:

Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text

D. Checking whether each profile name corresponds to one user id

1. We tried to check if each profile name corresponds to a single user id or not.
2. Doing this would help us in filtering out additional spam/duplicate rows from the dataset.
3. Through our analysis, we found that there were multiple users with the same name. The number of unique user Ids is greater than unique profile names.

```
# Count unique profile names
print("Total unique profile names in the reviews dataset are: {}".format(len(reviews_dataset.ProfileName.unique())))
```

```
Total unique profile names in the reviews dataset are: 218413
```

```
# Count unique user ids
print("Total unique user ids in the reviews dataset are: {}".format(len(reviews_dataset.UserId.unique())))
```

```
Total unique user ids in the reviews dataset are: 256047
```

```
# Count unique product ids
print("Total unique product ids in the revies dataset are: {}".format(len(reviews_dataset.ProductId.unique())))
```

```
Total unique product ids in the revies dataset are: 74258
```

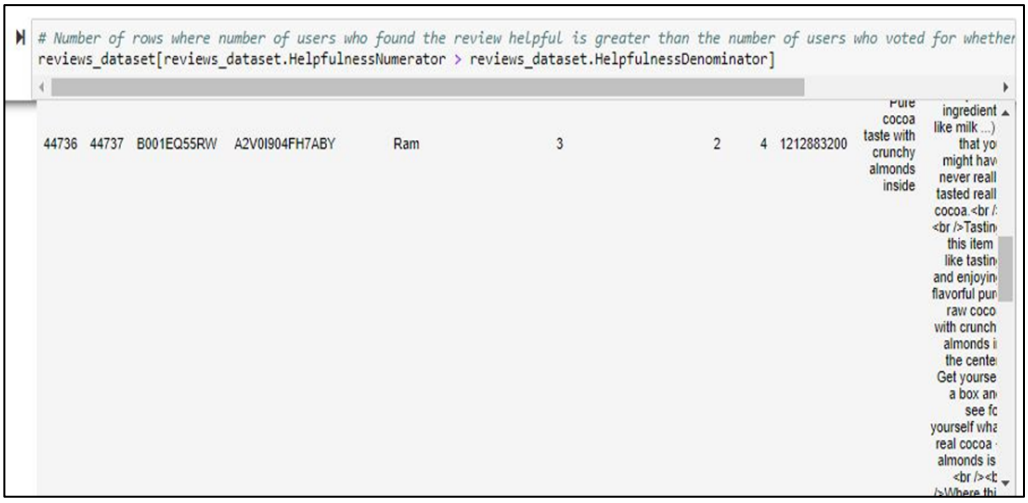
```
reviews_dataset[reviews_dataset.ProfileName == 'Karl'][["ProfileName","UserId"]]
```

7]:

	ProfileName	UserId
3	Karl	A395BORC6FGVXV
1622	Karl	A28FL0RSO744I8
14951	Karl	AO2EURP2Y4JZM
34393	Karl	A3ERW4BD2SMOTF
40645	Karl	A1XUMOH2FB2G6P
42806	Karl	AO2EURP2Y4JZM
133281	Karl	AO2EURP2Y4JZM
136303	Karl	A395BORC6FGVXV
141994	Karl	AO2EURP2Y4JZM
244813	Karl	A2K70G893JBLH8
256432	Karl	A2CU76B9ZDVLL3
258737	Karl	A2KDG54BLNOE2
315273	Karl	A11IJ39FJ33AGR
544172	Karl	A395BORC6FGVXV

E. HelpfulnessNumerator and HelpfulnessDenominator

1. HelpfulnessNumerator column indicates the number of users who found the review helpful.
2. HelpfulnessDenominator column indicates the number of users who found the review helpful in addition to the number of users who found it not helpful.
3. Ideally HelpfulnessNumerator should be less than Helpfulness Denominator. However, we found that it is not the case for few rows. Hence, we deleted such rows.



F. Spam Identification

1. The authenticity of a review drives many decisions. Fake reviews are a constant concern for companies.
2. A review is fake if it has been submitted multiple times by the same user, for the same product and at the same time. Generally, this is the case when a review is written by bots.
3. We tried to identify such fake reviews and filter out before carrying out data analysis.

```
check_multiple_records = reviews_dataset.groupby(["UserId","ProductId"]).size()
check_multiple_records[check_multiple_records > 1].head()
```

```
UserId          ProductId
A102TGNH1D915Z  B00008DFK5    2
                B0002DHNXC    2
                B0009YD7P2    2
                B000SP1CWW    2
A106N4B4EG4DR8  B0008IT4OM    2
dtype: int64
```

Here, we tried to identify fake/spam reviews written by bots. Using a couple of product and user ids from the above result to check if they have same reviews or not.

```
reviews_dataset[reviews_dataset.UserId.isin(["A29JUMRL1US6YP","A3TVZM3ZIXG8YW"]) & reviews_dataset.ProductId.isin(["B000084E
```

4. Based on the above analysis, we can see that for same UserId and ProductId we have multiple records with same reviews at the same time.
5. Irrespective of the product, the same user has rated multiple products at the same time with the same reviews.
6. We removed such fake reviews from our dataset.

```
# Preparing clean dataset.
clean_dataset = reviews_dataset.groupby(["UserId","Time"]).filter(lambda g: len(g) == 1)[["Id","ProductId","UserId","Profile
```

