```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#pip install plotly


#pip install cufflinks


import plotly as py
import cufflinks as cf


from plotly.offline import iplot,plot,init_notebook_mode


py.offline.init_notebook_mode(connected=True)
cf.go_offline()
```

+ Code + Text

DATA IMPORT

```
df = pd.read_csv("Womens Clothing E-Commerce Reviews.csv",index_col=0)
df.head()
```

	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Departme Na
0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intima

```
df.drop(labels=['Title','Clothing ID'],axis=1,inplace=True)


df.head()
```

	Age	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	33	Absolutely wonderful - silky and sexy and comf	4	1	0	Initmates	Intimate	Intimates

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

```
Age                          0
Review Text                845
Rating                       0
Recommended IND              0
Positive Feedback Count      0
Division Name               14
Department Name             14
Class Name                  14
dtype: int64
```

```
df.dropna(subset=['Review Text','Division Name'],inplace=True)
```

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

```
Age                         0
Review Text                 0
Rating                      0
Recommended IND             0
Positive Feedback Count     0
Division Name               0
Department Name             0
Class Name                  0
dtype: int64
```

```
' '.join(df["Review Text"].tolist())
```

```
'Absolutely wonderful - silky and sexy and comfortable Love this dress!  it\'s sooo pre
tty.  i happened to find it in a store, and i\'m glad i did bc i never would have order
ed it online bc it\'s petite.  i bought a petite and am 5\'8".  i love the length on me
- hits just a little below the knee.  would definitely be a true midi on someone who is
truly petite. I had such high hopes for this dress and really wanted it to work for me.
i initially ordered the petite small (my usual size) but i found this to be outrageousl
y small. so small in fact that i could not zip it up! i reordered it in petite medium.
```

Text Cleaning

```
contractions = {
    "ain't":"am not",
    "aren't":"are not",
```

```
        "can't":"cannot",
        "can't've":"cannot have",
        "'cause":"because",
        "could've":"could have",
        "couldn't":"could not",
        "couldn't've":"could not have",
        "didn't":"did not",
        "don't":"do not",
        "doesn't":"does not",
        "hadn't":"had not",
        "hasn't":"has not",
        "he'd":"he would",
        "he'd've":"he would have",
        "he'll":"he will",
        "he'll've":"he will have",
        "he's":"he is",
        "how'd":"how did",
        "how'd'y":"how do you",
        "how'll":"how will",
        "how's":"how does",
        "i'll":"i will",
        "i'd":"i would",
        "i'd've":"i would have",
        "i'll":"i will",
        "i'll've":"i will have",
        "i'm":"i am",
        "i've":"i have",
        "isn't":"is not",
        "it'd":"it would",
        "it'd've":"it would have",
        "it'll":"it will",
        "it'll've":"it will have",
        "it's":"it is",
        "let's":"let us",
        "ma'am":"madam",
        "mayn't":"may not",
        "might've":"might have",
        "mightn't've":"might not have",
        "must've":"must not",
        "mustn't":"must not",
        "mustn't've":"must not have",
        "need't":"need not",
        "needn't've":"need not have",
        "o'clock":"of the clock",
        "oughtn't":"ought not",
        "oughtn't've":"ought not have",
        "shan't":"shall not",
        "sha'n't":"shall not",
        "shan't've":"shall not have",
        "she'd":"she would",
        "she'd've":"she would have",
```

```
    "she'll":"she will",
    "she'll've":"she will have",
    "she's":"she is",
    "should've":"should have",
    "shouldn't":"should not",
    "shouldn't've":"should not have",
    "so've":"so have",
    "so's":"so is",
    "that's":"that is",
    "that'd":"that would",
    "that'd've":"that would have",
    "there'd":"there would",
    "there'd've":"there would have",
    "there's":"there is",
    "they'd":"they would",
    "they'd've":"they would have",
    "they'll":"they will",
    "they'll've":"they will have",
    "they're":"they are",
    "they've":"they have",
    "to've":"to have",
    "wasn't":"was not",
    " u ":" you ",
    " ur ":" your ",
    " n ":" and "
}

def cont_to_exp(x):
  if type(x) is str:
    x = x.replace('\\','')
    for key in contractions:
      value = contractions[key]
      x = x.replace(key,value)
    return x
  else:
    return x



x = "i don't know what it's going to cost. You should've told me the cost\""


print(cont_to_exp(x))
```

```
     i do not know what it is going to cost. You should have told me the cost"
```

```
df['Review Text'] = df['Review Text'].apply(lambda x: cont_to_exp(x))


df.head()
```

	Age	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	34	Love this dress! it is sooo pretty. i	5	1	4	General	Dresses	Dresses

```
print(' '.join(df['Review Text'].tolist())[:1000])
```

```
Absolutely wonderful - silky and sexy and comfortable Love this dress!  it is sooo pret
```

