2 - EDA with Text

November 23, 2021

Table of Contents

Get data from kaggle.com

Begin work

Basic pre-processing

New column for sentiment polarity. Two new columns for lengths of the review and word count.

Dist plot of review polarity score

Distribution of ratings

Reviewers age distribution

2D Density Jointplot of Age and Rating

2D Density Jointplot of Age and Sentiment Polarity

Ratings between age groups

2D Density Jointplot of Sentiment Polarity vs. Rating

Distribution of sentiment polarity score by recommendations

Distribution of review ratings based on recommendations

Top unigrams before removing stop words

Top unigrams after removing stop words

Top bigrams before removing stop words

Top bigrams after removing stop words

Top trigrams before removing stop words

Top trigrams after removing stop words

Top 20 part-of-speech tagging of review corpus

Distribution of review lengths by recommendations

Distribution of word count by recommendations

Originally - Li EDA

0.0.1 Get data from kaggle.com

[ ]:
```
#from google.colab import drive
#drive.mount('/content/drive')
```

[ ]:
```
#from google.colab import files

## Upload your kaggle json file (API Token)
#files.upload()

#!mkdir ~/.kaggle

#!cp kaggle.json ~/.kaggle/

#!chmod 600 ~/.kaggle/kaggle.json
```

[ ]:
```
#!kaggle datasets download -d nicapotato/womens-ecommerce-clothing-reviews
```

[ ]:
```
#!ls
```

[ ]:
```
#!mkdir data

#!unzip womens-ecommerce-clothing-reviews -d data
```

[ ]:
```
#!ls -l data/
```

0.0.2 Begin work

[ ]:
```
import pandas as pd
import numpy as np
from textblob import TextBlob
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (10.0, 6.0)
import plotly.graph_objs as go
import plotly.express as py
import cufflinks
# cufflinks connects the Pandas df with Plotly enabling users to create␣
 ↪visualizations directly from Pandas.
pd.options.display.max_columns = 30
from IPython.core.interactiveshell import InteractiveShell
import plotly.figure_factory as ff
InteractiveShell.ast_node_interactivity = 'all'
from plotly.offline import iplot
cufflinks.go_offline()
```

```
cufflinks.set_config_file(world_readable=True, theme='pearl')
output_notebook()
```

[ ]:
```
df = pd.read_csv('/Users/jimcody/Documents/2021Python/nlp/data/Womens Clothing␣
↪E-Commerce Reviews.csv')
df.head()
```

[ ]:
```
df.shape
```

[ ]:
```
df.info()
```

0.0.3 Basic pre-processing

[ ]:
```
df.drop('Unnamed: 0', axis=1, inplace=True)
```

[ ]:
```
#a = 0
#for i in range(a,a+4):
#    print(df['Review Text'][i])
#    print(df['Rating'][i])
#    print(df['Recommended IND'][i])
#    print()
```

[ ]:
```
df.drop('Title', axis=1, inplace=True)
df = df[~df['Review Text'].isnull()]
```

