Team #3: Milestone 2

Part 1. Data Cleaning

There exist null values in multiple variables:
Title, Review Text, Division Name, Department Name, Class Name.

```
#check for null values from each variable
review.isnull().sum()
```

```
Unnamed: 0                   0
Clothing ID                  0
Age                          0
Title                     3810
Review Text                845
Rating                       0
Recommended IND              0
Positive Feedback Count      0
Division Name               14
Department Name             14
Class Name                  14
dtype: int64
```

Part 1.2. Removing Null Values

To remove the null values from the raw dataset, we start by dropping off columns with missing values in Division Name, Department Name and Class Name using **`dropna()`**. Based on examination, the missing value from the three columns was completely removed with only removal of 14 instances, implicating that there exist 14 instances with missing values from all three categories. It seems common to have missing values from Title and review segments, implicating the customers' negligence to not input any content into their reviews. Therefore, those missing values front text columns can be properly addressed, by re-filling them with a new value: **`Missing`**.

```
#correct version
for i in review.index:
    # Check if the "Title" value at index i is null
    if pd.isnull(review["Title"][i]):
        # Set the "Title" value at index i to 'NA'
        review.loc[i, "Title"] = 'Missing'
```

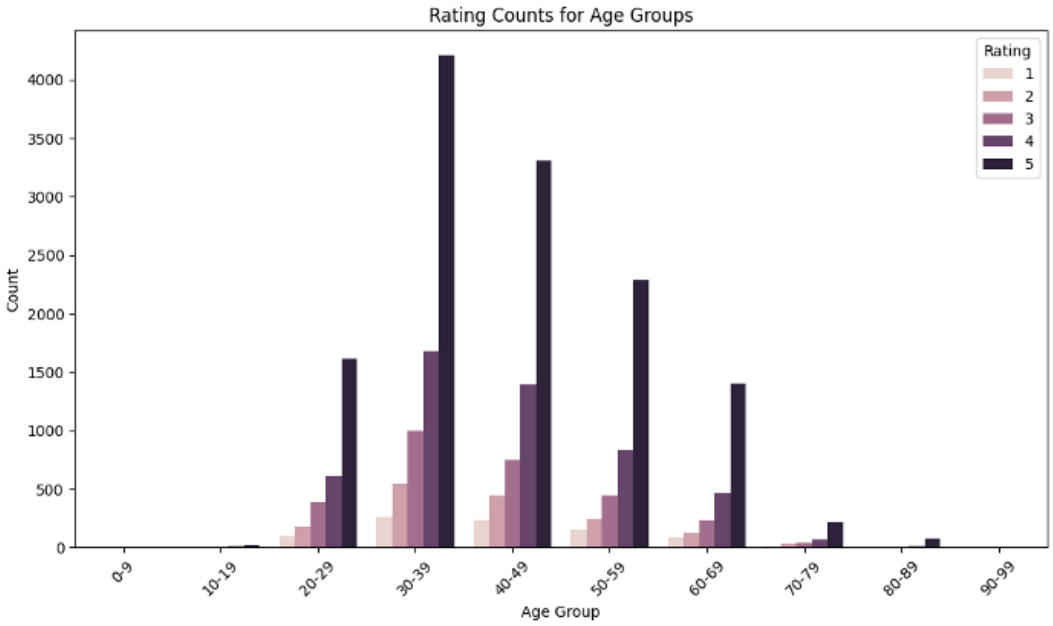
After transformation, the output dataset has all the null values filled and dropped.

```
[ ] review.head()
```

	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	767	33	Missing	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1080	34	Missing	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses
3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants
4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses

