Sentiment Analysis for Text using ML

Sentiment analysis using machine learning is a tool that analyses texts for polarity, from the text to either Positive or Negative or Neutral. By training machine learning tools with examples of emotions in text, machines automatically learn how to detect sentiment without human input.

The goal of sentiment analysis is to extract human emotions from text.

- **Problem Statement:**

Nowadays everything is getting digital, gone are those days where everyone used to go to shopping marts to buy products, now everything is just a click away. With the boom in internet companies, there is a high competition in the industry so to retain the customers, companies have the urge to analyse the feedback and evolve over time. With millions of customers, it is almost impossible to manually review the customer sentiment. That is where our problem is, we need to find the best fitting classifier which tells the sentiment of the customer based on their review of some product.
This is a text classification problem. This model predicts the sentiment of the customer from the text to either Positive or Negative or Neutral. It is expensive to check each review manually and label its sentiment. So, a better way is to rely on machine learning/deep learning models for that.

- **Rationale Statement:**

We use logistic regression on model as logistic regression works well for binary classification and for high dimensional sparse data which provides better insights on the customer sentiments

- **Data Acquisition:**

It is one of the key phases where you need to get the right data, your model is as good as your data. As simple as that.

In real-time we may have to deal with a lot of complexities to get the right data. You can do web scraping to get the data, but before going to that I would suggest going easy with existing datasets. Kaggle is a great place to start with, it has some decent datasets to work on. In this project, we will be working on Amazon Fine food reviews taken from Kaggle.

Data Source → https://www.kaggle.com/snap/amazon-fine-food-reviews

This dataset consists of ~500,000 reviews of fine foods from. The data span a period of more than 10 years. Reviews include several features like 'ProductId', 'UserId', 'Score' and 'text'.

Data includes:

- Reviews from Oct 1999 - Oct 2012
- Reviews from different Amazon categories.
- Number of users: 256,059
- Number of products: 74,258

- Number of reviews: 568,454
- Timespan: Oct 1999 - Oct 2012
- Number of Attributes/Columns in data: 10

Though there are many columns we will be considering just the review Text and their scores.

Though there are many columns we will be considering just the review Text and their scores.

The text contains the actual review and scores contain rating values like 1,2,3,4,5.

We could use Score/Rating feature to determine if a review is positive or negative.

Rating of 4 or 5 → Positive review

Rating of 1 or 2 → Considered as negative one +-

Rating of 3 → Neutral review

Then we are going to predict the sentiment from the review text

```
# Lets convert our sccores to three labels 0 -> negative(score<3), 1 -> Positive(score>3), 2 -> neutral(score==3)

def label(x):
    if x<3:
        return 0
    elif x>3:
        return 1
    elif x==3:
        return 2
```

- **Feature description:**

Here we can consider as - Text is our Feature and Score is our Label. So, we deal only with Text and Score columns

- ProductId (Categorical Variable) - Unique identifier for the product
- UserId (Categorical Variable) - Unique identifier for the user
- ProfileName (Text) - Profile of the user
- HelpfulnessNumerator (Numerical) - Number of users who found the review helpful
- HelpfulnessDenominator (Numerical) - Number of users who indicated whether they found the review helpful or not
- Score (Ordinal) - Rating between 1 and 5
- Time (Numerical) - Timestamp for the review
- Summary (Text)- Summary of the review
- Text (Text) - Text of the review