Final Tidy Dataset

1. In the above step we removed the fake reviews from the dataset and this new dataset which we have is the clean dataset that we are going to use for our data analysis.
2. For analysis, we will be creating various subsets of this dataset to answer other questions.

```
# Total number of rows in final dataset
total_rows_in_clean_dataset = clean_dataset.shape[0]
print("Total number of rows in clean dataset are: {}".format(total_rows_in_clean_dataset))
```

```
Total number of rows in clean dataset are: 275875
```

```
# Checking if we still have fake reviews left in the clean dataset
check_multiple_records_in_cleaned = clean_dataset.groupby(["UserId","Time"]).size()
check_multiple_records_in_cleaned[check_multiple_records_in_cleaned > 1].head()
```

```
Series([], dtype: int64)
```

First rows of our clean dataset

```
clean_dataset.head(10)
```

]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".
										This is a confection that has been around a few

Data Analysis

For the data analysis, we formulated five questions that we will try to answer through our analysis. The analysis for each question is mentioned below.

Q1. Which products are reviewed the most?

A. Data Preparation

1. We created a new dataframe containing two columns.
2. Column 1 contains the product id and column 2 contains the total review count per product id.

```
question1 = clean_dataset.groupby("ProductId").agg({"Text":"count"}).rename(columns={'Text': 'Review_Count'})
question1 = question1.reset_index()
question1.head()
```

	ProductId	Review_Count
0	0006641040	36
1	141278509X	1
2	2734888454	2
3	7800648702	2
4	9376674501	1

```
# Fetching top 20 products
question1_top20 = question1.sort_values(by = "Review_Count",ascending=False).head(20)
question1_top20.head()
```

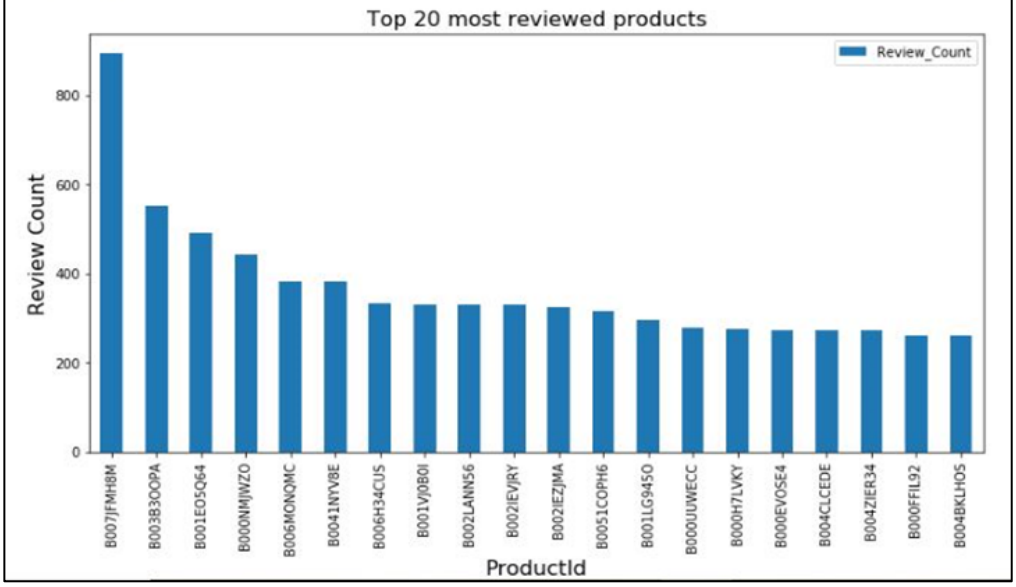
	ProductId	Review_Count
56389	B007JFMH8M	894
37010	B003B3OOPA	553
23004	B001EO5Q64	491
13658	B000NMJWZO	443
55007	B006MONQMC	382

B. Plotting

1. To answer this question, we plotted the chart of ProductId vs ReviewCount.
2. The chart displays top 20 most reviewed products.
3. We compared the review count for each product and derived the answer to the question.

4. **The most reviewed product is “Quaker Oats”**

```
question1_top20.plot(x="ProductId", y="Review_Count",kind="bar",figsize =(12,6))
plt.title("Top 20 most reviewed products", fontsize=16)
plt.ylabel('Review Count',fontsize=16)
plt.xlabel('ProductId',fontsize=16)
plt.show()
```



Q2. What is the review count trend over the years?

A. Data Preparation

1. We created a new dataframe containing two columns.
2. Column 1 contains year and column 2 contains the total review count per year.