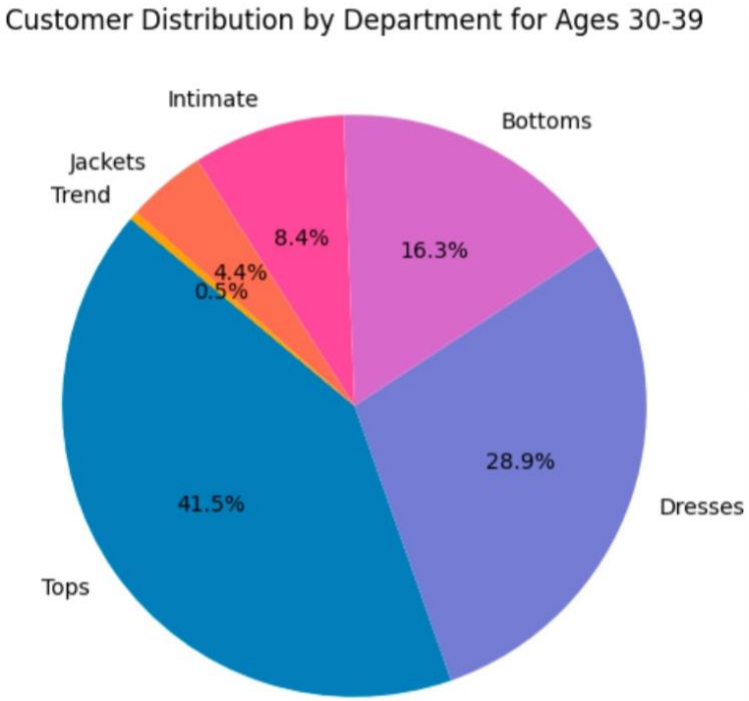
Part 2. EDA (Code Implementation *here*)

EDA #1. Rating counts for age groups



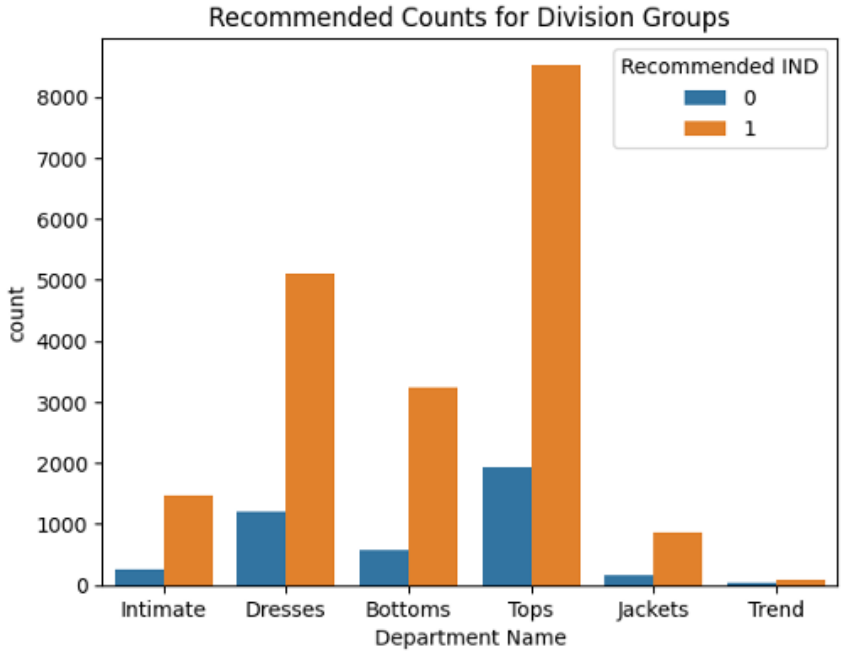
Among all the age groups, the age of reviewers between 30-39 has the most reviews count, and the most positive review counts as well, they are more generous than other age groups. After that, ages between 40-49 have the second highest number of reviews, slightly lower than 30-39 group. Surprisingly, the younger age groups of (20,29) has similar number of reviews as age group 60-69.

