Dataset

This dataset is of products that has been sold from different e-commerce websites.

Given data is in txt format and '|' seperator is used but this seperator is present more than no of columns. i.e we have to take a different delimiter on our dataset.

so i opened this using excel and using '|' seperator and excel automatically put 'tab' seperator at the end of each column. and also add some 'Unnamed' columns for the data outside the present columns. and saved this file as .txt.

Importing pandas

To load the dataset and visualize it

In [1]:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Load csv file using read_csv.

In [2]:

```
dataframe= pd.read_csv('C:\\Users\\RAJ\\Desktop\\data_engineering.txt', sep='\t')
dataframe.head()
```

Out[2]:

	_id	name	price	website_id	sku	url	
0	5d0b8aca0db7220b86cb4035	Joules Top Dog Underwear Three Pack	{'offer_price': {'currency': 'GBP', 'value': 3...	5cff5e7fe40f4900046735fa	312838	www.next.co.uk/style/st355408#312838	'sub
1	5d0b8aca0db7220b86cb4036	Figleaves Cheetah Satin Pyjama Set	{'offer_price': {'currency': 'GBP', 'value': 2...	5cff5e7fe40f4900046735fa	319571	https://www.next.co.uk/style/st324987#319571	'fi 'sub
2	5d0b8aca0db7220b86cb4037	Nike Solid 4" Swim Short	{'offer_price': {'currency': 'GBP', 'value': 1...	5cff5e7fe40f4900046735fa	335026	https://www.next.co.uk/style/st400645#335026	'sub
3	5d0b8aca0db7220b86cb4038	Collection Luxe Orchid	{'offer_price': {'currency': 'GBP', 'value': 1...	5cff5e7fe40f4900046735fa	552266	https://www.next.co.uk/style/st262195#552266	'sub
4	5d0b8aca0db7220b86cb4039	River Island White Sleeveless Blazer	{'offer_price': {'currency': 'GBP', 'value': 5...	5cff5e7fe40f4900046735fa	680971	https://www.next.co.uk/style/st440132#680971	'sub

In [3]:

```
dataframe.columns                          # to print all columns present in our data
```

Out[3]:

```
Index(['_id', 'name', 'price', 'website_id', 'sku', 'url', 'brand', 'media',
       'description_text', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11',
       'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15',
       'Unnamed: 16', 'Unnamed: 17'],
      dtype='object')
      dtype='object')
```

Here we can see that there are 9 columns with 'Unnamed: ' (9 to 17). when we scroll download the data we find that there is nothing in these columns so we can neglect this columns.

Make new dataframe

we will neglect the empty columns using 'iloc' function to access rows and columns of our data

In [4]:

```
new_dataframe=dataframe.iloc[:,0:9]
```

In [5]:

```
new_dataframe.head()
```

Out[5]:

	_id	name	price	website_id	sku	url	
0	5d0b8aca0db7220b86cb4035	Joules Top Dog Underwear Three Pack	{'offer_price': {'currency': 'GBP', 'value': 3...	5cff5e7fe40f4900046735fa	312838	www.next.co.uk/style/st355408#312838	'sub
1	5d0b8aca0db7220b86cb4036	Figleaves Cheetah Satin Pyjama Set	{'offer_price': {'currency': 'GBP', 'value': 2...	5cff5e7fe40f4900046735fa	319571	https://www.next.co.uk/style/st324987#319571	'fi 'sub
2	5d0b8aca0db7220b86cb4037	Nike Solid 4" Swim Short	{'offer_price': {'currency': 'GBP', 'value': 1...	5cff5e7fe40f4900046735fa	335026	https://www.next.co.uk/style/st400645#335026	'sub
3	5d0b8aca0db7220b86cb4038	Collection Luxe Orchid	{'offer_price': {'currency': 'GBP', 'value': 1...	5cff5e7fe40f4900046735fa	552266	https://www.next.co.uk/style/st262195#552266	'sub
4	5d0b8aca0db7220b86cb4039	River Island White Sleeveless Blazer	{'offer_price': {'currency': 'GBP', 'value': 5...	5cff5e7fe40f4900046735fa	680971	https://www.next.co.uk/style/st440132#680971	'sub

In [6]:

```
new_dataframe.columns
```

Out[6]:

```
Index(['_id', 'name', 'price', 'website_id', 'sku', 'url', 'brand', 'media',
       'description_text'],
      dtype='object')
```

Here only 9 rows available in new_dataframe that we needs.