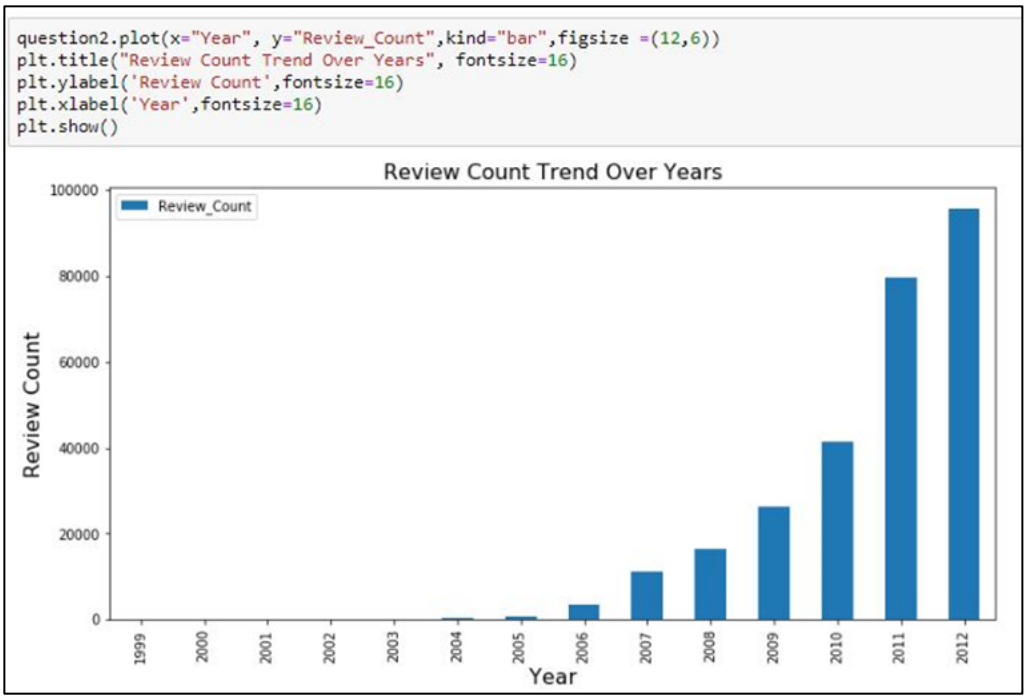
```
question2["Year"] = question2.Year.astype(np.int)
question2["Review_Count"] = question2.Review_Count.astype(np.int)
question2.head()
```

	Year	Review_Count
0	1999	3
1	2000	6
2	2001	5
3	2002	10
4	2003	69

B. Plotting

1. To answer this question, we plotted the chart of year vs review count.
2. **Here we can see that the since 2006, review counts have been increasing steadily. It indicates that the user engagement has been constantly increasing over the years.**

```
question2.plot(x="Year", y="Review_Count",kind="bar",figsize =(12,6))
plt.title("Review Count Trend Over Years", fontsize=16)
plt.ylabel('Review Count',fontsize=16)
plt.xlabel('Year',fontsize=16)
plt.show()
```



Q3. What are the monthly review trends?

A. Data Preparation

1. We created a dataframe containing two columns.
2. Column 1 contains month and column 2 contains total review count per month for all the years.

```
question3 = clean_dataset.groupby("Month").agg({"Text":"count"}).rename(columns={'Text': 'Review_Count'})
question3 = question3.reset_index()
question3.head()
```

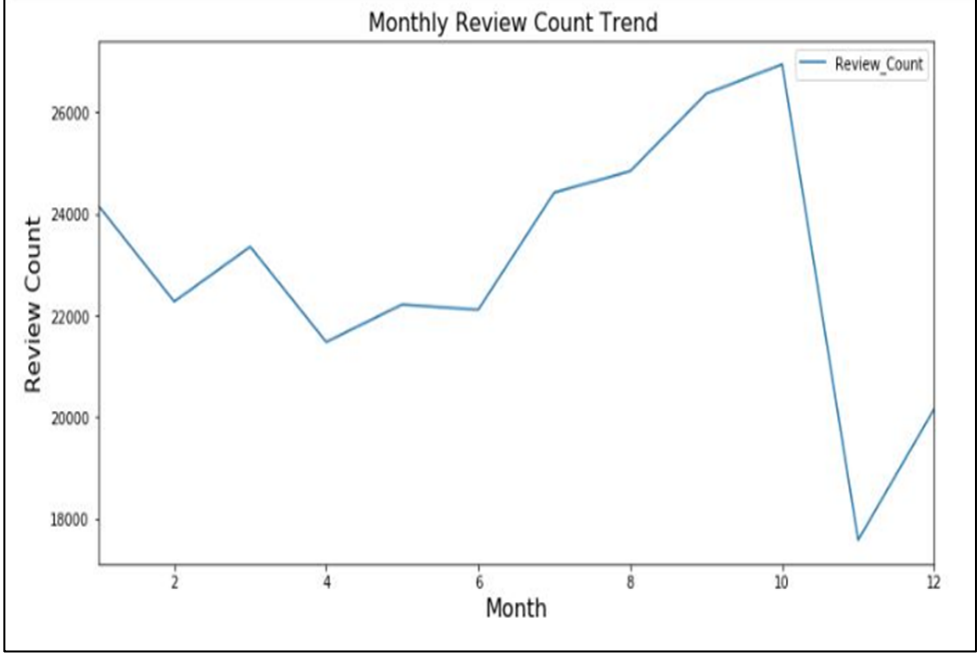
	Month	Review_Count
0	01	24161
1	02	22273
2	03	23349
3	04	21479
4	05	22216

```
question3["Month"] = question3.Month.astype(np.int)
question3["Review_Count"] = question3.Review_Count.astype(np.int)
```

B. Plotting

1. To answer this question, we plotted a chart of month vs total review count per month.
2. **From the chart we can see a sharp decline in the review count in the month of November whereas October recorded the maximum number of reviews.**

```
question3.plot(x="Month", y="Review_Count",kind="line",figsize =(12,6))
plt.title("Monthly Review Count Trend ", fontsize=16)
plt.ylabel('Review Count',fontsize=16)
plt.xlabel('Month',fontsize=16)
plt.show()
```



Q4. What are the three most important topics of a bad rating?