[ ]:
```
# checking for null values, if any

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

[ ]:
```
print(df.Rating.value_counts())
print(' ')
print(df['Division Name'].value_counts())
print(' ')
print(df['Department Name'].value_counts())
print(' ')
print(df['Class Name'].value_counts())
print(' ')
```

[ ]:
```
def preprocess(ReviewText):
    ReviewText = ReviewText.str.replace("(<br/>)", "")
    ReviewText = ReviewText.str.replace('(<a).*(>).*(</a>)', '')
    ReviewText = ReviewText.str.replace('(&amp)', '')
    ReviewText = ReviewText.str.replace('(&gt)', '')
    ReviewText = ReviewText.str.replace('(&lt)', '')
    ReviewText = ReviewText.str.replace('(\xa0)', ' ')
    return ReviewText
df['Review Text'] = preprocess(df['Review Text'])
```

0.0.4 New column for sentiment polarity. Two new columns for lengths of the review and word count.

[ ]:
```
df['polarity'] = df['Review Text'].map(lambda text: TextBlob(text).sentiment.
 ↪polarity)
df['review_len'] = df['Review Text'].astype(str).apply(len)
df['word_count'] = df['Review Text'].apply(lambda x: len(str(x).split()))
```

[ ]:
```
print('5 random reviews with the highest positive sentiment polarity: \n')
cl = df.loc[df.polarity == 1, ['Review Text']].sample(5).values
for c in cl:
    print(c[0])
```

[ ]:
```
print('5 random reviews with the most neutral sentiment(zero) polarity: \n')
cl = df.loc[df.polarity == 0, ['Review Text']].sample(5).values
for c in cl:
    print(c[0])
```

[ ]:
```
df.polarity.min()
```

[ ]:
```
df.loc[df.polarity == -0.97500000000000009]
```

[ ]:
```
print('2 reviews with the most negative polarity: \n')
cl = df.loc[df.polarity == -0.97500000000000009, ['Review Text']].sample(2).
 ↪values
for c in cl:
    print(c[0])
```

0.0.5 Dist plot of review polarity score

[ ]:
```
df['polarity'].iplot(
    kind='hist',
    bins=50,
    xTitle='polarity',
    linecolor='black',
    yTitle='count',
    title='Sentiment Polarity Distribution')
```

Vast majority of the polarity are greater than 0, means most of them are positive.

0.0.6 Distribution of ratings

[ ]:
```
df['Rating'].iplot(
    kind='hist',
    xTitle='rating',
    linecolor='black',
    yTitle='count',
    title='Review Rating Distribution')
```

The ratings are in align with the polarity, that is, most of the ratings are at 4 or 5 range.

0.0.7 Reviewers age distribution

[ ]:
```
df['Age'].iplot(
     kind='hist',
     bins=50,
     xTitle='age',
     linecolor='black',
     yTitle='count',
     title='Reviewers Age Distribution')
```

Most reviewers are in their 30s to 40s.

[ ]:
```
df['review_len'].iplot(
    kind='hist',
    bins=100,
    xTitle='review length',
    linecolor='black',
    yTitle='count',
    title='Review Text Length Distribution')
```

[ ]:
```
df['word_count'].iplot(
     kind='hist',
     bins=100,
     xTitle='word count',
     linecolor='black',
     yTitle='count',
     title='Review Text Word Count Distribution')
```

There were quite number of people like to leave long reviews.

[ ]:
```
df.groupby('Division Name').count()['Clothing ID'].iplot(kind='bar',␣
↪yTitle='Count', linecolor='black', opacity=0.8,
                                                         title='Bar chart of␣
↪Division Name', xTitle='Division Name')
```

[ ]:
```
df.groupby('Department Name').count()['Clothing ID'].
↪sort_values(ascending=False).iplot(kind='bar', yTitle='Count',␣
↪linecolor='black', opacity=0.8,
                                                          title='Bar chart of␣
↪Department Name', xTitle='Department Name')
```

[ ]:
```
df.groupby('Class Name').count()['Clothing ID'].sort_values(ascending=False).
↪iplot(kind='bar', yTitle='Count', linecolor='black', opacity=0.8,
                                                         title='Bar chart of␣
↪Class Name', xTitle='Class Name')
```

0.0.8 2D Density Jointplot of Age and Rating

[ ]:
```
trace1 = go.Scatter(
    x=df['Age'], y=df['Rating'], mode='markers', name='points',
    marker=dict(color='rgb(102,0,0)', size=2, opacity=0.4)
)
trace2 = go.Histogram2dContour(
    x=df['Age'], y=df['Rating'], name='density', ncontours=20,
    colorscale='Hot', reversescale=True, showscale=False
)
trace3 = go.Histogram(
    x=df['Age'], name='Age density',
    marker=dict(color='rgb(102,0,0)'),
    yaxis='y2'
)
trace4 = go.Histogram(
    y=df['Rating'], name='Rating density', marker=dict(color='rgb(102,0,0)'),
    xaxis='x2'
)
data = [trace1, trace2, trace3, trace4]

layout = go.Layout(
    showlegend=False,
    autosize=False,
    width=600,
    height=550,
    xaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    margin=dict(
        t=50
    ),
    hovermode='closest',
    bargap=0,
    xaxis2=dict(
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    ),
    yaxis2=dict(
```