In [7]:

```
new_dataframe['_id'].head(20)
```

Out[7]:

```
0                              5d0b8aca0db7220b86cb4035
1                              5d0b8aca0db7220b86cb4036
2                              5d0b8aca0db7220b86cb4037
3                              5d0b8aca0db7220b86cb4038
4                              5d0b8aca0db7220b86cb4039
5                              5d0b8aca0db7220b86cb403a
6                              5d0b8aca0db7220b86cb403b
7                              5d0b8aca0db7220b86cb403c
```

```
7                            5d0b8acb0db7220b86cb403c
8                            5d0b8acb0db7220b86cb403d
9                            5d0b8acb0db7220b86cb403e
10                           5d0b8acb0db7220b86cb403f
11                           5d0b8acb0db7220b86cb4040
12                           5d0b8acb0db7220b86cb4041
13                           5d0b8acb0db7220b86cb4042
14                           5d0b8acb0db7220b86cb4043
15                           5d0b8acb0db7220b86cb4044
16    Canvas upper with TOMS toe-stitch, and elastic...
17    TOMS classic suede insole with cushion for com...
18                   Latex arch insert for added support
19     One-piece outsole for flexibility and durability
Name: _id, dtype: object
```

Here we find that in id column there are some wrong id's available so first we remove those id's on the basis of length of valid 'id'.
we take first id as our reference id. (valid id)
we fill 'NaN' value present in id column with '0'. Because we removing invalid 'id's with length.

In [8]:

```
new_dataframe['_id'] = new_dataframe['_id'].fillna('0')
```

Removing empty ROWS and invalid Rows

In [9]:

```
for i in range(0, len(new_dataframe)):
    if len(new_dataframe['_id'][i]) > len(new_dataframe['_id'][0]) or len(new_dataframe['_id'][i])
== 1:
        new_dataframe.drop(i, inplace = True)
```

Reseting index

In [10]:

```
new_dataframe.reset_index(drop=True, inplace=True)
```

Price data

In [11]:

```
prices=new_dataframe['price'].fillna('0')       # fill NaN value with 0's for easily removal
```

In [12]:

```
len(prices)
```

Out[12]:

```
321720
```

Removing rows having 'Empty' price data

In [13]:

```
for j in range(0,len(prices)):
    entry=prices[j]
    if len(entry) ==1:
        new_dataframe.drop(j, inplace=True)
```

In [14]:

```
new_dataframe.reset_index(drop=True, inplace= True)     # reset index again
```

Removing rows having offer price > regular_price and price value= None

In [15]:

```
import re
for i in range(0,len(new_dataframe)):

    if re.findall("[+-]?\d+\.\d+",new_dataframe['price'][i]) == []:
        new_dataframe.drop(i, inplace=True)
    else:

        a=re.findall("[+-]?\d+\.\d+", new_dataframe['price'][i])
        offer_price=float(a[0])
        regular_price=float(a[1])
        if offer_price > regular_price:
            new_dataframe.drop(i, inplace=True)
```

In [16]:

```
new_dataframe.reset_index(drop=True, inplace=True)
```

In [17]:

```
len(new_dataframe)
```

Out[17]:

```
319125
```

Removing invalid 'Urls'

In [18]:

```
urls=new_dataframe['url']         # extract urls from data frame
urls.head()
```