A. Data Preparation

1. We have considered bad rating because ratings influence people more that the good ratings. Also, bad ratings can have an adverse impact on the businesses as well.
2. For the analysis purpose, we categorized score 1 and 2 as bad ratings and will use this subset of data to perform analysis.
3. To prepare the dataframe, we will use score(ratings) to filter and summary text for topic modeling.

```
bad_ratings = clean_dataset[(clean_dataset.Score ==1)|(clean_dataset.Score ==2)]
bad_ratings.head()
```

12	13	B0009XLVG0	A327PCT23YH90	LT	1	1	1	1339545600	My Cats Are Not Fans of the New Food	more than two years. I just got a new bag and the shape of the food is different. They tried the new food when I first put it in their bowls and now the bowls sit full and the kitties will not touch the food. I've noticed similar reviews related to formula changes in

4. Preparing text corpus.

```
# Convert to list
summary_list = bad_ratings.Summary.values.tolist()

# Remove Emails
summary_list = [re.sub(r'\S*@\S*\s?', '', sent) for sent in summary_list]

# Remove new line characters
summary_list = [re.sub(r'\s+', ' ', sent) for sent in summary_list]

# Remove distracting single quotes
summary_list = [re.sub(r"\'", "", sent) for sent in summary_list]

print(summary_list[:1])
```

```
['Not as Advertised']
```

5. Converted sentences into words.

```
# Function sentence_to_words
def sentence_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations

summary_words = list(sentence_to_words(summary_list))
print(summary_words[:1])
```

```
[['not', 'as', 'advertised']]
```

6. Reduced the words to root words.

```
# Function lemmatization
def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']): #'NOUN', 'ADJ', 'VERB', 'ADV'
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append(" ".join([token.lemma_ if token.lemma_ not in ['-PRON-'] else '' for token in doc if token.pos_ in allowed_postags]))
    return texts_out

# Initialize spacy 'en' model, keeping only tagger component (for efficiency)

nlp = spacy.load('en', disable=['parser', 'ner'])

# Do lemmatization keeping only Noun, Adj, Verb, Adverb
summary_lemmatized = lemmatization(summary_words, allowed_postags=['NOUN', 'VERB']) #select noun and verb

print(summary_lemmatized[0:10])
```

```
['advertise', 'cat fan food', 'taste', 'flavor', 'like', 'product', 'taste', '', 'tea flavor', '']
```

7. Building a DTM (Document Term Matrix) in order to vectorize the inputs for LDA.

```
# Intializing vectorizer
vectorizer = CountVectorizer(analyzer='word',
                             min_df=10,                        # minimum reqd occurences of a word
                             stop_words='english',             # remove stop words
                             lowercase=True,                   # convert all words to lowercase
                             token_pattern='[a-zA-Z0-9]{3,}')  # num chars > 3
                             # max_features=50000,             # max number of uniq words
summary_vectorized = vectorizer.fit_transform(summary_lemmatized)
```

B. Data Modeling

1. Building LDA model

```
# Build LDA Model
lda_model = LatentDirichletAllocation(n_components=5,           # Number of topics
                                      max_iter=10,              # Max learning iterations
                                      learning_method='online',
                                      random_state=100,         # Random state
                                      batch_size=128,           # n docs in each learning iter
                                      evaluate_every = -1,      # compute perplexity every n iters, default: Don't
                                      n_jobs = -1,              # Use all available CPUs
                                     )
lda_output = lda_model.fit_transform(summary_vectorized)
print(lda_model)  # Model attributes
```

```
LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
                          evaluate_every=-1, learning_decay=0.7,
                          learning_method='online', learning_offset=10.0,
                          max_doc_update_iter=100, max_iter=10,
                          mean_change_tol=0.001, n_components=5, n_jobs=-1,
                          perp_tol=0.1, random_state=100, topic_word_prior=None,
                          total_samples=1000000.0, verbose=0)
```

2. Finding best number of topics for LDA

```
# Define Search Param
search_params = {'n_components': [5, 10], 'learning_decay': [.5, .7, .9]}
# Init the Model
lda = LatentDirichletAllocation(max_iter=5, learning_method='online', learning_offset=50.,random_state=0)
# Init Grid Search Class
model = GridSearchCV(lda, param_grid=search_params)
# Do the Grid Search
model.fit(summary_vectorized)
GridSearchCV(cv=None, error_score='raise',
       estimator=LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
             evaluate_every=-1, learning_decay=0.7, learning_method=None,
             learning_offset=10.0, max_doc_update_iter=100, max_iter=10,
             mean_change_tol=0.001, n_components=10, n_jobs=1, perp_tol=0.1, random_state=None,
             topic_word_prior=None, total_samples=1000000.0, verbose=0),iid=True, n_jobs=1,
       param_grid={'n_topics': [5, 10], 'learning_decay': [0.5, 0.7, 0.9]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
/Users/krishns18/opt/anaconda3/lib/python3.7/site-packages/sklearn/model_selection/_split.py:1978: FutureWar
n 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

```
GridSearchCV(cv=None, error_score='raise',
             estimator=LatentDirichletAllocation(batch_size=128,
                                                 doc_topic_prior=None,
                                                 evaluate_every=-1,
                                                 learning_decay=0.7,
                                                 learning_method=None,
                                                 learning_offset=10.0,
                                                 max_doc_update_iter=100,
                                                 max_iter=10,
                                                 mean_change_tol=0.001,
                                                 n_components=10, n_jobs=1,
                                                 perp_tol=0.1,
                                                 random_state=None,
                                                 topic_word_prior=None,
                                                 total_samples=1000000.0,
                                                 verbose=0),
             iid=True, n_jobs=1,
             param_grid={'learning_decay': [0.5, 0.7, 0.9],
                         'n_topics': [5, 10]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
```

3. Best Model for LDA