```
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    )
)

fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='2dhistogram-2d-density-plot-subplots')
```

People in their 30s are likely to give high ratings.

0.0.9 2D Density Jointplot of Age and Sentiment Polarity

[ ]:
```
trace1 = go.Scatter(
    x=df['Age'], y=df['polarity'], mode='markers', name='points',
    marker=dict(color='rgb(102,0,0)', size=2, opacity=0.4)
)
trace2 = go.Histogram2dContour(
    x=df['Age'], y=df['polarity'], name='density', ncontours=20,
    colorscale='Hot', reversescale=True, showscale=False
)
trace3 = go.Histogram(
    x=df['Age'], name='Age density',
    marker=dict(color='rgb(102,0,0)'),
    yaxis='y2'
)
trace4 = go.Histogram(
    y=df['polarity'], name='Sentiment Polarity density',␣
↪marker=dict(color='rgb(102,0,0)'),
    xaxis='x2'
)
data = [trace1, trace2, trace3, trace4]

layout = go.Layout(
    showlegend=False,
    autosize=False,
    width=600,
    height=550,
    xaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
```

```
    ),
    margin=dict(
        t=50
    ),
    hovermode='closest',
    bargap=0,
    xaxis2=dict(
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    ),
    yaxis2=dict(
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    )
)

fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='2dhistogram-2d-density-plot-subplots')
```

There were few people are very positive or very negative, people who give neutral to positive reviews are more likely to be in their 30s. Probably people at these age are likely to be active.

0.0.10 Ratings between age groups

[ ]:
```
df[['Rating', 'Age']].iplot(secondary_y='Age', secondary_y_title='Age',
     kind='box', yTitle='Rating', title='Box Plot of Age and Rating')
```

0.0.11 2D Density Jointplot of Sentiment Polarity vs. Rating

[ ]:
```
trace1 = go.Scatter(
    x=df['polarity'], y=df['Rating'], mode='markers', name='points',
    marker=dict(color='rgb(102,0,0)', size=2, opacity=0.4)
)
trace2 = go.Histogram2dContour(
    x=df['polarity'], y=df['Rating'], name='density', ncontours=20,
    colorscale='Hot', reversescale=True, showscale=False
)
trace3 = go.Histogram(
    x=df['polarity'], name='Sentiment polarity density',
    marker=dict(color='rgb(102,0,0)'),
    yaxis='y2'
)
trace4 = go.Histogram(
    y=df['Rating'], name='Rating density', marker=dict(color='rgb(102,0,0)'),
    xaxis='x2'
```

```
)
data = [trace1, trace2, trace3, trace4]

layout = go.Layout(
    showlegend=False,
    autosize=False,
    width=600,
    height=550,
    xaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    margin=dict(
        t=50
    ),
    hovermode='closest',
    bargap=0,
    xaxis2=dict(
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    ),
    yaxis2=dict(
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    )
)

fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='2dhistogram-2d-density-plot-subplots')
```

0.0.12 Distribution of sentiment polarity score by recommendations

[ ]:
```
x1 = df.loc[df['Recommended IND'] == 1, 'polarity']
x0 = df.loc[df['Recommended IND'] == 0, 'polarity']

trace1 = go.Histogram(
    x=x0, name='Not recommended',
    opacity=0.75
)
```

```
trace2 = go.Histogram(
    x=x1, name = 'Recommended',
    opacity=0.75
)