Feature Engineering

```
df.head()
```

	Age	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	34	Love this dress! it is sooo pretty. i happen	5	1	4	General	Dresses	Dresses

```
from textblob import TextBlob

df['polarity'] = df['Review Text'].apply(lambda x: TextBlob(x).sentiment.polarity)

#pip install textblob

df['review_length'] = df['Review Text'].apply(lambda x:len(x))

df['word_count'] = df['Review Text'].apply(lambda x:len(x.split()))

def get_avg_word_len(x):
  words = x.split()
  word_len = 0
```

```
    for word in words:
      word_len = word_len + len(word)

    return word_len/len(words)

df['avg_word_len'] = df['Review Text'].apply(lambda x: get_avg_word_len(x))
```

```
df.head()
```

	Age	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name	polarit
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates	0.63333
1	34	Love this dress! it is sooo pretty. i happen...	5	1	4	General	Dresses	Dresses	0.33958

Distribution of Sentiment Polarity

```
df.iplot()
```

```
plt=df['polarity'].iplot(kind="hist",title="Sentiment Polarity Distribution",color="red",xTit
                         linecolor="white",bins=50)
```

Distribution of Reviews Rating and Reviewers Age

```
df['Rating'].iplot(kind='hist',xTitle='Rating',yTitle='Count',title='Review Rating Distributi
```

```
df['Age'].iplot(kind='hist',bins=40,xTitle='Age',yTitle='Count',
                title='Reviewers Age Distribution',color="blue",linecolor="black")
```

Distribution of Review Text Length and Word Length

```
df['review_length'].iplot(kind='hist',xTitle='Review Len',yTitle='Count',
                          title='Review Text Length Distribution')
```

```
df['word_count'].iplot(kind='hist',xTitle='Review Len',yTitle='Count',linecolor="black"
                       ,title='Review Text Length Distribution',color='green')
```

```
df['avg_word_len'].iplot(kind='hist',xTitle='Avg Word Len',yTitle='Count',linecolor="black"
                         ,title='Average Word Length Distribution',color='coral',bins=50)
```

Distribution of Department, Division and Class

```
df['Department Name'].value_counts()
```

```
Tops        10048
Dresses      6145
Bottoms      3662
Intimate     1653
Jackets      1002
Trend         118
Name: Department Name, dtype: int64
```

```
df.groupby('Department Name').count()
```

Department Name	Age	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Class Name	polarity	rev
Bottoms	3662	3662	3662	3662	3662	3662	3662	3662	
Dresses	6145	6145	6145	6145	6145	6145	6145	6145	
Intimate	1653	1653	1653	1653	1653	1653	1653	1653	
Jackets	1002	1002	1002	1002	1002	1002	1002	1002	

```
df['Department Name'].value_counts().iplot(kind="bar",yTitle="Count",xTitle='Department')
```

```
df['Division Name'].value_counts().iplot(kind="bar",yTitle="Count",
                                         xTitle='Division',color="pink")
```

```
df['Class Name'].value_counts().iplot(kind="bar",yTitle="Count",
                                         xTitle='Division',color="brown")
```

Distribution of Unigram, Bigram and Trigram

```
from sklearn.feature_extraction.text import CountVectorizer

x = ['this is the list list this']

vec = CountVectorizer().fit(x)
vec
```

```
CountVectorizer()
```

```
bow = vec.transform(x)
bow
```

```
<1x4 sparse matrix of type '<class 'numpy.int64'>'
        with 4 stored elements in Compressed Sparse Row format>
```

Unigram

```
vec = CountVectorizer().fit(x)
bow = vec.transform(x)
sum_words = bow.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
words_freq[:2]
```

```
[('this', 2), ('list', 2)]
```

```
sum_words
```

```
matrix([[1, 2, 1, 2]])
```

```
def get_top_n_words(x,n):
    vec = CountVectorizer().fit(x)
    bow = vec.transform(x)
    sum_words = bow.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    return words_freq[:n]
```

```
get_top_n_words(x,2)
```

```
[('this', 2), ('list', 2)]
```

```
words = get_top_n_words(df['Review Text'],20)
```

```
words
```

```
[('the', 76159),
 ('it', 49273),
 ('and', 49008),
 ('is', 38103),
 ('this', 25758),
 ('to', 24577),
 ('in', 20722),
 ('but', 16549),
 ('not', 16203),
 ('on', 15325),
 ('for', 13994),
 ('of', 13428),
 ('was', 12923),
 ('with', 12797),
 ('so', 12017),
```

```
        ('am', 11174),
        ('my', 11027),
        ('dress', 10567),
        ('that', 10011),
        ('love', 8945)]
```

```
df1 = pd.DataFrame(words, columns=['Unigram','Frequency'])
df1 = df1.set_index('Unigram')
df1.iplot(kind="bar",xTitle='Unigram',yTitle='Count',title='Top 20 Unigram Words')
```