EDA #2. Customer distribution by department for age groups 30-39



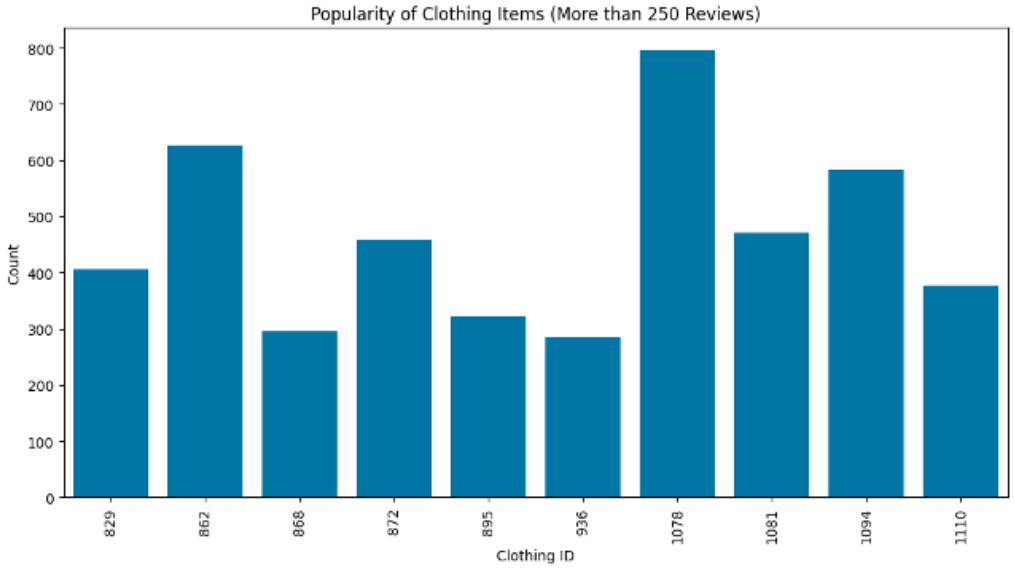
For the age group 30-39, *Tops* department accounts for the most popular department of all divisions with 41.5% of overall reviews. Then *Dresses* takes up 28.9% of all reviews. We next explore all the categories among the most popular age groups.

EDA #3. Division with most/least recommended counts



In our age group 30-39, which has the highest number of reviews, indicating the *Tops* and *Dresses* category are the most popular divisions. Inspected deeply into divisions, we use *Recommended IND* to measure the customer satisfaction and their willingness to give high ratings. Above graph clearly shows the *Tops* and *Dresses* are popular among all age groups.

EDA #4. Top 10 Popular Items



To determine the popularity of items, we firstly filtered the counts of reviews higher than 4 and picked the top 10 counts of reviews of items. Item 1078 has over 700 reviews with the rating higher than 4, followed by item 862, slightly above 600.

Part 3. Text Analysis (Code Implementation *here*)

3.1. Text Preprocessing

- Converting all text to lowercase. This will make sure that all text is consistently formatted across the reviews. So that words can be grouped together in the use case of a clustering use-case.
- Removing punctuation and special characters. This will remove unnecessary noise needed for the classification model used for text analysis.
- Removing numbers and digits from text. This is also to remove noise from the textual data.
- Stemming tokens to obtain the root form of the word using **`PortStemmer`**. This will further group words written in multiple parts of speech into the same root group to prevent the model from having illusion errors.

3.2. Text Classification

One initial approach and two alternative approaches were tried to classify the review text into a positive or negative sentiment value.

3.2.1. Initial Approach:

Text Representation: Word Embeddings with Dimension 100
Classification Model: Logistic Regression
Code and Output:

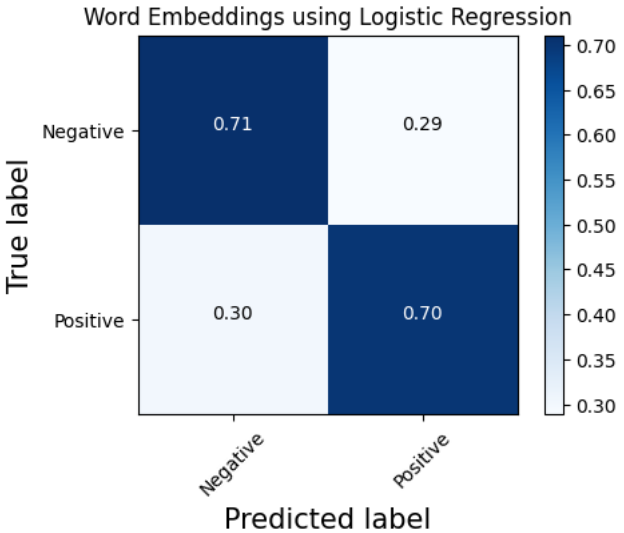
```
# Fit model and test for prediction
logreg = LogisticRegression(class_weight='balanced')
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

print("Logistic Regression Word Embeddings F1 score", metrics.f1_score(y_test, y_pred))
```

✓ 0.6s

```
Logistic Regression Word Embeddings F1 score 0.7241159604969735
```

Word Embedding text representation is delivering an F1-score of 0.72, which is not good enough model performance. Alternative approaches of text representation or model selection will be attempted to achieve a better model performance.