data = [trace1, trace2]
layout = go.Layout(barmode='overlay', title='Distribution of Sentiment polarity␣
 ↪of reviews based on Recommendation')
fig = go.Figure(data=data, layout=layout)

iplot(fig, filename='overlaid histogram')
```

It is obvious that reviews have higher polarity score are more likely to be recommended.

Apparently, the polarity score for recommended reviews is higher than the polarity score for not recommended reviews.

0.0.13 Distribution of review ratings based on recommendations

[ ]:
```
x1 = df.loc[df['Recommended IND'] == 1, 'Rating']
x0 = df.loc[df['Recommended IND'] == 0, 'Rating']

trace1 = go.Histogram(
    x=x0, name='Not recommended',
    opacity=0.75
)
trace2 = go.Histogram(
    x=x1, name = 'Recommended',
    opacity=0.75
)

data = [trace1, trace2]
layout = go.Layout(barmode='overlay', title='Distribution of Sentiment polarity␣
 ↪of reviews based on Recommendation')
fig = go.Figure(data=data, layout=layout)

iplot(fig, filename='overlaid histogram')
```

Recommended reviews have higher ratings than otherwise.

0.0.14 Top unigrams before removing stop words

[ ]:
```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.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]
common_words = get_top_n_words(df['Review Text'], 20)
for word, freq in common_words:
    print(word, freq)
df1 = pd.DataFrame(common_words, columns = ['ReviewText' , 'count'])
```

[ ]:
```
df1.groupby('ReviewText').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 words in␣
↪review before removing stop words')
```

0.0.15 Top unigrams after removing stop words

[ ]:
```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words = 'english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.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]
common_words = get_top_n_words(df['Review Text'], 20)
for word, freq in common_words:
    print(word, freq)
df2 = pd.DataFrame(common_words, columns = ['ReviewText' , 'count'])
```

[ ]:
```
df2.groupby('ReviewText').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 words in␣
↪review after removing stop words')
```

0.0.16 Top bigrams before removing stop words

[ ]:
```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.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]
common_words = get_top_n_bigram(df['Review Text'], 20)
for word, freq in common_words:
    print(word, freq)
df3 = pd.DataFrame(common_words, columns = ['ReviewText' , 'count'])
```

[ ]:
```
df3.groupby('ReviewText').sum()['count'].sort_values(ascending=False).iplot(
```

```
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 bigrams in␣
 ↪review before removing stop words')
```

0.0.17 Top bigrams after removing stop words

[ ]:
```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.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]
common_words = get_top_n_bigram(df['Review Text'], 20)
for word, freq in common_words:
    print(word, freq)
df4 = pd.DataFrame(common_words, columns = ['ReviewText' , 'count'])
```

[ ]:
```
df4.groupby('ReviewText').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 bigrams in␣
 ↪review after removing stop words')
```

0.0.18 Top trigrams before removing stop words

[ ]:
```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.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]
common_words = get_top_n_trigram(df['Review Text'], 20)
for word, freq in common_words:
    print(word, freq)
df5 = pd.DataFrame(common_words, columns = ['ReviewText' , 'count'])
```

[ ]:
```
df5.groupby('ReviewText').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 trigrams in␣
 ↪review before removing stop words')
```

0.0.19 Top trigrams after removing stop words

[ ]:
```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.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]
common_words = get_top_n_trigram(df['Review Text'], 20)
for word, freq in common_words:
    print(word, freq)
df6 = pd.DataFrame(common_words, columns = ['ReviewText' , 'count'])
```

[ ]:
```
df6.groupby('ReviewText').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 trigrams in␣
 ↪review after removing stop words')
```

0.0.20 Top 20 part-of-speech tagging of review corpus

[ ]:
```
blob = TextBlob(str(df['Review Text']))
pos_df = pd.DataFrame(blob.tags, columns = ['word' , 'pos'])
pos_df = pos_df.pos.value_counts()[:20]
pos_df.iplot(
    kind='bar',
    xTitle='POS',
    yTitle='count',
    title='Top 20 Part-of-speech tagging for review corpus')
```

0.0.21 Distribution of review lengths by recommendations

[ ]:
```
x1 = df.loc[df['Recommended IND'] == 1, 'review_len']
x0 = df.loc[df['Recommended IND'] == 0, 'review_len']