Bigram

```
vec = CountVectorizer(ngram_range=(2,2)).fit(x)
bow = vec.transform(x)
sum_words = bow.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
words_freq[:2]
```

```
[('this is', 1), ('is the', 1)]
```

```
def get_top_n_words(x,n):
    vec = CountVectorizer(ngram_range=(2,2)).fit(x)
    bow = vec.transform(x)
    sum_words = bow.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    return words_freq[:n]


get_top_n_words(x,2)
```

```
[('this is', 1), ('is the', 1)]
```

```
words = get_top_n_words(df['Review Text'],20)


words
```

```
[('it is', 12525),
 ('in the', 7169),
 ('and the', 5613),
 ('this dress', 4758),
 ('on the', 4337),
 ('of the', 3933),
 ('and it', 3723),
 ('it was', 3287),
 ('this top', 2939),
 ('this is', 2772),
 ('is very', 2729),
 ('the fabric', 2655),
 ('love the', 2641),
 ('did not', 2486),
 ('love this', 2331),
 ('to wear', 2266),
 ('but it', 2196),
 ('the dress', 2180),
 ('is not', 2130),
 ('do not', 2117)]
```

```
df2 = pd.DataFrame(words, columns=['Bigram','Frequency'])
df2 = df2.set_index('Bigram')
df2.iplot(kind="bar",xTitle='Bigram',yTitle='Count',title='Top 20 Bigram Words')
```

Trigram

```
def get_top_n_words(x,n):
    vec = CountVectorizer(ngram_range=(3,3)).fit(x)
    bow = vec.transform(x)
    sum_words = bow.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    return words_freq[:n]


get_top_n_words(x,3)
```

```
[('this is the', 1), ('is the list', 1), ('the list list', 1)]
```

```
words = get_top_n_words(df['Review Text'],20)
words
```

```
[('true to size', 1316),
 ('the fabric is', 1301),
 ('and it is', 1124),
 ('this dress is', 1123),
 ('it is very', 976),
 ('but it is', 921),
 ('it is not', 910),
```

```
        ('the material is', 896),
        ('in the store', 728),
        ('on the model', 725),
        ('the fit is', 696),
        ('this top is', 672),
        ('the color is', 630),
        ('love this dress', 605),
        ('it is so', 584),
        ('tried it on', 574),
        ('and it was', 550),
        ('in the back', 528),
        ('the dress is', 518),
        ('up or down', 518)]


df3 = pd.DataFrame(words, columns=['Trigram','Frequency'])
df3 = df3.set_index('Trigram')
df3.iplot(kind="bar",xTitle='Trigram',yTitle='Count',title='Top 20 Trigram Words')
```

Distribution of Unigram, Bigram and Trigram without STOPWORDS

Unigram

```
def get_top_n_words(x,n):
    vec = CountVectorizer(ngram_range=(1,1),stop_words='english').fit(x)
    bow = vec.transform(x)
    sum_words = bow.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    return words_freq[:n]


get_top_n_words(x,3)
```

```
     [('list', 2)]
```

```
words = get_top_n_words(df['Review Text'],20)
words
```

```
     [('dress', 10567),
      ('love', 8945),
      ('size', 8770),
      ('fit', 7322),
      ('like', 7148),
      ('wear', 6434),
      ('great', 6114),
      ('just', 5604),
      ('fabric', 4797),
      ('small', 4726),
      ('color', 4604),
      ('look', 4039),
      ('really', 3924),
      ('ordered', 3850),
      ('little', 3773),
      ('perfect', 3772),
      ('flattering', 3519),
      ('did', 3447),
      ('soft', 3336),
      ('comfortable', 3058)]
```

```
df2 = pd.DataFrame(words, columns=['Unigram','Frequency'])
df2 = df2.set_index('Unigram')
df2.iplot(kind="bar",xTitle='Unigram',yTitle='Count',title='Top 20 Unigram Words')
```

Bigram

```
def get_top_n_words(x,n):
    vec = CountVectorizer(ngram_range=(2,2),stop_words='english').fit(x)
    bow = vec.transform(x)
    sum_words = bow.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    return words_freq[:n]


get_top_n_words(x,2)
```

```
     [('list list', 1)]
```

```
words = get_top_n_words(df['Review Text'],20)
words
```

```
     [('true size', 1347),
      ('love dress', 766),
      ('usually wear', 694),
      ('looks great', 620),
```

```
        ('fit perfectly', 609),
        ('size small', 540),
        ('fits perfectly', 489),
        ('usual size', 479),
        ('just right', 434),
        ('look like', 426),
        ('ordered size', 375),
        ('love love', 374),
        ('looks like', 373),
        ('runs large', 367),
        ('super cute', 363),
        ('highly recommend', 363),
        ('wear size', 361),
        ('fabric soft', 356),
        ('feel like', 349),
        ('fit great', 348)]


df2 = pd.DataFrame(words, columns=['Bigram','Frequency'])
df2 = df2.set_index('Bigram')
df2.iplot(kind="bar",xTitle='Unigram',yTitle='Count',title='Top 20 Bigram Words')
```