3.2.2. Alternative Approach #1:

Text Representation: Bag of N-Grams
Classification Model: Logistic Regression
Code and Output:

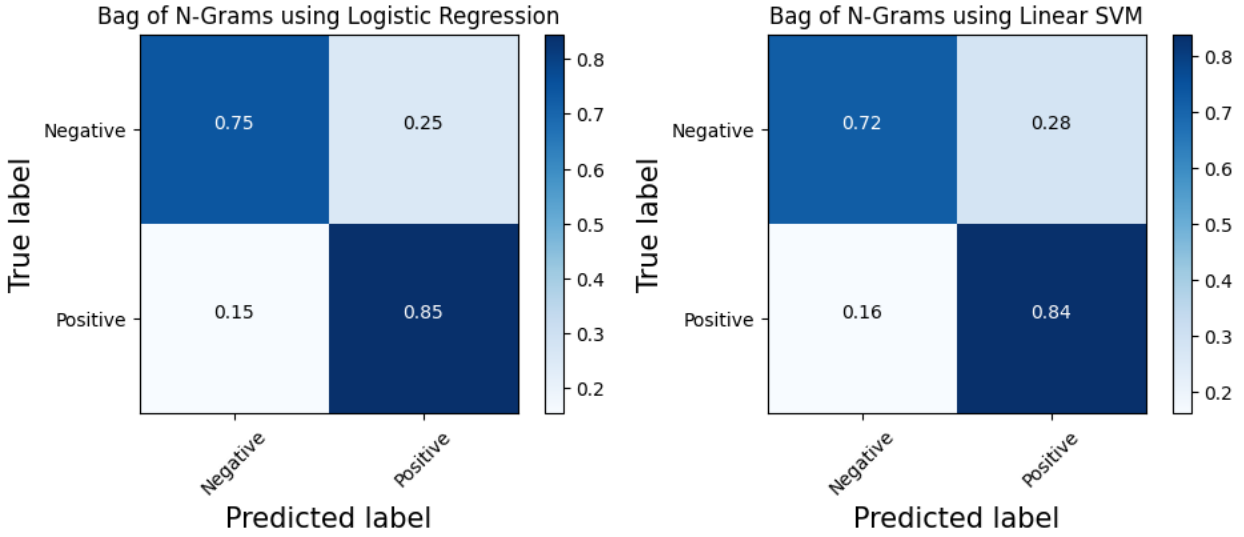
```
# Build model and test for prediction
logreg.fit(X_train_dtm, y_train)
y_pred = logreg.predict(X_test_dtm)

print("Logistic Regression N-Grams F1 score:",metrics.f1_score(y_test, y_pred))
```

✓ 7.7s

```
Logistic Regression N-Grams F1 score: 0.8240949376596065
```

The F-1 score of the text classification using Bag of N-Grams text representation on a Logistic Regression Model is 0.82, which is much higher than the Word Embeddings approach.



3.2.3. Alternative Approach #2:

Text Representation: Bag of N-Grams
Classification Model: Support Vector Machine (SVM)
Code and Output:

```
# Build model and test for prediction
svm = LinearSVC(class_weight='balanced')
svm.fit(X_train_dtm, y_train)
y_pred = svm.predict(X_test_dtm)

print("SVM N-Grams F1 score:",metrics.f1_score(y_test, y_pred))
```

✓ 3.8s

```
SVM N-Grams F1 score: 0.8116980005968367
```

The F-1 score of the text classification using Bag of N-Grams text representation on a Linear SVM Model is 0.81, which is much higher than the Word Embeddings approach, but still lower than the performance of the Logistic Regression Model.

Part 4. Bonus Questions and Exploration

Techniques Explored:

1. CNNs as a classification model (**`TensorFlow Keras`**) – [Code Implementation: *here*]
2. Sentiment Analysis (**`Textblob`** and **`VaderSentiment`**) – [Code Implementation: *here*]

4.1. CNNs as a Classification Model

Why do you apply this new method/tools/packages in this project?
We applied CNN in this project because we didn't employ this model in-class and exercises before. Also, CNNs are efficient in text classification tasks as they can automatically and adaptively learn spatial hierarchies of features from text data, which is crucial for understanding the sentiment or classification of the text.