```
# Best Model
best_lda_model = model.best_estimator_
# Model Parameters
print("Best Model's Params: ", model.best_params_)
# Log Likelihood Score
print("Best Log Likelihood Score: ", model.best_score_)
# Perplexity
print("Model Perplexity: ", best_lda_model.perplexity(summary_vectorized))
```

```
Best Model's Params:  {'learning_decay': 0.9, 'n_components': 5}
Best Log Likelihood Score:  -99669.35744834488
Model Perplexity:  361.4697995137236
```

```
# column names
topicnames = ["Topic" + str(i) for i in range(best_lda_model.n_components)]
```

```
# Topic-Keyword Matrix
df_topic_keywords = pd.DataFrame(best_lda_model.components_)
# Assign Column and Index
df_topic_keywords.columns = vectorizer.get_feature_names()
df_topic_keywords.index = topicnames
# View
df_topic_keywords.head()
```

	acid	acquire	add	advertise	advertisement	advertising	aftertaste	agree	air	alcohol	alert	allerg
Topic0	0.202050	0.201744	77.252342	0.201935	27.587908	0.203124	0.203414	0.468638	0.203684	0.203816	0.202923	0.20601
Topic1	0.204540	0.201690	0.203529	0.203263	0.202057	0.202517	125.934836	47.819802	0.222330	0.203503	12.582363	0.20452
Topic2	25.174386	0.202042	0.255154	146.604525	0.202116	0.202611	0.211701	0.203262	12.090714	0.204495	0.202923	0.20249
Topic3	0.202532	0.201836	0.204978	0.202776	0.201856	0.201866	0.203542	0.207142	0.202364	0.201599	0.206368	0.20438
Topic4	0.205435	10.199585	0.250440	0.202677	0.203486	130.028940	0.204434	0.202408	0.207200	14.963298	0.207684	26.31199

4. Top N keywords for each topic

```
# Show top n keywords for each topic
def show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=20):
    keywords = np.array(vectorizer.get_feature_names())
    topic_keywords = []
    for topic_weights in lda_model.components_:
        top_keyword_locs = (-topic_weights).argsort()[:n_words]
        topic_keywords.append(keywords.take(top_keyword_locs))
    return topic_keywords

topic_keywords = show_topics(vectorizer=vectorizer, lda_model=best_lda_model, n_words=15)

# Topic - Keywords Dataframe
df_topic_keywords = pd.DataFrame(topic_keywords)
df_topic_keywords.columns = ['Word '+str(i) for i in range(df_topic_keywords.shape[1])]
df_topic_keywords.index = ['Topic '+str(i) for i in range(df_topic_keywords.shape[0])]
df_topic_keywords
```

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11	Word 12	Word 13	Word 14
Topic 0	dog	sugar	chocolate	ingredient	stuff	cup	box	salt	package	think	review	bar	mix	gluten	service
Topic 1	product	like	expect	cat	smell	love	overprice	change	bag	star	receive	shipping	time	disappointment	drink
Topic 2	flavor	coffee	price	tea	use	eat	item	packaging	water	date	break	cookie	advertise	bean	bottle
Topic 3	buy	make	money	quality	work	order	waste	food	rip	candy	look	want	arrive	ship	say
Topic 4	taste	beware	picture	contain	pack	try	oil	syrup	corn	purchase	advertising	disappoint	size	mislead	read

C. Interpretations and Insights

1. The most important topics of bad reviews are:
    a. Ingredient, quality and packaging related aspects of products.
    b. Overpriced products and shipping disappointments.
    c. Beware of misleading advertised product pictures, leading to shipping disappointments.
2. A few helpful insights are:
    a. Improve the quality and packaging of products.
    b. Scrutinize partners to reduce the product costs.
    c. Ensure advertisements meet customer expectations.

Q5.What are the most important characteristics of a helpful review?

For this question we will measure helpfulness of a review using Helpfulness Score which we are calculating using HelpfulnessNumerator and HelpfulnessDenominator given in the dataset. Following are the variable definitions given in the dataset:

HelpfulnessNumerator - Number of users who found the review helpful

HelpfulnessDenominator - Number of users who indicated whether they found the review helpful or not.

A. Data Preparation

The fundamental use of this business question is to find out whether we can find any meaningful characteristics which would make a review more helpful for other consumers. Most of our data is in the form of text. To answer the above question, we needed to quantify our data. We have used several Natural Language Processing libraries such as TextStat, Gensim and Sklearn to convert the available text into meaningful values.

The several steps in our data preparation process are as described below:

1. Calculating HelpfulnessScore

   Instead of using the existing variable, we have created our own response variable i.e HelpfulnessScore to determine the characteristics of a helpful review.

```
In [50]: #Removing HelpfulnessNumerator = 0
clean_dataset = clean_dataset[clean_dataset["HelpfulnessNumerator"] != 0]

#Removing HelpfulnessDenominator = 0
clean_dataset = clean_dataset[clean_dataset["HelpfulnessDenominator"] != 0]

#Calculating helpfulness Score using existing features
clean_dataset["HelpfulnessScore"] = clean_dataset["HelpfulnessNumerator"]/clean_dataset["HelpfulnessDenominator"]

clean_dataset.head()
```

UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore
UHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it	2011-04-27	2011	04	27	1.0

Transforming the Helpfulness Score into binary variable for classification. If helpfulness score > 0.5 then 1 else 0.