Out[18]:

```
0             www.next.co.uk/style/st355408#312838
1    https://www.next.co.uk/style/st324987#319571
2    https://www.next.co.uk/style/st400645#335026
3    https://www.next.co.uk/style/st262195#552266
4    https://www.next.co.uk/style/st440132#680971
Name: url, dtype: object
```

In [19]:

```
from validator_collection import validators, checkers

for index in range(0,len(urls)):
    url=urls[index]
    if checkers.is_url(url) == False :
        new_dataframe.drop(index, inplace= True)
```

In [20]:

```
new_dataframe.reset_index(drop= True, inplace= True)      # Reseting the index of dataframe again
```

In [21]:

```
new_dataframe.head()
```

Out[21]:

Out[21]:

	_id	name	price	website_id	sku	url	
0	5d0b8aca0db7220b86cb4036	Figleaves Cheetah Satin Pyjama Set	{'offer_price': {'currency': 'GBP', 'value': 2...	5cff5e7fe40f4900046735fa	319571	https://www.next.co.uk/style/st324987#319571	'fi 'sub
1	5d0b8aca0db7220b86cb4037	Nike Solid 4" Swim Short	{'offer_price': {'currency': 'GBP', 'value': 1...	5cff5e7fe40f4900046735fa	335026	https://www.next.co.uk/style/st400645#335026	'sub
2	5d0b8aca0db7220b86cb4038	Collection Luxe Orchid	{'offer_price': {'currency': 'GBP', 'value': 1...	5cff5e7fe40f4900046735fa	552266	https://www.next.co.uk/style/st262195#552266	'sub
3	5d0b8aca0db7220b86cb4039	River Island White Sleeveless Blazer	{'offer_price': {'currency': 'GBP', 'value': 5...	5cff5e7fe40f4900046735fa	680971	https://www.next.co.uk/style/st440132#680971	'sub
4	5d0b8aca0db7220b86cb403a	Faith Animal Print Heel	{'offer_price': {'currency': 'GBP', 'value': 5...	5cff5e7fe40f4900046735fa	L07550	https://www.next.co.uk/style/esl07550#l07550	'sub

In [22]:

```
len(new_dataframe)
```

Out[22]:

```
319107
```

Analysing sentances in Text data

In [23]:

```
text_data=new_dataframe['description_text'].fillna('0')    # fill NAN value with 0's
```

In [24]:

```
text_data.head(10)
```

Out[24]:

```
0                                      100% Polyester.
1     Nike Swim Boys' Solid Lap 4 Volley Short is a ...
2                                          Height 85cm
3                                      100% Polyester.
4                  In a leopard print this tie up heel.
5                  70% Viscose, 26% Nylon, 4% Elastane.
6     Wish your loved one a Hoppy Birthday with this...
7                      With adjustable strap fastening.
8     18 carat gold-plated sterling silver. Comes in...
9                                 With blackout lining.
Name: description_text, dtype: object
```

Count Words present in text data

In [25]:

```
new_dataframe['word_count'] = new_dataframe['description_text'].apply(lambda x: len(str(x).split("
")))
```

In [26]:

```
new_dataframe[['description_text','word_count']].head(10)
```

Out[26]:

	description_text	word_count
0	100% Polyester.	2
1	Nike Swim Boys' Solid Lap 4 Volley Short is a ...	51
2	Height 85cm	3
3	100% Polyester.	2
4	In a leopard print this tie up heel.	8
5	70% Viscose, 26% Nylon, 4% Elastane.	6
6	Wish your loved one a Hoppy Birthday with this...	72
7	With adjustable strap fastening.	4
8	18 carat gold-plated sterling silver. Comes in...	10
9	With blackout lining.	3

Number of characters

In [27]:

```
new_dataframe['char_count'] = new_dataframe['description_text'].str.len() ## this also includes
spaces
new_dataframe[['description_text','char_count']].head()
```