- ➢ **Sample Dataset:**

Id	ProductId	UserId	ProfileName	HelpfulnessNumerato	HelpfulnessDenominato	Score	Time	Summary	Text
1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1.304E+09	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found
2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1.347E+09	Not as Advertised	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually sn
3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres	1	1	4	1.219E+09	"Delight" says it all	This is a confection that has been around a few centuries. It is a light, pillowy c
4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1.308E+09	Cough Medicine	If you are looking for the secret ingredient in Robitussin I believe I have found i
5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassi	0	0	5	1.351E+09	Great taffy	Great taffy at a great price. There was a wide assortment of yummy taffy. Deli
6	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	0	0	4	1.342E+09	Nice Taffy	I got a wild hair for taffy and ordered this five pound bag. The taffy was all very
7	B006K2ZZ7K	A1SP2KVKFXXRU1	David C. Sullivan	0	0	5	1.34E+09	Great! Just as good as the expensive brands!	This saltwater taffy had great flavors and was very soft and chewy. Each candy
8	B006K2ZZ7K	A3JRGQVEQN31IQ	Pamela G. Williams	0	0	5	1.336E+09	Wonderful, tasty taffy	This taffy is so good. It is very soft and chewy. The flavors are amazing. I woul
9	B000E7L2R4	A1MZYO9TZK0BBI	R. James	1	1	5	1.322E+09	Yay Barley	Right now I'm mostly just sprouting this so my cats can eat the grass. They love
10	B00171APVA	A21BT40VZCCYT4	Carol A. Reed	0	0	5	1.351E+09	Healthy Dog Food	This is a very healthy dog food. Good for their digestion. Also good for small pu
11	B0001PB9FE	A3HDKO7OW0QNK4	Canadian Fan	1	1	5	1.108E+09	The Best Hot Sauce in the World	I don't know if it's the cactus or the tequila or just the unique combination of i
12	B0009XLVG0	A2725IB4YY9JEB	A Poeng "SparkyGoHome"	4	4	5	1.283E+09	My cats LOVE this "diet" food better than thei	One of my boys needed to lose some weight and the other didn't. I put this fo
13	B0009XLVG0	A327PCT23YH90	LT	1	1	1	1.34E+09	My Cats Are Not Fans of the New Food	My cats have been happily eating Felidae Platinum for more than two years. I j
14	B001GVISJM	A18ECVX2RJ7HUE	willie "roadie"	2	2	4	1.289E+09	fresh and greasy!	good flavor! these came securely packed... they were fresh and delicious! i love
15	B001GVISJM	A2MUGFV2TDQ47K	Lynrie "Oh HELL no"	4	5	5	1.268E+09	Strawberry Twizzlers - Yummy	The Strawberry Twizzlers are my guilty pleasure - yummy. Six pounds will be ar
16	B001GVISJM	A1CZX3CP8IKQIJ	Brian A. Lee	4	5	5	1.262E+09	Lots of twizzlers, just what you expect.	My daughter loves twizzlers and this shipment of six pounds really hit the spot
17	B001GVISJM	A3KLWF6WQ5BNYO	Erica Neathery	0	0	2	1.348E+09	poor taste	I love eating them and they are good for watching TV and looking at movies! It
18	B001GVISJM	AFKW14U97Z6QO	Becca	0	0	5	1.345E+09	Love it!	I am very satisfied with my Twizzler purchase. I shared these with others and v
19	B001GVISJM	A2A9X58G2GTBLP	Wolfee1	0	0	5	1.325E+09	GREAT SWEET CANDY!	Twizzlers, Strawberry my childhood favorite candy, made in Lancaster Pennsylv
20	B001GVISJM	A3IV7CL2C13K2U	Greg	0	0	5	1.318E+09	Home delivered twizlers	Candy was delivered very fast and was purchased at a reasonable price. I was h
21	B001GVISJM	A1WO0KGLPR5PV6	mom2emma	0	0	5	1.313E+09	Always fresh	My husband is a Twizzlers addict. We've bought these many times from Amaz
22	B001GVISJM	AZOF9E17RGZH8	Tammy Anderson	0	0	5	1.309E+09	TWIZZLERS	I bought these for my husband who is currently overseas. He loves these, and a
23	B001GVISJM	ARYVQL4N737A1	Charles Brown	0	0	5	1.305E+09	Delicious product!	I can remember buying this candy as a kid and the quality hasn't dropped in all
24	B001GVISJM	AJ613OLZZUG7V	Mare's	0	0	5	1.304E+09	Twizzlers	I love this candy. After weight watchers I had to cut back but still have a cravin
25	B001GVISJM	A22P2J09NJ9HKE	S. Cabanaugh "jilly pepper"	0	0	5	1.295E+09	Please sell these in Mexico!!	I have lived out of the US for over 7 yrs now, and I so miss my Twizzlers!! Whe
26	B001GVISJM	A3FONPR03H3PJS	Deborah S. Linzer "Cat Lady"	0	0	5	1.288E+09	Twizzlers - Strawberry	Product received is as advertised. <a href="http://www.amazon.cor

We could use Score/Rating feature to determine if a review is positive or negative.