What is the benefit of using it?
CNNs are great for classifying text because they're good at picking up on important words or phrases, no matter where they are in the input. They can handle this all at once, which makes them faster and more efficient than models that work step-by-step, like RNNs. Plus, CNNs can deal with texts of different lengths and formats, which is handy since natural language can be messy. Hence, CNNs are quick and accurate at pulling out useful information, which helps a lot in getting good results for classification tasks.

Screenshot of the Code:

```
# Define and split training and testing data
X = embedding_feats(reviews_preprocessed)
y = sentiments
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=120994)
```

```
[34] from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Conv1D, Flatten, MaxPooling1D, Embedding
     from tensorflow.keras.optimizers import Adam
     from sklearn.metrics import f1_score
```

```
[40] # Convert X_train and y_train to NumPy arrays
     X_train = np.array(X_train)
     y_train = np.array(y_train)

     # Reshape the data for CNN
     X_train = X_train.reshape((len(X_train), 100, 1))
     X_test = np.array(X_test).reshape((len(X_test), 100, 1))

     # Fit the model
     model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1, verbose=1)
```

```
[41] # Define the CNN model
     model = Sequential()
     model.add(Conv1D(filters=128, kernel_size=3, activation='relu', input_shape=(100, 1)))
     model.add(MaxPooling1D(pool_size=2))
     model.add(Flatten())
     model.add(Dense(10, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))

     # Compile the model
     model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
[42] # Fit the model
     model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1, verbose=1)

     # Predictions
     y_pred = model.predict(X_test)
     y_pred = (y_pred > 0.5).astype(int)
```

Outcome of CNN:

```
[43] # Evaluate the model
     print("CNNS GloVe F1 score:",metrics.f1_score(y_test, y_pred))

     CNNS GloVe F1 score: 0.7616008509506714
```

The F1 score value of 0.76 shows that by using CNN as a classification model, the overall performance is better when compared to using a Logistic Regression model. But still, it is not good enough.

Source: https://towardsdatascience.com/nlp-with-cnns-a6aa743bdc1e

4.2. Sentiment Analysis

Why do you apply this new method/tools/packages in this project?
In the realm of E-Commerce, Sentiment Analysis can be useful to reveal key insights about how your brand, product, or company is viewed by your customers and stakeholders. The insights retrieved can be a great asset for the E-retailer to improve average product qualities via the vendor selection/filtering process.

What is the benefit of using it?

- **`Textblob`** is a great library that can detect both the subjectivity and polarity of the words' context. In such a way we can use this tool to filter out those reviews that tend to be more rational, thus providing higher quality of information.
- **`VaderSentiment`** is a great library that helps to transfer context's sentiment into metric scores, which provides a perspective to view the sentiment quantitatively. Scaling the extent of sentiment can be useful when it comes to market segmentations as well as summative statistics.

Screenshots of the Code:

Using TextBlob to measure sentiment

```
[ ] complaint_df['ReviewText_Polarity'] = complaint_df['Text'].map(lambda text: TextBlob(text).sentiment.polarity)
    complaint_df['ReviewText_Subjectivity'] = complaint_df['Text'].map(lambda text: TextBlob(text).sentiment.subjectivity)
    complaint_df.head()
```

	Clothing ID	Age	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name	Missing	Text	Text_Length	ReviewText_Polarity	ReviewText_Subjectivity
0	767	33	4	1	0	Initmates	Intimate	Intimates	1	Absolutely wonderful - silky and sexy and comf...	16	0.633333	0.933333
1	1080	34	5	1	4	General	Dresses	Dresses	1	Love this dress! it's sooo pretty. i happene...	134	0.339583	0.725000
2	1077	60	3	0	0	General	Dresses	Dresses	0	I had such high hopes for this dress and reall...	196	0.073675	0.356294
3	1049	50	5	1	0	General Petite	Bottoms	Pants	0	I love, love, love this jumpsuit. it's fun, fl...	44	0.550000	0.625000
4	847	47	5	1	6	General	Tops	Blouses	0	This shirt is very flattering to all due to th...	72	0.512891	0.568750