trace1 = go.Histogram(
    x=x0, name='Not recommended',
    opacity=0.75
)
trace2 = go.Histogram(
    x=x1, name = 'Recommended',
    opacity=0.75
)

data = [trace1, trace2]
layout = go.Layout(barmode = 'group', title='Distribution of Review Lengths␣
 ↪Based on Recommendation')
fig = go.Figure(data=data, layout=layout)

iplot(fig, filename='stacked histogram')
```

0.0.22 Distribution of word count by recommendations

[ ]:
```
x1 = df.loc[df['Recommended IND'] == 1, 'word_count']
x0 = df.loc[df['Recommended IND'] == 0, 'word_count']

trace1 = go.Histogram(
    x=x0, name='Not recommended',
    opacity=0.75
)
trace2 = go.Histogram(
    x=x1, name = 'Recommended',
    opacity=0.75
)

data = [trace1, trace2]
layout = go.Layout(barmode = 'group', title='Distribution of Word Count Based␣
↪on Recommendation')
fig = go.Figure(data=data, layout=layout)

iplot(fig, filename='grouped histogram')
```

Recommended reviews tend to be more lengthier than those not recommended reviews.

[ ]:
```
y0 = df.loc[df['Division Name'] == 'General']['polarity']
y1 = df.loc[df['Division Name'] == 'General Petite']['polarity']
y2 = df.loc[df['Division Name'] == 'Initmates']['polarity']

trace0 = go.Box(
    y=y0,
    name = 'General',
    marker = dict(
        color = 'rgb(214, 12, 140)',
    )
)
trace1 = go.Box(
    y=y1,
    name = 'General Petite',
    marker = dict(
        color = 'rgb(0, 128, 128)',
    )
)
trace2 = go.Box(
    y=y2,
    name = 'Initmates',
    marker = dict(
        color = 'rgb(10, 140, 208)',
    )
)
```

```
data = [trace0, trace1, trace2]
layout = go.Layout(
    title = "Sentiment Polarity Boxplot of Division Name"
)

fig = go.Figure(data=data,layout=layout)
iplot(fig, filename = "Sentiment Polarity Boxplot of Division Name")
```

The highest sentiment polarity score was achieved by all of three divisions, and the lowest sentiment polarity score was collected by General division.

We don't see any significant difference in terms of sentiment polarity between division names.

[ ]:
```
y0 = df.loc[df['Department Name'] == 'Tops']['polarity']
y1 = df.loc[df['Department Name'] == 'Dresses']['polarity']
y2 = df.loc[df['Department Name'] == 'Bottoms']['polarity']
y3 = df.loc[df['Department Name'] == 'Intimate']['polarity']
y4 = df.loc[df['Department Name'] == 'Jackets']['polarity']
y5 = df.loc[df['Department Name'] == 'Trend']['polarity']

trace0 = go.Box(
    y=y0,
    name = 'Tops',
    marker = dict(
        color = 'rgb(214, 12, 140)',
    )
)
trace1 = go.Box(
    y=y1,
    name = 'Dresses',
    marker = dict(
        color = 'rgb(0, 128, 128)',
    )
)
trace2 = go.Box(
    y=y2,
    name = 'Bottoms',
    marker = dict(
        color = 'rgb(10, 140, 208)',
    )
)
trace3 = go.Box(
    y=y3,
    name = 'Intimate',
    marker = dict(
        color = 'rgb(12, 102, 14)',
    )
)
```