Trigram

```
def get_top_n_words(x,n):
    vec = CountVectorizer(ngram_range=(3,3),stop_words='english').fit(x)
    bow = vec.transform(x)
    sum_words = bow.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    return words_freq[:n]

#get_top_n_words(x,3)



words = get_top_n_words(df['Review Text'],20)
words
```

```
     [('fits true size', 280),
      ('fit true size', 229),
      ('runs true size', 148),
      ('love love love', 143),
      ('usually wear size', 136),
      ('ordered usual size', 107),
      ('does run large', 96),
      ('really wanted love', 94),
      ('wanted love dress', 88),
      ('usually wear small', 80),
      ('small fit perfectly', 77),
      ('just did work', 71),
      ('looks great jeans', 70),
      ('fit like glove', 69),
      ('usually wear medium', 68),
      ('normally wear size', 68),
      ('small fits perfectly', 65),
      ('size fit perfectly', 65),
      ('fits like glove', 65),
      ('usual size small', 64)]
```

```
df2 = pd.DataFrame(words, columns=['Trigram','Frequency'])
df2 = df2.set_index('Trigram')
df2.iplot(kind="bar",xTitle='Trigram',yTitle='Count',title='Top 20 Trigram Words')
```

Distribution of Top 20 Parts-of-Speech POS tags

```
import nltk


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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
print(str(df['Review Text']))
```

```
0        Absolutely wonderful - silky and sexy and comf...
1        Love this dress!  it is sooo pretty.  i happen...
2        I had such high hopes for this dress and reall...
3        I love, love, love this jumpsuit. it is fun, f...
4        This shirt is very flattering to all due to th...
                              ...
23481    I was very happy to snag this dress at such a ...
23482    It reminds me of maternity clothes. soft, stre...
23483    This fit well, but the top was very see throug...
23484    I bought this dress for a wedding i have this ...
23485    This dress in a lovely platinum is feminine an...
Name: Review Text, Length: 22628, dtype: object
```

```
blob = TextBlob(str(df['Review Text']))
```

```
blob
```

```
TextBlob("0        Absolutely wonderful - silky and sexy and comf...
1        Love this dress!  it is sooo pretty.  i happen...
2        I had such high hopes for this dress and reall...
3        I love, love, love this jumpsuit. it is fun, f...
4        This shirt is very flattering to all due to th...
                              ...
23481    I was very happy to snag this dress at such a ...
23482    It reminds me of maternity clothes. soft, stre...
23483    This fit well, but the top was very see throug...
23484    I bought this dress for a wedding i have this ...
23485    This dress in a lovely platinum is feminine an...
Name: Review Text, Length: 22628, dtype: object")
```

```
blob.tags
```

```
('was', 'VBD'),
('very', 'RB'),
('happy', 'JJ'),
('to', 'TO'),
('snag', 'VB'),
('this', 'DT'),
('dress', 'NN'),
('at', 'IN'),
('such', 'PDT'),
('a', 'DT'),
('23482', 'CD'),
('It', 'PRP'),
('reminds', 'VBZ'),
('me', 'PRP'),
('of', 'IN'),
('maternity', 'NN'),
('clothes', 'NNS'),
('soft', 'JJ'),
('stre', 'NN'),
('23483', 'CD'),
('This', 'DT'),
('fit', 'NN'),
('well', 'RB'),
('but', 'CC'),
('the', 'DT'),
('top', 'NN'),
('was', 'VBD'),
('very', 'RB'),
('see', 'JJ'),
('throug', 'NN'),
('23484', 'CD'),
('I', 'PRP'),
('bought', 'VBD'),
('this', 'DT'),
('dress', 'NN'),
('for', 'IN'),
('a', 'DT')
```

```
 ('a', 'DT'),
 ('wedding', 'NN'),
 ('i', 'NN'),
 ('have', 'VBP'),
 ('this', 'DT'),
 ('23485', 'CD'),
 ('This', 'DT'),
 ('dress', 'NN'),
 ('in', 'IN'),
 ('a', 'DT'),
 ('lovely', 'JJ'),
 ('platinum', 'NN'),
 ('is', 'VBZ'),
 ('feminine', 'JJ'),
 ('an', 'DT'),
 ('Name', 'NN'),
 ('Review', 'NNP'),
 ('Text', 'NNP'),
 ('Length', 'NNP'),
 ('22628', 'CD'),
 ('dtype', 'NN'),
 ('object', 'NN')]
```

```
nltk.download('tagsets')
```

```
[nltk_data] Downloading package tagsets to /root/nltk_data...
[nltk_data]   Unzipping help/tagsets.zip.
True
```

```
print(nltk.help.upenn_tagset())
```

```
    occasionally unabatingly maddeningly adventurously professedly
    stirringly prominently technologically magisterially predominately
    swiftly fiscally pitilessly ...
RBR: adverb, comparative
    further gloomier grander graver greater grimmer harder harsher
    healthier heavier higher however larger later leaner lengthier less-
    perfectly lesser lonelier longer louder lower more ...
RBS: adverb, superlative
    best biggest bluntest earliest farthest first furthest hardest
    heartiest highest largest least less most nearest second tightest worst
RP: particle
    aboard about across along apart around aside at away back before behind
    by crop down ever fast for forth from go high i.e. in into just later
    low more off on open out over per pie raising start teeth that through
    under unto up up-pp upon whole with you
SYM: symbol
    % & ' '' ''. ) ). * + ,. < = > @ A[fj] U.S U.S.S.R * ** ***
TO: "to" as preposition or infinitive marker
    to
UH: interjection
    Goodbye Goody Gosh Wow Jeepers Jee-sus Hubba Hey Kee-reist Oops amen
    huh howdy uh dammit whammo shucks heck anyways whodunnit honey golly
    man baby diddle hush sonuvabitch ...
```

```
    VB: verb, base form
        ask assemble assess assign assume atone attention avoid bake balkanize
        bank begin behold believe bend benefit bevel beware bless boil bomb
        boost brace break bring broil brush build ...
    VBD: verb, past tense
        dipped pleaded swiped regummed soaked tidied convened halted registered
        cushioned exacted snubbed strode aimed adopted belied figgered
        speculated wore appreciated contemplated ...
    VBG: verb, present participle or gerund
        telegraphing stirring focusing angering judging stalling lactating
        hankerin' alleging veering capping approaching traveling besieging
        encrypting interrupting erasing wincing ...
    VBN: verb, past participle
        multihulled dilapidated aerosolized chaired languished panelized used
        experimented flourished imitated reunifed factored condensed sheared
        unsettled primed dubbed desired ...
    VBP: verb, present tense, not 3rd person singular
        predominate wrap resort sue twist spill cure lengthen brush terminate
        appear tend stray glisten obtain comprise detest tease attract
        emphasize mold postpone sever return wag ...
    VBZ: verb, present tense, 3rd person singular
        bases reconstructs marks mixes displeases seals carps weaves snatches
        slumps stretches authorizes smolders pictures emerges stockpiles
        seduces fizzes uses bolsters slaps speaks pleads ...
    WDT: WH-determiner
        that what whatever which whichever
    WP: WH-pronoun
        that what whatever whatsoever which who whom whosoever
    WP$: WH-pronoun, possessive
        whose
    WRB: Wh-adverb
        how however whence whenever where whereby whereever wherein whereof why
    ``: opening quotation mark
        ` ``
    None
```

```
pos_df = pd.DataFrame(blob.tags, columns=['words','pos'])
pos_df = pos_df['pos'].value_counts()
pos_df
```

```
    NN     23
    DT     15
    JJ     13
    CD     11
    PRP     8
    RB      6
    VBZ     5
    VBP     5
    IN      5
    CC      4
    NNP     4
    VBD     4
    TO      3
    NNS     2
    VB      2
```