Out[27]:

	description_text	char_count
0	100% Polyester.	15.0
1	Nike Swim Boys' Solid Lap 4 Volley Short is a ...	312.0
2	Height 85cm	12.0
3	100% Polyester.	15.0
4	In a leopard print this tie up heel.	36.0

In [28]:

```
new_dataframe['description_text'].head()
```

Out[28]:

```
0                                   100% Polyester.
1     Nike Swim Boys' Solid Lap 4 Volley Short is a ...
2                                       Height 85cm
3                                   100% Polyester.
4                  In a leopard print this tie up heel.
Name: description_text, dtype: object
```

Number of stopwords

In [31]:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')

new_dataframe['stopwords']=text_data.apply(lambda x: len([x for x in x.split() if x in stop]))
new_dataframe[['description_text','stopwords']].head()
```

Out[31]:

	description_text	stopwords

	description_text	stopwords
0	100% Polyester.	0
1	Nike Swim Boys' Solid Lap 4 Volley Short is a ...	14
2	Height 85cm	0
3	100% Polyester.	0
4	In a leopard print this tie up heel.	3

Text Pre-Processing

Lower case

In [32]:

```
text_data = text_data.apply(lambda x: " ".join(x.lower() for x in x.split()))
text_data.head()
```

Out[32]:

```
0                                      100% polyester.
1    nike swim boys' solid lap 4 volley short is a ...
2                                          height 85cm
3                                      100% polyester.
4                 in a leopard print this tie up heel.
Name: description_text, dtype: object
```

Removing punctuation

In [33]:

```
text_data = text_data.str.replace('[^\w\s]','')
text_data.head()
```

Out[33]:

```
0                                        100 polyester
1    nike swim boys solid lap 4 volley short is a d...
2                                          height 85cm
3                                        100 polyester
4                  in a leopard print this tie up heel
Name: description_text, dtype: object
```

Removing Stopwords

In [34]:

```
stop = stopwords.words('english')
text_data= text_data.apply(lambda x: " ".join(x for x in x.split() if x not in stop))
text_data.head()
```

Out[34]:

```
0                                        100 polyester
1    nike swim boys solid lap 4 volley short dual p...
2                                          height 85cm
3                                        100 polyester
4                               leopard print tie heel
Name: description_text, dtype: object
```

Removing Common Words

We will remove commonly occurring words from our text data First, let’s check the 10 most frequently occurring words in our text data then take call to remove or retain.

In [35]:

```
freq = pd.Series(' '.join(text_data).split()).value_counts()[:10]
freq
```

Out[35]:

```
featuring    102489
fit           94355
design        65225
fastening     61894
sleeves       56784
neck          55814
front         55519
black         48249
cotton        47787
logo          47024
dtype: int64
```

In [36]:

```
freq = list(freq.index)
text_data = text_data.apply(lambda x: " ".join(x for x in x.split() if x not in freq))
text_data.head()
```

Out[36]:

```
0                                        100 polyester
1    nike swim boys solid lap 4 volley short dual p...
2                                          height 85cm
3                                        100 polyester
4                               leopard print tie heel
Name: description_text, dtype: object
```

Removing Rare Words

In [37]:

```
freq = pd.Series(' '.join(text_data).split()).value_counts()[-10:]
freq = list(freq.index)
text_data = text_data.apply(lambda x: " ".join(x for x in x.split() if x not in freq))
text_data.head()
```

Out[37]:

```
0                                        100 polyester
1    nike swim boys solid lap 4 volley short dual p...
2                                          height 85cm
3                                        100 polyester
4                               leopard print tie heel
Name: description_text, dtype: object
```

spellings Correction

In [38]:

```
from textblob import TextBlob
text_data[:10].apply(lambda x: str(TextBlob(x).correct()))
```

Out[38]:

```
0                                        100 polyester
1    like swim boys solid lap 4 volley short dual p...
2                                            height cm
3                                        100 polyester
4                               leopard print tie heel
5                           70 dispose 26 non 4 latane
6    wish loved one happy birthday beer hawk birthd...
7                                     adjustable strap
8    18 cart goldplated sterling silver comes gift box
9                                      blackest lining
```

```
Name: description_text, dtype: object
```

Tokenization

In [39]:

```
import nltk
nltk.download('punkt')

TextBlob(text_data[1]).words
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\RAJ\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[39]:

```
WordList(['nike', 'swim', 'boys', 'solid', 'lap', '4', 'volley', 'short', 'dual', 'purpose', 'swim
ming', 'short', 'ideal', 'practice', 'laps', 'recreational', 'swimming', 'waterrepellent',
'fabric', 'soft', 'next', 'skin', 'stretch', 'waistband', 'inside', 'drawcord', 'builtin', 'mesh',
'brief', 'provide', 'support', 'great', 'dives', 'flips', 'cannonballs'])
```

Stemming

In [40]:

```
from nltk.stem import PorterStemmer
st = PorterStemmer()
text_data[:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

Out[40]:

```
0                                          100 polyest
1    nike swim boy solid lap 4 volley short dual pu...
2                                          height 85cm
3                                          100 polyest
4                               leopard print tie heel
Name: description_text, dtype: object
```

Lemmatization

In [41]:

```
from textblob import Word
import nltk
nltk.download('wordnet')

text_data = text_data.apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
text_data.head()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\RAJ\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[41]:

```
0                                        100 polyester
1    nike swim boy solid lap 4 volley short dual pu...
2                                          height 85cm
3                                        100 polyester
4                               leopard print tie heel
Name: description_text, dtype: object
```

Text Processing

N-Grams

In [42]:

```
TextBlob(text_data[0]).ngrams(2)
```

Out[42]:

```
[WordList(['100', 'polyester'])]
```

Term-Frequency

In [43]:

```
tf1 = (text_data[0:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis = 0).reset_index()
tf1.columns = ['words','tf']
tf1
```

Out[43]:

	words	tf
0	polyester	1.0
1	100	1.0
2	short	2.0
3	swimming	2.0
4	lap	2.0
5	practice	1.0
6	inside	1.0
7	ideal	1.0
8	solid	1.0
9	stretch	1.0
10	brief	1.0
11	soft	1.0
12	purpose	1.0
13	fabric	1.0
14	recreational	1.0
15	drawcord	1.0
16	cannonball	1.0
17	support	1.0
18	great	1.0
19	4	1.0
20	flip	1.0
21	swim	1.0
22	provide	1.0
23	nike	1.0
24	skin	1.0
25	waistband	1.0
26	volley	1.0
27	boy	1.0
28	mesh	1.0
29	dive	1.0
30	dual	1.0
31	next	1.0
32	waterrepellent	1.0
33	builtin	1.0

words tf

Inverse Document Frequency

In [44]:

```
import numpy as np
for i,word in enumerate(tf1['words']):
    tf1.loc[i, 'idf'] = np.log(text_data.shape[0]/(len(text_data[text_data.str.contains(word)])))

tf1
```

Out[44]:

	words	tf	idf
0	polyester	1.0	3.621703
1	100	1.0	2.940167
2	short	2.0	2.470134
3	swimming	2.0	7.093552
4	lap	2.0	3.607274
5	practice	1.0	8.048309
6	inside	1.0	4.662922
7	ideal	1.0	4.573424
8	solid	1.0	6.256549
9	stretch	1.0	3.176560
10	brief	1.0	4.839285
11	soft	1.0	3.129330
12	purpose	1.0	6.996528
13	fabric	1.0	2.904069
14	recreational	1.0	10.476057
15	drawcord	1.0	6.219657
16	cannonball	1.0	9.900693
17	support	1.0	4.513621
18	great	1.0	4.494643
19	4	1.0	3.166696
20	flip	1.0	5.951856
21	swim	1.0	4.458275
22	provide	1.0	4.613689
23	nike	1.0	4.370520
24	skin	1.0	3.064769
25	waistband	1.0	2.918874
26	volley	1.0	9.454406
27	boy	1.0	5.128950
28	mesh	1.0	3.988035
29	dive	1.0	6.178016
30	dual	1.0	5.606815
31	next	1.0	4.909835
32	waterrepellent	1.0	7.395167
33	builtin	1.0	6.795546