Rating of 4 or 5 → Positive review

Rating of 1 or 2 → Considered as negative one

Rating of 3 → Neutral review

- ➢ **Overview of the review and summary text:**

The review text contains a detailed description of the product from the user's perspective. Moreover, it also describes the overall sentiment of the user toward the product apart from the description of the product itself. Some of the examples of the review text are shown below:

1) "This is great stuff. Made some really tasty banana bread. Good quality and lowest price in town."

2) This coffee is great because it's all organic ingredients! No pesticides to worry about plus it tastes good, and you have the healing effects of Ganoderma.

3) These condiments are overpriced and terrible. The classic is disgustingly sweet. The spiced tastes like a bad spicy marinara sauce from a chain restaurant.

On the other hand, the summary text represents a user's sentiment in a very confined manner. The information is conveyed in few words in this case. Some of the examples of the summary text are shown below:

1) "Best deal ever!"

2) "Waste of money"

3) "Great beans!!!"

4) "Big disappointment"

➢ Methodology:

There are 5 major steps involved in the building a ML model for sentiment classification. This encapsulates the following steps:

1) Data Acquisition & Performing exploratory data analysis for generating the binary response variable

2) Performing various text pre-processing steps which are used to remove noisy terms

3) Modelling & tuning the data

4) Evaluating the Model

5) Deploying the model

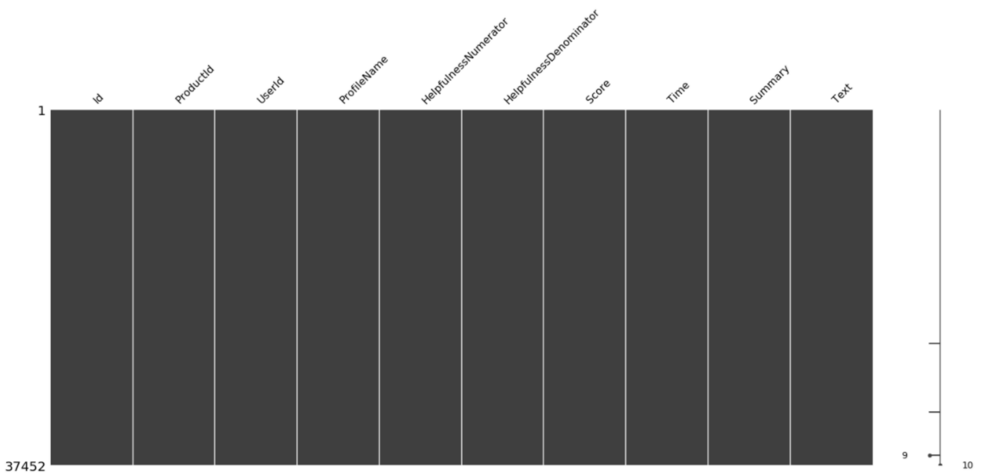
➢ Exploratory Data Analysis & Feature Engineering

Data Cleaning

1) Handling Null Values - We have no **Null values** in Text & Score columns as they are our Feature and Label respectively

```
In [10]: # Now lets check for the missing values.
         ## We can see that Score & Text columns do not have null values.
         ### Here we can consider as - Text is our Feature and Score is our Label. So from now on we deal only with Text and Score columns

         Food_Reviewsdf.isnull().any()

Out[10]: Id                        False
         ProductId                 False
         UserId                    False
         ProfileName                True
         HelpfulnessNumerator      False
         HelpfulnessDenominator    False
         Score                     False
         Time                      False
         Summary                    True
         Text                      False
         dtype: bool
```

```
In [12]: #We have no missing values.
         #To confirm, we can visualize missing values using missingno module.

         %matplotlib inline

         import missingno as msno
         msno.matrix(Food_Reviewsdf)

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x2430375b588>
```



2) Checking Data for **Duplicates** and dropping them. We could see that we have duplicate entries in our data.

```
In [8]: # Lets check the data for duplicate values

        Food_Reviews[Food_Reviews[["UserId","ProfileName","Time","Text"]].duplicated()]
```

Out[8]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
29	30	B0001PB9FY	A3HDKO7OW0QNK4	Canadian Fan	1	1	1	1107820800	The Best Hot Sauce in the World	I don't know if it's the cactus or the tequila...
574	575	B000G6RYNE	A3PJZ8TU8FDQ1K	Jared Castle	2	2	1	1231718400	One bite and you'll become a "chippoisseur"	I'm addicted to salty and tangy flavors, so wh...
1973	1974	B0017165OG	A2EPNS38TTLZYN	tedebear	0	0	2	1312675200	Pok Chops	The pork chops from Omaha Steaks were very tas...
2309	2310	B0001VWE0M	AQM74O8Z4FMS0	Sunshine	0	0	0	1127606400	Below standard	Too much of the white pith on this orange