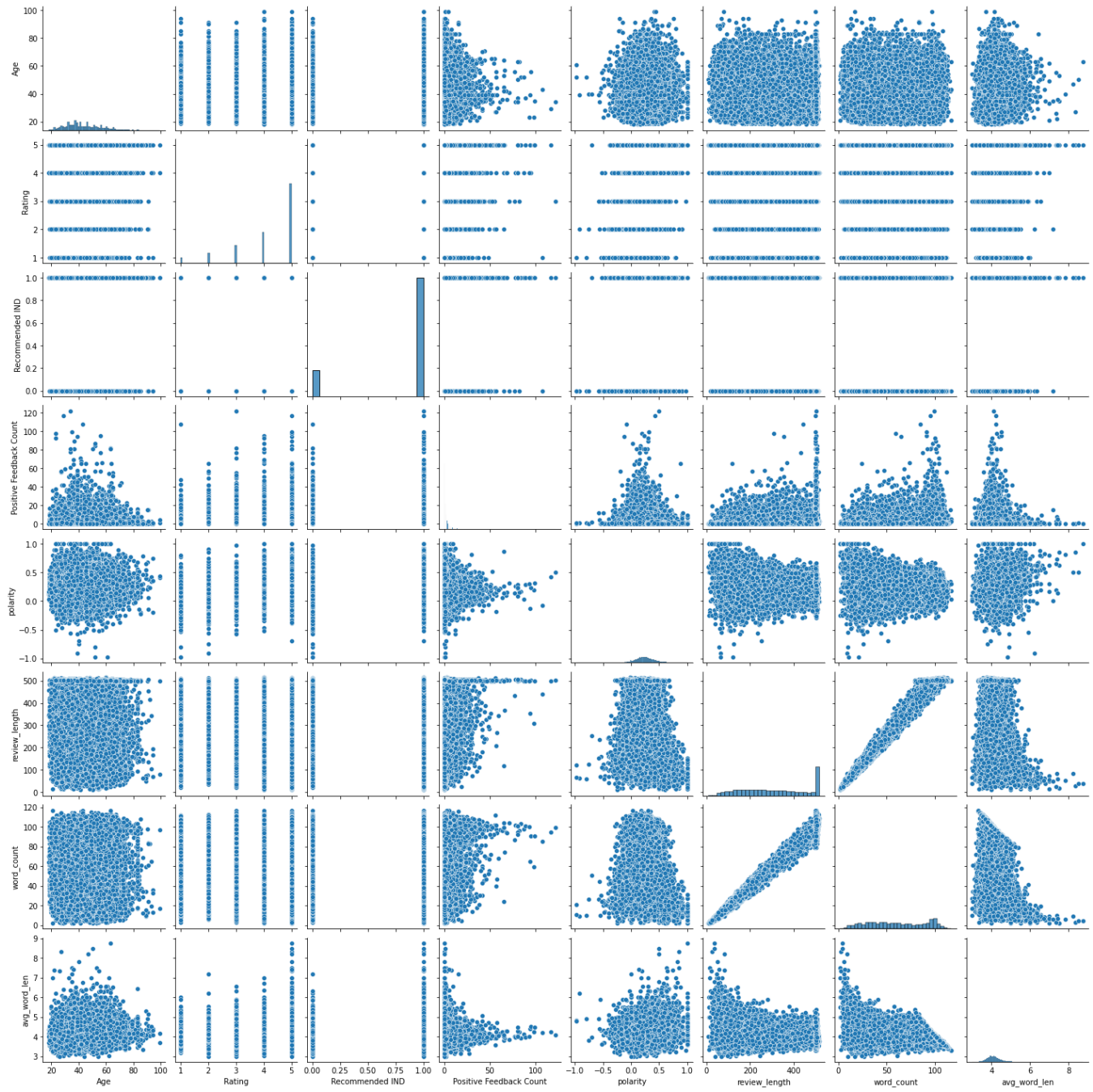
```
PDT     1
Name: pos, dtype: int64
```

```
pos_df.iplot(kind='bar')
```