Using VaderSentiment to measure sentiment

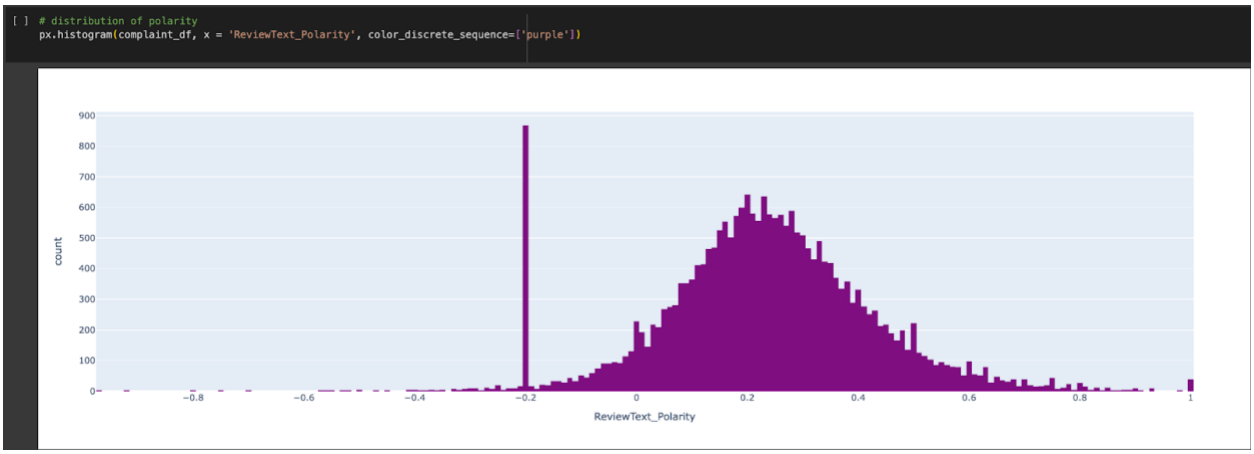
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
complaint_df['vader_score'] = complaint_df['Text'].map(lambda text: analyzer.polarity_scores(text))
complaint_df['vader_neg'] = complaint_df['Text'].map(lambda text: analyzer.polarity_scores(text)['neg'])
complaint_df['vader_neu'] = complaint_df['Text'].map(lambda text: analyzer.polarity_scores(text)['neu'])
complaint_df['vader_pos'] = complaint_df['Text'].map(lambda text: analyzer.polarity_scores(text)['pos'])
complaint_df.head()
```

	Clothing ID	Age	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name	Missing	Text	Text_Length	ReviewText_Polarity	ReviewText_Subjectivity	vader_score	vader_neg	vader_neu	vader_pos
0	767	33	4	1	0	Initmates	Intimate	Intimates	1	Absolutely wonderful - silky and sexy and comf...	16	0.633333	0.933333	{'neg': 0.0, 'neu': 0.319, 'pos': 0.681, 'comp...	0.000	0.319	0.681
1	1080	34	5	1	4	General	Dresses	Dresses	1	Love this dress! it's sooo pretty. i happene...	134	0.339583	0.725000	{'neg': 0.0, 'neu': 0.702, 'pos': 0.298, 'comp...	0.000	0.702	0.298
2	1077	60	3	0	0	General	Dresses	Dresses	0	I had such high hopes for this dress and reall...	196	0.073675	0.356294	{'neg': 0.025, 'neu': 0.824, 'pos': 0.151, 'co...	0.025	0.824	0.151
3	1049	50	5	1	0	General Petite	Bottoms	Pants	0	I love, love, love this jumpsuit. it's fun, fl...	44	0.550000	0.625000	{'neg': 0.171, 'neu': 0.327, 'pos': 0.502, 'co...	0.171	0.327	0.502
4	847	47	5	1	6	General	Tops	Blouses	0	This shirt is very flattering to all due to th...	72	0.512891	0.568750	{'neg': 0.0, 'neu': 0.704, 'pos': 0.296, 'comp...	0.000	0.704	0.296

Outcomes:

Output #1: Sentiment Analysis using **`Textblob`**

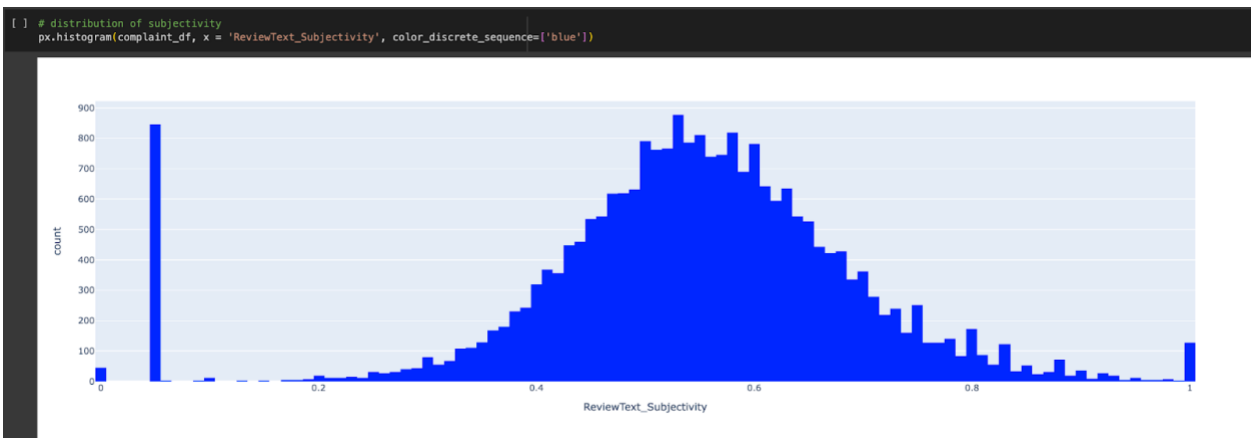
```
# distribution of polarity
px.histogram(complaint_df, x = 'ReviewText_Polarity', color_discrete_sequence=['purple'])
```



By interpreting the polarity, it can be found that most of the reviews' comments are centered around -0.2 to 0.2, implicating that most of the customers have slightly positive views of their products, whereas there also exist many customer reviews that are slightly negative.

Output #2 : Sentiment analysis using **`VaderSentiment`**