```
In [51]: def transform_helpfulness_score(HelpfulnessScore):
             """
             Function to return binary value 0,1 using HelpfulnessScore
             """
             if HelpfulnessScore >= 0.5:
                 return 1
             else:
                 return 0

         clean_dataset["Bin_HelpfulnessScore"] = clean_dataset.HelpfulnessScore.apply(transform_helpfulness_score)
         clean_dataset.head()
```

e	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore	Bin_HelpfulnessScore
n	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a	2011-04-27	2011	04	27	1.0	1

2. Calculate Word Count

WordCount column indicates the number of words present in a review. We created a function which returns the word count for the review text in every row.

Calculate Word Count

```
In [52]: def word_count(text):
             """
             Function to return word count of a text
             """
             return textstat.lexicon_count(text, removepunct=True)

         clean_dataset["WordCount"] = clean_dataset.Text.apply(word_count)
         clean_dataset.head(5)
```

essNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore	Bin_HelpfulnessScore	WordCount
1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it	2011-04-27	2011	04	27	1.0	1	48

3. Calculate Sentence Count

SentenceCount column indicates the number of sentences present in a review. We created a function which returns the sentence count for the review text in every row.

Calculate Sentence Count

```
In [53]: def sentence_count(text):
             """
             Function to return sentence count of a text
             """
             return textstat.sentence_count(text)

         clean_dataset["SentenceCount"] = clean_dataset.Text.apply(sentence_count)
         clean_dataset.head(5)
```

2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	a ligh pillowy citru gelatin wit nuts - in thi case Filbert And it is c into tin squares an then liberall coated wit powdere sugar. And is a tin mouthful heaven. N too chew and ver flavorful. highl

4. Calculate Sentiment Score

Sentiment Score/ polarity is a value which is returned based on the tone of a text. We created a function which returns the Sentiment Score for the review text in every row.

Calculate Sentiment Score

```
In [54]: def calculate_sentiment(text):
             """
             Function to return the sentiment polarity of a text
             """
             return TextBlob(text).sentiment.polarity

         # Lowercase Text
         clean_dataset["Text"] = clean_dataset["Text"].str.lower()
         clean_dataset['SentimentScore'] = clean_dataset.Text.apply(calculate_sentiment)
         clean_dataset.head()
```

ator	Score	Time	Summary	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore	Bin_HelpfulnessScore	WordCount	SentenceCount	SentimentScore
1	5	1303862400	Good Quality Dog Food	i have bought several of the vitality canned dog food products and have found them all to be of good quality. the product looks more like a stew than a processed meat and it smells better. my labrador	2011-04-27	2011	04	27	1.0	1	48	3	0.450000

5. Calculate Similarity Score

Similarity score represents the similarity of a text with the top 200 words of a helpful review. We calculated the similarity score for each text with the top 200 words of helpful reviews

   a. We filtered helpful scores using Bin_HelfulnessScore variable

   b. We built a Document-Term-Matrix(DTM)

   c. We determined the top 200 words

   d. We calculate Jaccard similarity score using the DTM for each review text

```
In [55]: #1. Filter helpful scores using HelfulnessScore_bin variable
helpful_reviews = clean_dataset[clean_dataset["Bin_HelpfulnessScore"] == 1]
helpful_reviews.head()
```

ominator	Score	Time	Summary	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore	Bin_HelpfulnessScore	WordCount	SentenceCount	SentimentSco
1	5	1303862400	Good Quality Dog Food	i have bought several of the vitality canned dog food products and have found them all to be of good quality. the product looks more like a stew than a processed meat and it	2011-04-27	2011	04	27	1.0	1	48	3	0.4500

```
#2. Bulding DTM
# Convert to list
text_list = helpful_reviews.Text.values.tolist()

# Remove Emails
text_list = [re.sub(r'\S*@\S*\s?', '', sent) for sent in text_list]
# Remove new line characters
text_list = [re.sub(r'\s+', ' ', sent) for sent in text_list]