```
trace4 = go.Box(
    y=y4,
    name = 'Jackets',
    marker = dict(
        color = 'rgb(10, 0, 100)',
    )
)
trace5 = go.Box(
    y=y5,
    name = 'Trend',
    marker = dict(
        color = 'rgb(100, 0, 10)',
    )
)
data = [trace0, trace1, trace2, trace3, trace4, trace5]
layout = go.Layout(
    title = "Sentiment Polarity Boxplot of Department Name"
)

fig = go.Figure(data=data,layout=layout)
iplot(fig, filename = "Sentiment Polarity Boxplot of Department Name")
```

[ ]:
```
y0 = df.loc[df['Department Name'] == 'Tops']['review_len']
y1 = df.loc[df['Department Name'] == 'Dresses']['review_len']
y2 = df.loc[df['Department Name'] == 'Bottoms']['review_len']
y3 = df.loc[df['Department Name'] == 'Intimate']['review_len']
y4 = df.loc[df['Department Name'] == 'Jackets']['review_len']
y5 = df.loc[df['Department Name'] == 'Trend']['review_len']

trace0 = go.Box(
    y=y0,
    name = 'Tops',
    marker = dict(
        color = 'rgb(214, 12, 140)',
    )
)
trace1 = go.Box(
    y=y1,
    name = 'Dresses',
    marker = dict(
        color = 'rgb(0, 128, 128)',
    )
)
trace2 = go.Box(
    y=y2,
    name = 'Bottoms',
    marker = dict(
```

```
        color = 'rgb(10, 140, 208)',
    )
)
trace3 = go.Box(
    y=y3,
    name = 'Intimate',
    marker = dict(
        color = 'rgb(12, 102, 14)',
    )
)
trace4 = go.Box(
    y=y4,
    name = 'Jackets',
    marker = dict(
        color = 'rgb(10, 0, 100)',
    )
)
trace5 = go.Box(
    y=y5,
    name = 'Trend',
    marker = dict(
        color = 'rgb(100, 0, 10)',
    )
)
data = [trace0, trace1, trace2, trace3, trace4, trace5]
layout = go.Layout(
    title = "Review length Boxplot of Department Name"
)

fig = go.Figure(data=data,layout=layout)
iplot(fig, filename = "Review Length Boxplot of Department Name")
```

[ ]:
```
y0 = df.loc[df['Department Name'] == 'Tops']['Rating']
y1 = df.loc[df['Department Name'] == 'Dresses']['Rating']
y2 = df.loc[df['Department Name'] == 'Bottoms']['Rating']
y3 = df.loc[df['Department Name'] == 'Intimate']['Rating']
y4 = df.loc[df['Department Name'] == 'Jackets']['Rating']
y5 = df.loc[df['Department Name'] == 'Trend']['Rating']

trace0 = go.Box(
     y=y0,
     name = 'Tops',
     marker = dict(
         color = 'rgb(214, 12, 140)',
     )
 )
trace1 = go.Box(
```

```
     y=y1,
     name = 'Dresses',
     marker = dict(
        color = 'rgb(0, 128, 128)',
     )
 )
trace2 = go.Box(
  y=y2,
     name = 'Bottoms',
     marker = dict(
         color = 'rgb(10, 140, 208)',
     )
 )
trace3 = go.Box(
     y=y3,
     name = 'Intimate',
     marker = dict(
         color = 'rgb(12, 102, 14)',
     )
 )
trace4 = go.Box(
     y=y4,
     name = 'Jackets',
     marker = dict(
         color = 'rgb(10, 0, 100)',
     )
 )
trace5 = go.Box(
     y=y5,
     name = 'Trend',
    marker = dict(
         color = 'rgb(100, 0, 10)',
     )
 )
data = [trace0, trace1, trace2, trace3, trace4, trace5]
layout = go.Layout(
     title = "Rating Boxplot of Department Name"
 )

fig = go.Figure(data=data,layout=layout)
iplot(fig, filename = "Rating Boxplot of Department Name")
```