Bivariate Analysis

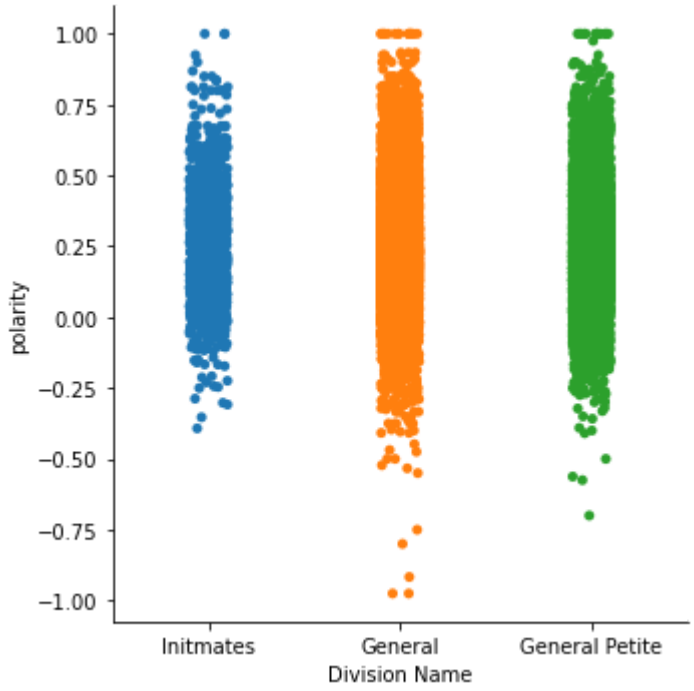
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f023491c4d0>
```



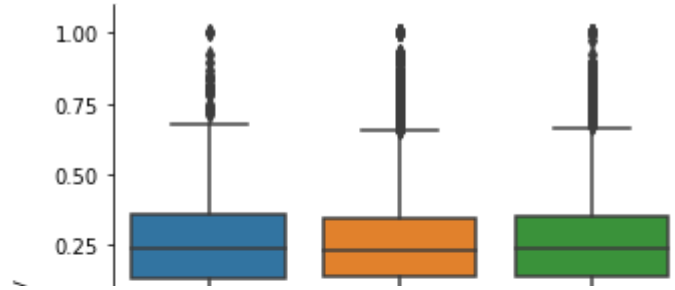
```
sns.catplot(x='Division Name',y='polarity',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0231d56a90>
```



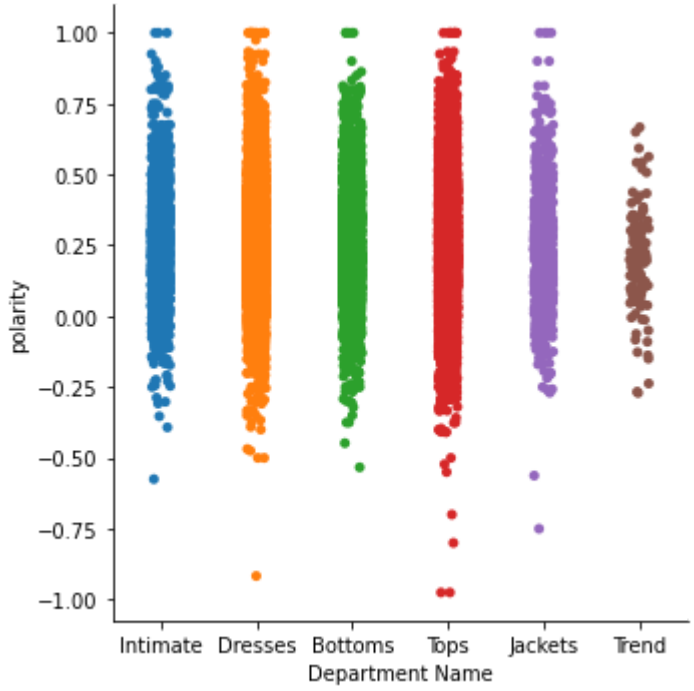