# Remove distracting single quotes
text_list = [re.sub(r"\'", "", sent) for sent in text_list]
print(text_list[:1])

def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations

text_words = list(sent_to_words(text_list))
print(text_words[0:10])
```

```
['i have bought several of the vitality canned dog food products and have found them all to be of good quality. the product
looks more like a stew than a processed meat and it smells better. my labrador is finicky and she appreciates this product b
etter than most.']
[['have', 'bought', 'several', 'of', 'the', 'vitality', 'canned', 'dog', 'food', 'products', 'and', 'have', 'found', 'them',
'all', 'to', 'be', 'of', 'good', 'quality', 'the', 'product', 'looks', 'more', 'like', 'stew', 'than', 'processed', 'meat',
'and', 'it', 'smells', 'better', 'my', 'labrador', 'is', 'finicky', 'and', 'she', 'appreciates', 'this', 'product', 'bette
r', 'than', 'most'], ['this', 'is', 'confection', 'that', 'has', 'been', 'around', 'few', 'centuries', 'it', 'is', 'light',
'pillowy', 'citrus', 'gelatin', 'with', 'nuts', 'in', 'this', 'case', 'filberts', 'and', 'it', 'is', 'cut', 'into', 'tiny',
'squares', 'and', 'then', 'liberally', 'coated', 'with', 'powdered', 'sugar', 'and', 'it', 'is', 'tiny', 'mouthful', 'of',
'heaven', 'not', 'too', 'chewy', 'and', 'very', 'flavorful', 'highly', 'recommend', 'this', 'yummy', 'treat', 'if', 'you',
'are', 'familiar', 'with', 'the', 'story', 'of', 'lewis', 'the', 'lion', 'the', 'witch', 'and', 'the', 'wardrobe', 'this',
'is', 'the', 'treat', 'that', 'seduces', 'edmund', 'into', 'selling', 'out', 'his', 'brother', 'and', 'sisters', 'to', 'th
e', 'witch'], ['right', 'now', 'im', 'mostly', 'just', 'sprouting', 'this', 'so', 'my', 'cats', 'can', 'eat', 'the', 'gras
s', 'they', 'love', 'it', 'rotate', 'it', 'around', 'with', 'wheatgrass', 'and', 'rye', 'too'], ['one', 'of', 'my', 'boys',
'needed', 'to', 'lose', 'some', 'weight', 'and', 'the', 'other', 'didnt', 'put', 'this', 'food', 'on', 'the', 'floor', 'fo
r', 'the', 'chubby', 'guy', 'and', 'the', 'protein', 'rich', 'no', 'by', 'product', 'food', 'up', 'higher', 'where', 'only',
'my', 'skinny', 'boy', 'can', 'jump', 'the', 'higher', 'food', 'sits', 'going', 'stale', 'they', 'both', 'really', 'go', 'fo
r', 'this', 'food', 'and', 'my', 'chubby', 'boy', 'has', 'been', 'losing', 'about', 'an', 'ounce', 'week'], ['my', 'cats',
'have', 'been', 'happily', 'eating', 'felidae', 'platinum', 'for', 'more', 'than', 'two', 'years', 'just', 'got', 'new', 'ba
```

```
In [62]: def get_jacc_sim(text):
             """
             Function to return the jaccard similarity score between text review and Top 200 Helpful reviews
             """
             a = set(text.split())
             b = set(final_top_200)
             c = a.intersection(b)
             return float(len(c))/ (len(a) + len(b) - len(c))

         clean_dataset['JaccardSimScore'] = clean_dataset.Text.apply(get_jacc_sim)
         clean_dataset.head()
```

	Time	Summary	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore	Bin_HelpfulnessScore	WordCount	SentenceCount	SentimentScore	JaccardSimSco
	1303862400	Good Quality Dog Food	i have bought several of the vitality canned dog food products and have found them all to be of good quality. the product looks more like a stew than a processed meat and it smells better. my labrador	2011-04-27	2011	04	27	1.0	1	48	3	0.450000	0.02127

6. Calculate Readability Score

   Readability indicates how readable a text is. We created a function which returns the Readability Score for the review text in every row.

Calculate Readability Score

```
In [63]: def get_readability_score(text):
             """
             Function to return the readability score
             """
             return textstat.flesch_reading_ease(text)

         clean_dataset['ReadabilityScore'] = clean_dataset.Text.apply(get_readability_score)
         clean_dataset.head()
```

ıry	Text	ConvertedTime	Year	Month	Day	HelpfulnessScore	Bin_HelpfulnessScore	WordCount	SentenceCount	SentimentScore	JaccardSimScore	ReadabilityScor
od lity od	i have bought several of the vitality canned dog food products and have found them all to be of good quality. the product looks more like a stew than a processed meat and it smells better. my labrador	2011-04-27	2011	04	27	1.0	1	48	3	0.450000	0.021277	39.6

B. Data Modeling

Before performing regression, we checked the correlation among all predictors. We did not see any high positive correlation which could affect the analysis.

```
X = clean_dataset[["Score","WordCount","SentenceCount","SentimentScore","JaccardSimScore","ReadabilityScore"]]
Y = clean_dataset["Bin_HelpfulnessScore"]
```

```
# To find the correlation among
# the columns using pearson method
X.corr(method ='pearson')
```

]:

	Score	WordCount	SentenceCount	SentimentScore	JaccardSimScore	ReadabilityScore
Score	1.000000	-0.040401	-0.027599	0.439448	-0.002401	0.043188
WordCount	-0.040401	1.000000	0.575575	-0.177818	0.652688	-0.993449
SentenceCount	-0.027599	0.575575	1.000000	-0.102989	0.393726	-0.558507
SentimentScore	0.439448	-0.177818	-0.102989	1.000000	-0.140453	0.180414
JaccardSimScore	-0.002401	0.652688	0.393726	-0.140453	1.000000	-0.639533
ReadabilityScore	0.043188	-0.993449	-0.558507	0.180414	-0.639533	1.000000

We split the data into training and test sets

In [129]:
```
# Using the X and Y above
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.30, random_state=101)
X_train.head()
```

Out[129]:

	Score	SentimentScore	WordCount	SentenceCount	JaccardSimScore	ReadabilityScore
244806	1	0.487500	49	3	0.012876	38.66
68837	5	0.266667	14	1	0.009434	14.97
438382	4	0.413333	38	1	0.004329	58.29
200021	5	0.650000	33	1	0.008772	63.36
549950	5	-0.080000	45	5	0.021459	34.26

In [130]:
```
y_train.head()
```

Out[130]:
```
244806    0
68837     1
438382    1
200021    1
549950    1
Name: Bin_HelpfulnessScore, dtype: int64
```

In [131]:
```
print(len(X_train))
print(len(X_test))
```