TF-IDF

In [45]:

```
tf1['tfidf'] = tf1['tf'] * tf1['idf']
tf1
```

Out[45]:

	words	tf	idf	tfidf
0	polyester	1.0	3.621703	3.621703
1	100	1.0	2.940167	2.940167
2	short	2.0	2.470134	4.940268
3	swimming	2.0	7.093552	14.187104
4	lap	2.0	3.607274	7.214547
5	practice	1.0	8.048309	8.048309
6	inside	1.0	4.662922	4.662922
7	ideal	1.0	4.573424	4.573424
8	solid	1.0	6.256549	6.256549
9	stretch	1.0	3.176560	3.176560
10	brief	1.0	4.839285	4.839285
11	soft	1.0	3.129330	3.129330
12	purpose	1.0	6.996528	6.996528
13	fabric	1.0	2.904069	2.904069
14	recreational	1.0	10.476057	10.476057
15	drawcord	1.0	6.219657	6.219657
16	cannonball	1.0	9.900693	9.900693
17	support	1.0	4.513621	4.513621
18	great	1.0	4.494643	4.494643
19	4	1.0	3.166696	3.166696
20	flip	1.0	5.951856	5.951856
21	swim	1.0	4.458275	4.458275
22	provide	1.0	4.613689	4.613689
23	nike	1.0	4.370520	4.370520
24	skin	1.0	3.064769	3.064769
25	waistband	1.0	2.918874	2.918874
26	volley	1.0	9.454406	9.454406
27	boy	1.0	5.128950	5.128950
28	mesh	1.0	3.988035	3.988035
29	dive	1.0	6.178016	6.178016
30	dual	1.0	5.606815	5.606815
31	next	1.0	4.909835	4.909835
32	waterrepellent	1.0	7.395167	7.395167
33	builtin	1.0	6.795546	6.795546

Bags of Words

In [46]:

```
from sklearn.feature_extraction.text import CountVectorizer
bow = CountVectorizer(max_features=1000, lowercase=True, ngram_range=(1,1),analyzer = "word")
train_bow = bow.fit_transform(text_data)
train_bow
```

Out[46]:

```
<319107x1000 sparse matrix of type '<class 'numpy.int64'>'
 with 4241394 stored elements in Compressed Sparse Row format>
```

Sentiment Analysis

In [47]:

```
text_data[:5].apply(lambda x: TextBlob(x).sentiment)
```

Out[47]:

```
0                       (0.0, 0.0)
1     (0.225, 0.39166666666666666)
2                       (0.0, 0.0)
3                       (0.0, 0.0)
4                       (0.0, 0.0)
Name: description_text, dtype: object
```

Above, we can see that it returns a tuple representing polarity and subjectivity of each tweet. Here, we only extract polarity as it indicates the sentiment as value nearer to 1 means a positive sentiment and values nearer to -1 means a negative sentiment

In [48]:

```
text_data[0:20].apply(lambda x: TextBlob(x).sentiment[0])
```

Out[48]:

```
0     0.000000
1     0.225000
2     0.000000
3     0.000000
4     0.000000
5     0.000000
6     0.300000
7     0.000000
8     0.000000
9     0.000000
10    0.250000
11    0.000000
12    0.227548
13    0.350000
14    0.000000
15    0.000000
16    0.000000
17    0.000000
18    0.128571
19    0.500000
Name: description_text, dtype: float64
```

In [ ]:

```

```