```
sns.catplot(x='Division Name',y='polarity',data=df,kind='box')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0233e88e50>
```




```
sns.catplot(x='Department Name',y='polarity',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0231bce850>
```

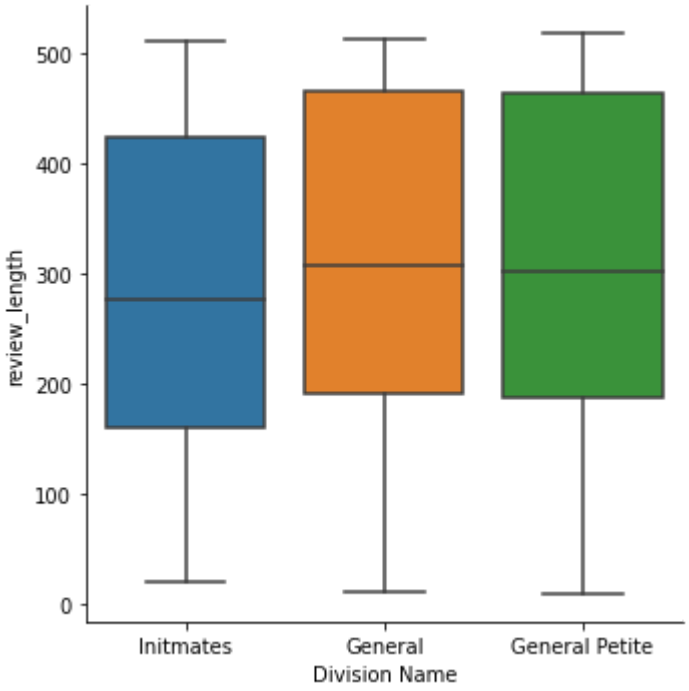



```
sns.catplot(x='Department Name',y='polarity',data=df,kind='box')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0231613710>
```

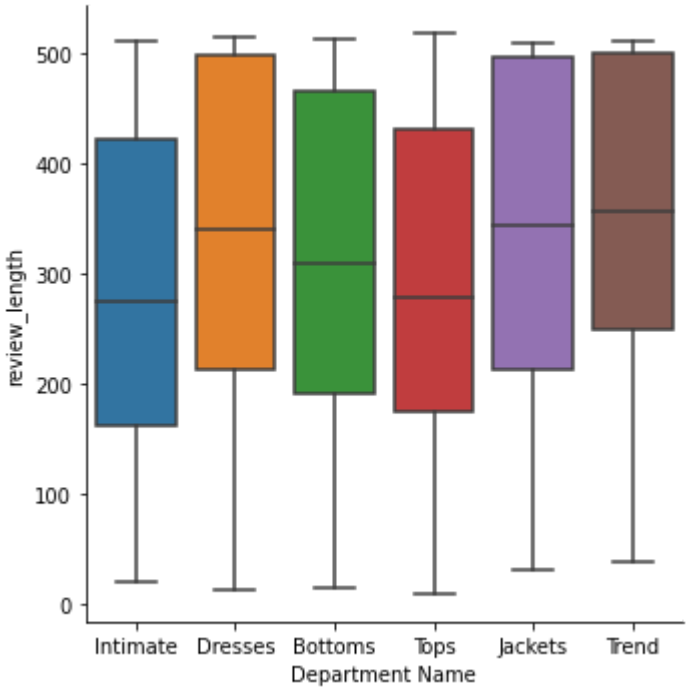

```
sns.catplot(x='Division Name',y="review_length",data=df,kind="box")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0231499e50>
```




```
sns.catplot(x='Department Name',y="review_length",data=df,kind="box")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f02319851d0>
```



Distribution of Sentiment Polarity with Recommended Scores

```
import plotly.express as px
import plotly.graph_objects as go


x1 = df[df['Recommended IND']==1]['polarity']
x0=df[df['Recommended IND']==0]['polarity']



trace0 = go.Histogram(x=x0,name='Not Recommended',opacity=0.7)


trace1 = go.Histogram(x=x1,name='Recommended',opacity=0.7)