3) So, we can drop the duplicate entries to make data clean and accurate.

```
In [9]: # So, there are duplicates in our data. Lets drop the duplicate entries.

        Food_Reviewsdf = Food_Reviews.drop_duplicates(subset={"UserId","ProfileName","Time","Text"},keep='first')
```

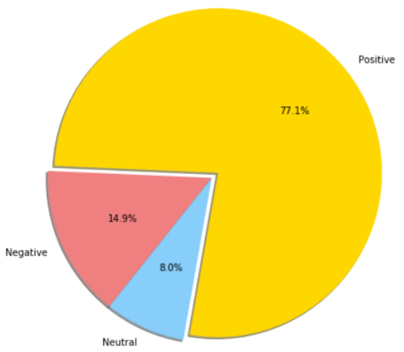
Let's understand our data first before we apply any modelling to it. It's a good practice to know what your data is trying to tell you.

1) From the below **pie chart and bar graph**, we can observe that 1->Positive class contributes almost 77.1% percent and 0 -> Negative class with 14.9% and 2->Neutral class with 8% of total labels. Clearly, we can say that this is an imbalanced data set.
   Our model has more data for positive reviews followed by negative. Neutral have the least data. This is one of the reasons for our model's poor performance.
   We deal the imbalanced dataset using Oversampling Methodology. **Oversampling** methods duplicate or create new synthetic examples in the minority class, we will be using one such oversampling technique called **Smote.**

Which class contributes to maximum and minimum percentage?

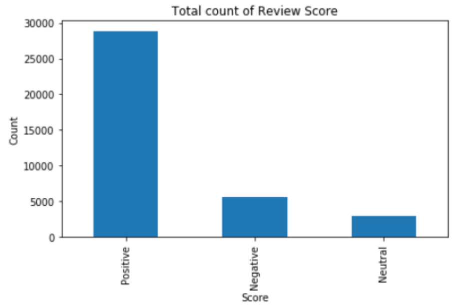
Pie Chart:

```
In [89]: # Pie Chart to

         labels = reviews_df["Score"].replace({0:'Negative',1:'Positive', 2:'Neutral'}).unique()
         colors = ['gold', 'lightcoral', 'lightskyblue']
         explode = (0.1, 0, 0)  # explode 1st slice

         # Plot
         plt.pie(reviews_df["Score"].value_counts(),autopct='%1.1f%%',radius=2,  explode=explode,labels=labels,colors=colors, shadow=True,
         plt.show()
```



Bar Graph:

```
In [22]: Food_Reviewsdf["Score"].replace({0:'Negative',1:'Positive', 2:'Neutral'}).value_counts().plot(kind='bar',figsize=(7,4))
         plt.title("Total count of Review Score");
         plt.xlabel("Score");
         plt.ylabel("Count");
```



2) Using the **TF-IDF weight** methodology, we retrive the top 15 words. Weight is a statistical measure used to evaluate how important the word is to a document in a collection.

Top 15 Words:

```
from wordcloud import WordCloud
plt.figure(figsize=(10,8))
wc = WordCloud(background_color="black",max_font_size=150, random_state=42)
wc.generate(str(top15))
plt.imshow(wc, interpolation='bilinear')
plt.suptitle('Top 15 words', size=30, y=0.88,color="r")
plt.axis("off")
plt.savefig("top15_words.png")
plt.show()
```

Top 15 words

Resampling the minority class

➢ <u>**Pre-Processing & Featurization of Text**</u>

Pre - Processing

We must transfer the text data from human language to machine-readable format for further processing. We need to apply pre-processing method to remove unnecessary characters, words & transform words into numerical features that work with machine learning algorithms. We will be using the NLTK (Natural Language Toolkit) library here. The Pre – Processing includes:

i) **NLTK** — The Natural Language ToolKit is one of the best-known and most-used NLP libraries, useful for all sorts of tasks from t tokenization, stemming, tagging, parsing, and beyond

ii) **BeautifulSoup** — Library for extracting data from HTML and XML documents

iii) **Tokenization** - Tokenization is the process of splitting the given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens

iv) **Stemming** - Stemming is the process of getting the root form of a word. Stem or root is the part to which inflectional affixes (-ed, -ize, -de, -s, etc.) are added. The stem of a word is created by removing the prefix or suffix of a word. So, stemming a word may not result in actual words.