```
93866
40229
```

In [132]:
```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
```

Performing logistic regression

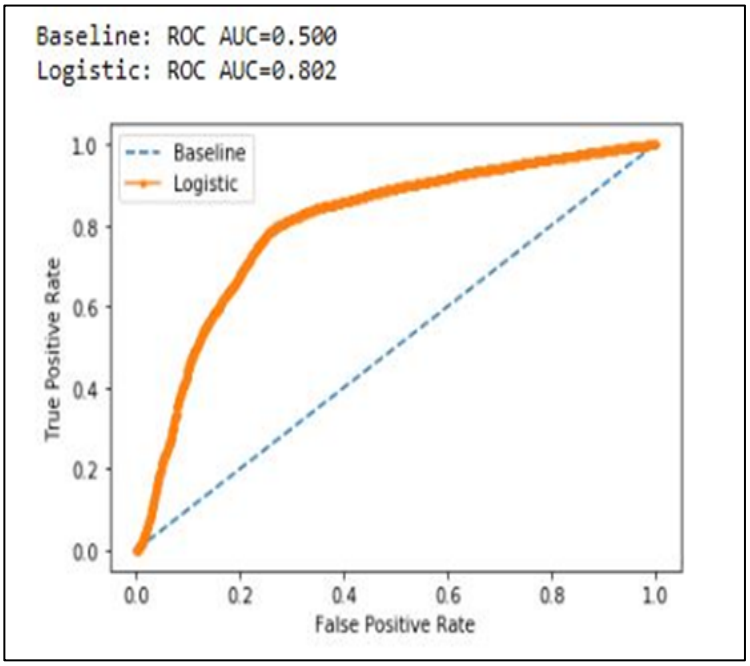
```
logit_model=sm.Logit(y_train,X_train)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.182059
         Iterations 8
                          Results: Logit
==================================================================
Model:              Logit                Pseudo R-squared: 0.173
Dependent Variable: Bin_HelpfulnessScore AIC:              34190.3455
Date:               2019-12-11 09:00     BIC:              34247.0432
No. Observations:   93866                Log-Likelihood:   -17089.
Df Model:           5                    LL-Null:          -20656.
Df Residuals:       93860                LLR p-value:      0.0000
Converged:          1.0000               Scale:            1.0000
No. Iterations:     8.0000
------------------------------------------------------------------
                   Coef.   Std.Err.     z     P>|z|    [0.025   0.975]
------------------------------------------------------------------
Score              0.7194    0.0102  70.3971  0.0000   0.6994   0.7394
WordCount          0.0024    0.0005   4.7471  0.0000   0.0014   0.0034
SentenceCount      0.0073    0.0057   1.2747  0.2024  -0.0039   0.0185
SentimentScore    -0.0328    0.0702  -0.4674  0.6402  -0.1705   0.1048
JaccardSimScore    8.7209    1.5841   5.5054  0.0000   5.6162  11.8256
ReadabilityScore   0.0022    0.0004   5.6110  0.0000   0.0014   0.0030
==================================================================
```

C. Interpretations

1. Rating of a review impacts the degree of helpfulness of a review.
2. Number of words in the review impacts the degree of helpfulness of a review
3. Surprisingly, sentence count and sentiment score doesn't have an impact on the helpfulness of a review.
4. Reviews containing more words which describe the product and packaging attributes and provide recommendation are considered more helpful
5. Readability score impacts the degree of helpfulness of a review.

To verify the performance of the model, we plotted the ROC curve. We see that the R-square is 80.2%, which is a good value.



Q6. What are the most important words in a helpful review?

To answer this question, we used the approach of plotting a Word Cloud for top 50 words.

```
import matplotlib.pyplot as pPlot
from wordcloud import WordCloud, STOPWORDS
import numpy as npy
from PIL import Image

text = str(final_top_200)
print(text)
wordcloud = WordCloud(width=1200, height=1000, min_font_size = 4, max_font_size = 200, font_step=1, max_words=100, backgrounc
plt.figure()
plt.imshow(wordcloud, interpolation ="bilinear")
plt.axis("off")
plt.show()
```

The word with the most occurrence is displayed the largest. From the wordcloud, we see that the most used words in helpful reviews are ‘Product’, ‘Flavor’, ‘Taste’ and so on. This indicates that the reviews are helpful if they provide information about the product such as taste, flavor, etc.

Conclusion

1. We found out that majority of the reviews were positive falling in the range of 4-5.
2. The most reviewed product was Quaker Soft Baked Oatmeal Cookie, which had close to 900 reviews, followed by Nature's Way Organic Extra Virgin Coconut Oil with 550 reviews.
3. The number of reviews from 99 to 2012 has consistently gone up which might be due to more people having access to internet or more people shopping on online platform, with October being the month with the greatest number of reviews.
4. Length of the review was corelated to the usefulness of the score. The more the number of words in the review the higher was the helpfulness score.
5. Ratings are considered to be the most helpful in a review, apart from how easy to read the review is. Easier to read and understand reviews are considered helpful. Lengthy reviews which go into detail about the product, its features are considered to be helpful.

Overall after doing this project we feel that we know a lot more about how reviews work, what constitute a helpful review and how it affects buying decisions of consumers. We had to be detail oriented during the course of the project especially during data cleaning process mainly because the outcome of out exploratory data analysis would depend a lot on well the data has been cleaned.

References

[1] Kaggle, https://www.kaggle.com/snap/amazon-fine-food-reviews

[2] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013.