data = [trace0,trace1]
layout = go.Layout(barmode='overlay',
 title='Distribution of sentiment polarity based on the Recommended Scores')
fig = go.Figure(data=data,layout=layout)
iplot(fig)
```

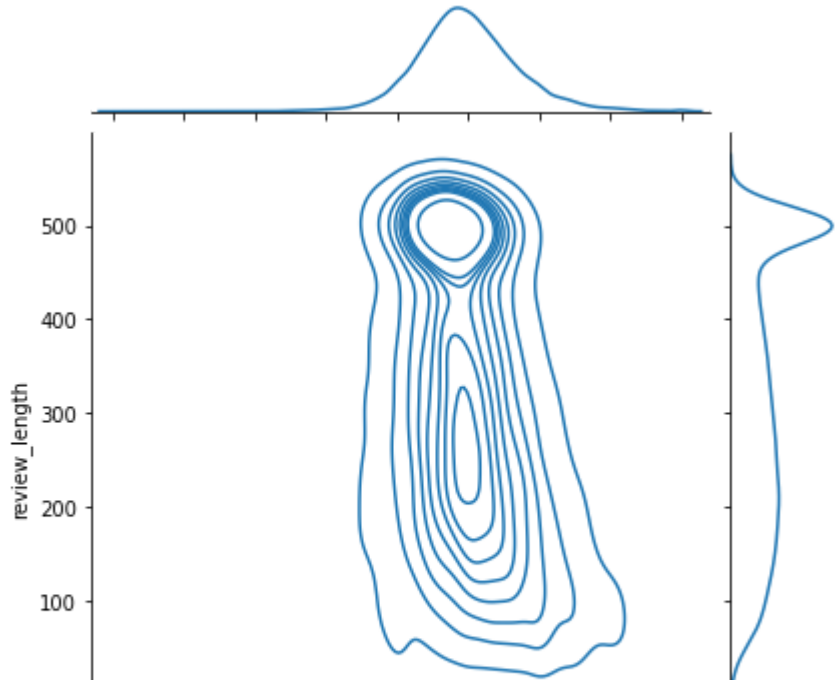
Distribution of Ratings Based on the Recommendation

```
x1 = df[df['Recommended IND']==1]['Rating']
x0=df[df['Recommended IND']==0]['Rating']



trace0 = go.Histogram(x=x0,name='Not Recommended',opacity=0.7)


trace1 = go.Histogram(x=x1,name='Recommended',opacity=0.7)


data = [trace0,trace1]
layout = go.Layout(barmode='overlay',
 title='Distribution of Rating Reviews based on the Recommended Scores')
fig = go.Figure(data=data,layout=layout)
iplot(fig)
```

```
x1 = df[df['Recommended IND']==1]['polarity']
x0=df[df['Recommended IND']==0]['polarity']


trace0 = go.Histogram(x=x0,name='Not Recommended',opacity=0.7)

trace1 = go.Histogram(x=x1,name='Recommended',opacity=0.7)

data = [trace0,trace1]
layout = go.Layout(barmode='overlay',
 title='Distribution of sentiment polarity based on the Recommended Scores')
fig = go.Figure(data=data,layout=layout)
iplot(fig)
```

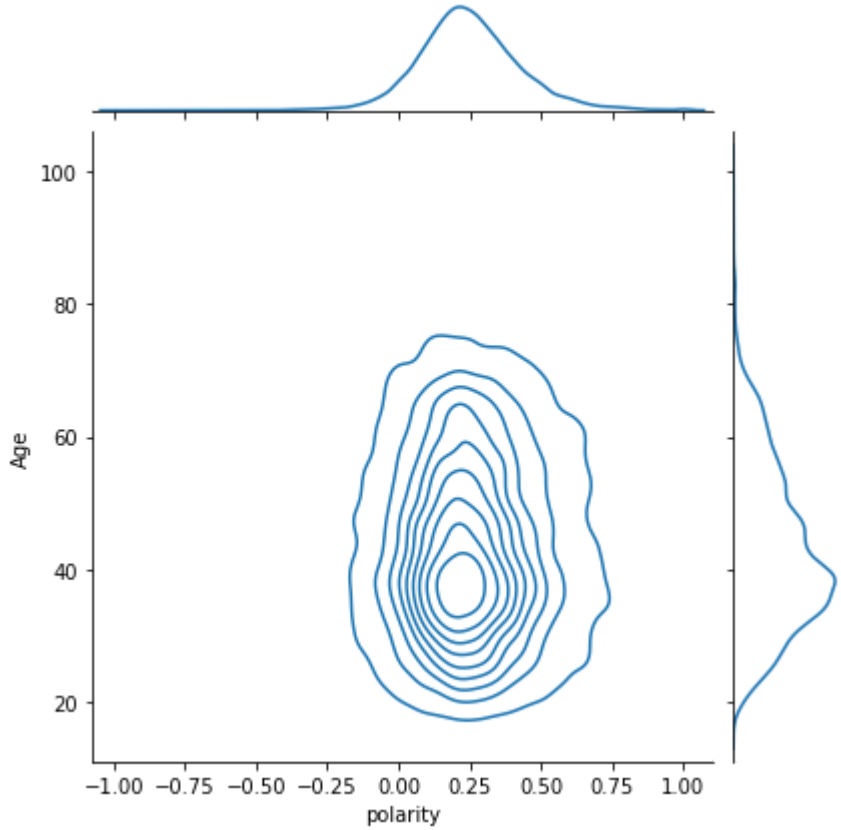
```
sns.jointplot(x='polarity',y='review_length',data=df,kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7f0230ce4590>
```




```
sns.jointplot(x='polarity',y='Age',data=df,kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7f0230dd6210>
```



✓ 0s completed at 1:15 AM