v) **Remove stop words** - “Stop words” are the most common words in a language like “the”, “a”, “on”, “is”, “all”. These words do not carry important meaning and are usually removed from texts. It is possible to remove stop words using Natural Language Toolkit (NLTK), a suite of libraries and programs for symbolic and statistical natural language processing.

vi) Removing unnecessary punctuation, tags

```
In [28]: def text_Preprocessing(reviews):
             pre_processed_reviews=[]
             for review in tqdm(reviews):
                 review= BeautifulSoup(review,'lxml').getText()#remove html tags
                 review=re.sub('[^A-Za-z]+',' ',review) #remove special chars
                 review=re.sub("n't","not",review)
                 review=word_tokenize(str(review.lower())) #tokenize the reviews into word tokens
                 review=' '.join(PorterStemmer().stem(word) for word in review if word not in stop_words)
                 pre_processed_reviews.append(review.strip())
             return pre_processed_reviews

In [29]: preprocessed_reviews=text_Preprocessing(reviews_df["Text"])

100%|██████████| 37452/37452 [01:09<00:00, 540.16it/s]

In [30]: # Here we  have a Numpy array consisting of a list of lists

         preprocessed_reviews[72]

Out[30]: 'buyer bewar pleas sweeten not everybodi maltitol alcohol sugar undigest bodi know short time consum one unsuspect mani not dig
est extrem intestin bloat cramp massiv amount ga person experi nausea diarrhea headach also experienc learn lesson hard way yea
r ago fell love sugar free chocol suzann sommer use sell thought found sugar free chocol nirvana first tast bliss short live te
rribl side effect maltitol kick discomfort unlik anyth ever felt blew like balloon pain abdomin cramp symptom pass unpleas thou
gh hard believ low calori sweeten could culprit symptom gone stop eat chocol hunch someth maltitol unfortun confirm year later
purchas delici sugar free popcorn local market tast amaz look label wonder could possibl make yummi new sugarfre treat tast goo
d heart sank follow littl asterisk next sugarfre sweeten bottom label read maltitol tini littl letter thank good eaten littl st
ill end side effect much shorter durat peopl use maltitol heart content other like bad reaction case like not maltitol'

In [31]: # We create a pandas dataframe from numpy array

         preprocessed_reviews = pd.DataFrame({'text':preprocessed_reviews,'sentiment':reviews_df.Score})

In [32]: preprocessed_reviews

Out[32]:
```

	text	sentiment
0	bought sever vital can dog food product found ...	1
1	product arriv label jumbo salt peanut peanut a...	0
2	confect around centuri light pillowi citru gel...	1
3	look secret ingredi robitussin believ found go...	0
4	great taffi great price wide assort yummi taff...	1
...	...	...

Featurization:

In text processing, words of the text represent discrete, categorical features. How do we encode such data in a way which is ready to be used by the algorithms? The mapping from textual data to real valued vectors is called feature extraction.

The most popular approach is using the **Term Frequency-Inverse Document Frequency (TF-IDF)** technique.

Term Frequency (TF) = (Number of times term t appears in a document)/(Number of terms in the document)

Inverse Document Frequency (IDF) = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a frequent word is likely to be low. Thus, having the effect of highlighting words that are distinct.