```
# distribution of subjectivity
px.histogram(complaint_df, x = 'ReviewText_Subjectivity', color_discrete_sequence=['blue'])
```



In terms of the subjectivity, the plot above indicates a breath of bimodal distribution as a major subgroup of customers provided highly objective reviews while the majority of customers just gave very subjective reviews.

Part 5. Meeting Agenda

Meeting #1:

Prepared by: Niharika Chunduru
Date and time: 02/27/2024, 3 PM
Location: In-person
Team members in attendance: Shi Yang (SY), Yifei Zhao (YZ), Niharika Chunduru (NC), Xiaoxiao Huang (XH)
Meeting objectives: Allocate tasks for Milestone-2 Coding
Agenda: Assigned explicit portions of the Milestone-2 tasks to work on.
Next Actions: Finish coding to compile into a milestone report.

Action Item	Assigned To	Time Spent (approx.)	Due Date
Data Cleaning and Preprocessing Code	YZ	2 hours	02/29/2024
Exploratory Data Analysis Code	SY	3 hours	03/03/2024
Text Representation and Analysis Code	NC	3 hours	03/03/2014
Bonus Questions and Exploration Code	XH, YZ	3 hours each	03/03/2024

Time meeting ended: 3:30PM 02/27/2024.
Date and time of next meeting: 03/03/2024, 7 PM

Meeting #2:

Prepared by: Niharika Chunduru
Date and time: 03/03/2024, 7 PM
Location: Virtual (Zoom)
Team members in attendance: Shi Yang (SY), Yifei Zhao (YZ), Niharika Chunduru (NC), Xiaoxiao Huang (XH)
Meeting objectives: Discuss coding outputs and compile milestone report.
Agenda: The findings from EDA and Text Analysis were discussed, and the milestone compilation to begin.
Next Actions: Compile the findings into the milestone report per the requirements mentioned in the milestone description document.

Action Item	Assigned To	Time Spent (approx.)	Due Date
Data Cleaning and Preprocessing Report	YZ	30 mins	03/04/2024
Exploratory Data Analysis Report	SY	15 mins	03/04/2024
Text Representation and Analysis Report	NC	30 mins	03/04/2024
Bonus Questions and Exploration Report	XH, YZ	30 mins each	03/04/2024
Proof-reading and Editing Report	SY, YZ, NC, XH	2 hours	04/04/2024

Time meeting ended: 03/03/2024, 7:30 PM
Date and time of next meeting: *TBD*