We calculate TF-IDF value of a term as = TF * IDF

```
In [35]: #Applying TF-IDF

tfidf_model=TfidfVectorizer(ngram_range=(1,2),min_df=10, max_features=10000)
tfidf_model.fit(reviews_train,sentiment_train)
reviews_train_tfidf=tfidf_model.transform(reviews_train)
reviews_test_tfidf=tfidf_model.transform(reviews_test)
```

```
In [36]: print(tfidf_model.get_feature_names())
```

```
['abil', 'abl', 'abl buy', 'abl drink', 'abl eat', 'abl enjoy', 'abl find', 'abl get', 'abl make', 'abl order', 'abl purcha
s', 'abl use', 'absolut', 'absolut best', 'absolut delici', 'absolut favorit', 'absolut love', 'absolut wonder', 'absorb', 'a
bsorb flavor', 'abund', 'acai', 'accept', 'access', 'accid', 'accident', 'accompani', 'accomplish', 'accord', 'accord direc
t', 'account', 'accur', 'accustom', 'acerola', 'ach', 'achiev', 'acid', 'acid coffe', 'acid not', 'acid reflux', 'acid tast',
'acquir', 'acquir tast', 'across', 'act', 'act like', 'action', 'activ', 'actual', 'actual eat', 'actual good', 'actual lik
e', 'actual look', 'actual prefer', 'actual tast', 'actual use', 'actual work', 'ad', 'ad bit', 'ad bonu', 'ad flavor', 'ad l
ittl', 'ad milk', 'ad salt', 'ad sugar', 'ad sweeten', 'ad water', 'adapt', 'add', 'add anyth', 'add bit', 'add cup', 'add ex
tra', 'add flavor', 'add hot', 'add littl', 'add lot', 'add milk', 'add much', 'add nice', 'add one', 'add salt', 'add suga
r', 'add water', 'addict', 'addit', 'address', 'adequ', 'adjust', 'admit', 'admittedli', 'adopt', 'ador', 'adult', 'adult do
g', 'advanc', 'advantag', 'adventur', 'advers', 'advertis', 'advic', 'advis', 'affect', 'afford', 'afford price', 'afraid',
'african', 'afternoon', 'afternoon snack', 'aftertast', 'aftertast not', 'afterward', 'agav', 'agav nectar', 'age', 'agent',
'aggress', 'aggress chewer', 'ago', 'ago not', 'agre', 'agre review', 'ahead', 'ahoy', 'aid', 'air', 'air popper', 'air tigh
t', 'airi', 'airtight', 'aisl', 'akin', 'al', 'al dent', 'ala', 'alcohol', 'ale', 'alert', 'alfredo', 'alik', 'aliv', 'aller
g', 'allerg reaction', 'allergen', 'allergi', 'allot', 'allow', 'almond', 'almond butter', 'almond milk', 'almost', 'almost a
lway', 'almost anyth', 'almost everi', 'almost everyth', 'almost good', 'almost immedi', 'almost imposs', 'almost like', 'alm
ost tast', 'almost year', 'alo', 'alon', 'along', 'alot', 'alreadi', 'alright', 'also', 'also ad', 'also add', 'also bought',
'also buy', 'also contain', 'also eat', 'also enjoy', 'also find', 'also found', 'also get', 'also give', 'also good', 'also
got', 'also great', 'also healthi', 'also help', 'also keep', 'also like', 'also love', 'also made', 'also make', 'also muc
h', 'also nice', 'also not', 'also notic', 'also one', 'also purchas', 'also quit', 'also realli', 'also recommend', 'also sa
```

```
In [37]: tfidf_df=pd.DataFrame(reviews_train_tfidf.toarray(),columns=tfidf_model.get_feature_names(),index=reviews_train.index)
tfidf_df
```

Out[37]:

	abil	abl	abl buy	abl drink	abl eat	abl enjoy	abl find	abl get	abl make	abl order	...	zing	zip	zip lock	zipfizz	ziploc	ziplock	ziplock bag	ziwipeak	zuke	zuke hip
18039	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3584	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12377	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27293	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
306	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

- **Over Sampling**

The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important.

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique or SMOTE for short.

Resampling the minority class

```
In [92]: from imblearn import over_sampling
         from imblearn.over_sampling import SMOTE

In [93]: sm = SMOTE(sampling_strategy='auto', random_state=7)

         oversampled_trainX, oversampled_trainY = sm.fit_resample(reviews_train_tfidf,sentiment_train)
         oversampled_trainY.value_counts()

Out[93]: 2    21664
         1    21664
         0    21664
         Name: sentiment, dtype: int64
```

- **How Does Sentiment Analysis with Machine Learning Work? - “No Free Lunch” theorem states that there is no one model that works best for every problem.**

There are several techniques and complex algorithms used to command and train machines to perform sentiment analysis. There are pros and cons to each. But, used together, they can provide exceptional results.

We use logistic regression method, SVM, Decision Tree & Native Bias for classifying our problem and measure accuracy. We evaluate the models and choose the best model among the above according to their accuracy score to deploy.

- **Deploy the model**

1) We will create a web application using Flask.
2) We will create a face for our application. For this, we will use simple HTML and CSS.