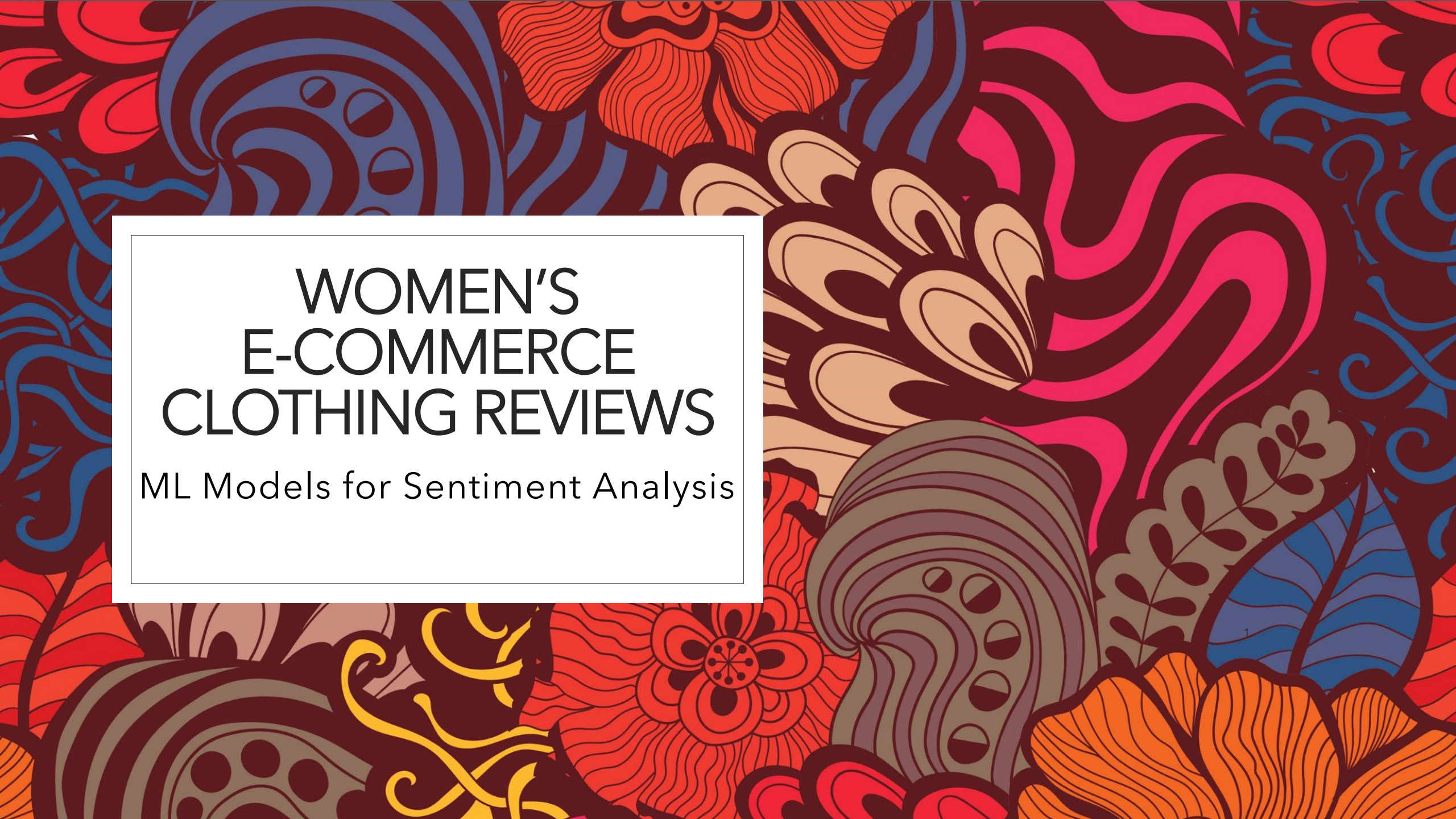

WOMEN'S
E-COMMERCE
CLOTHING REVIEWS
ML Models for Sentiment Analysis

Alfonso Toruno
Chris Campbell
Himani Manglik
Kamal Mukherjee
Shivani Thakkar

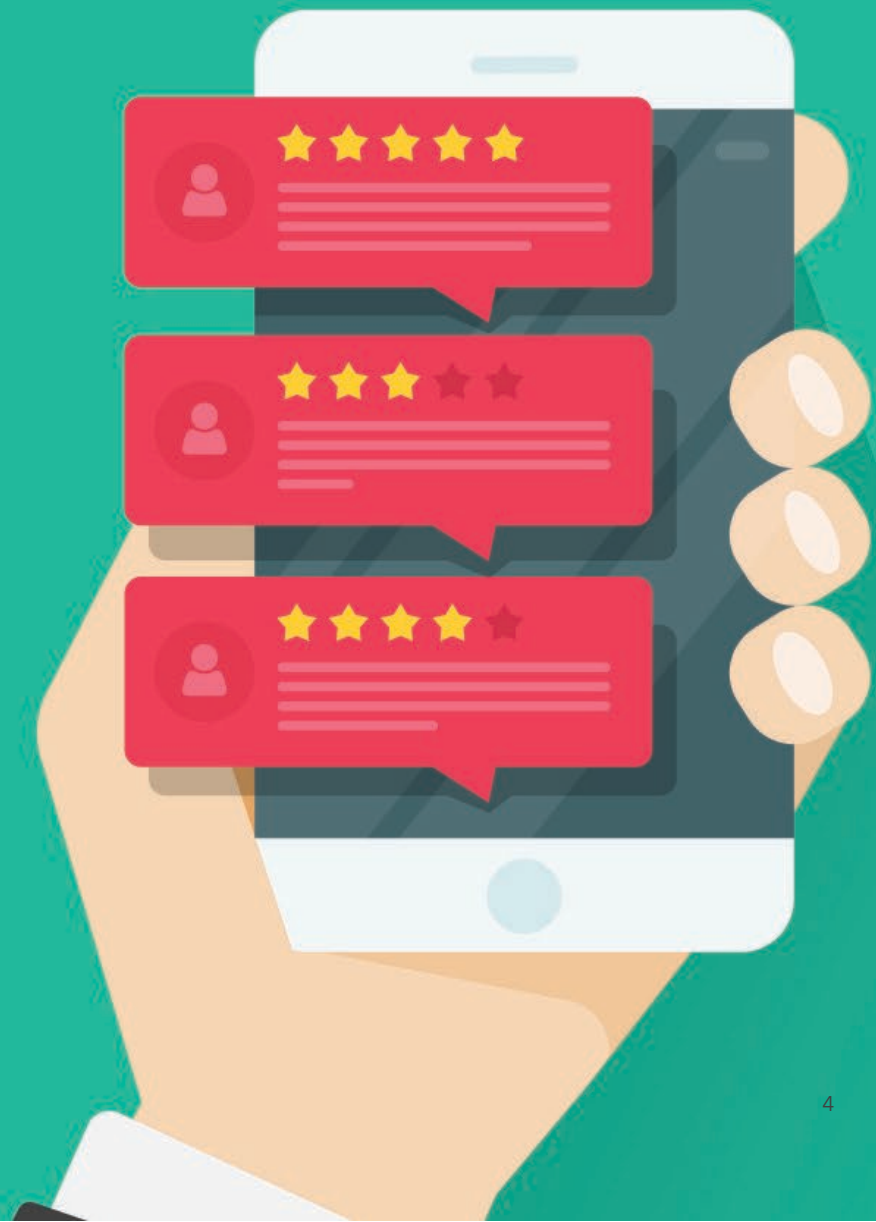
Why it Matters

89% of consumers worldwide make the effort to read reviews before buying products (Trustpilot, 2020), and almost half of all internet users worldwide say they post reviews about a product, company, or service each month (GlobalWebIndex, 2019).

Needless to say, companies looking to retain and grow their customer base must be able to interpret and act on the feedback they receive.

About Our Data

- Our dataset came from Kaggle and is a collection of women's clothing e-commerce reviews, written perspective buyers and actual customers.
- All references to retailers have been anonymized.
- The raw columns consist of the following:
  - Clothing ID
  - Reviewer Age
  - Review Title
  - Review Text
  - Rating
  - Recommended (binary values)
  - Positive Feedback Count
  - Division Name
  - Department Name
  - Class Name



Process Overview

ETL Process

- Raw Data Cleanse
  - Replaced null values with mode values
  - Transposed the dataframe
  - Calculated polarity score and sentiment
- Data Visualization
  - Plotly, PIL
  - Matplotlib, Seaborn
  - Python, Pandas

ML Analysis

- ML Toolkits
  - Wordcloud
  - NLTK
  - SciKit-Learn (SkLearn)
- ML Analysis Benchmark
  - Drop stop words
  - Tokenize & Lemmatize
  - Identify Positive and Negative words
  - Processing TD-IDF Matrix
- Train, Test, Predict (x5)
  - Logistic Regression
  - Support Vector Machines
  - Naïve Bayes
  - Random Forest
  - Neural Network

Data Cleaning

1. Drop & Average Null Values
   - Drop null 'Title' and 'Review Text' rows.
   - Replace null values in the Division, Department and Class Names columns with the mode value of each column.
2. Calculate polarity score off of the 'Title' text and assign a 'Positive,' 'Negative' or 'Neutral' sentiment accordingly.
3. Save and export the cleaned dataframe for further exploration.

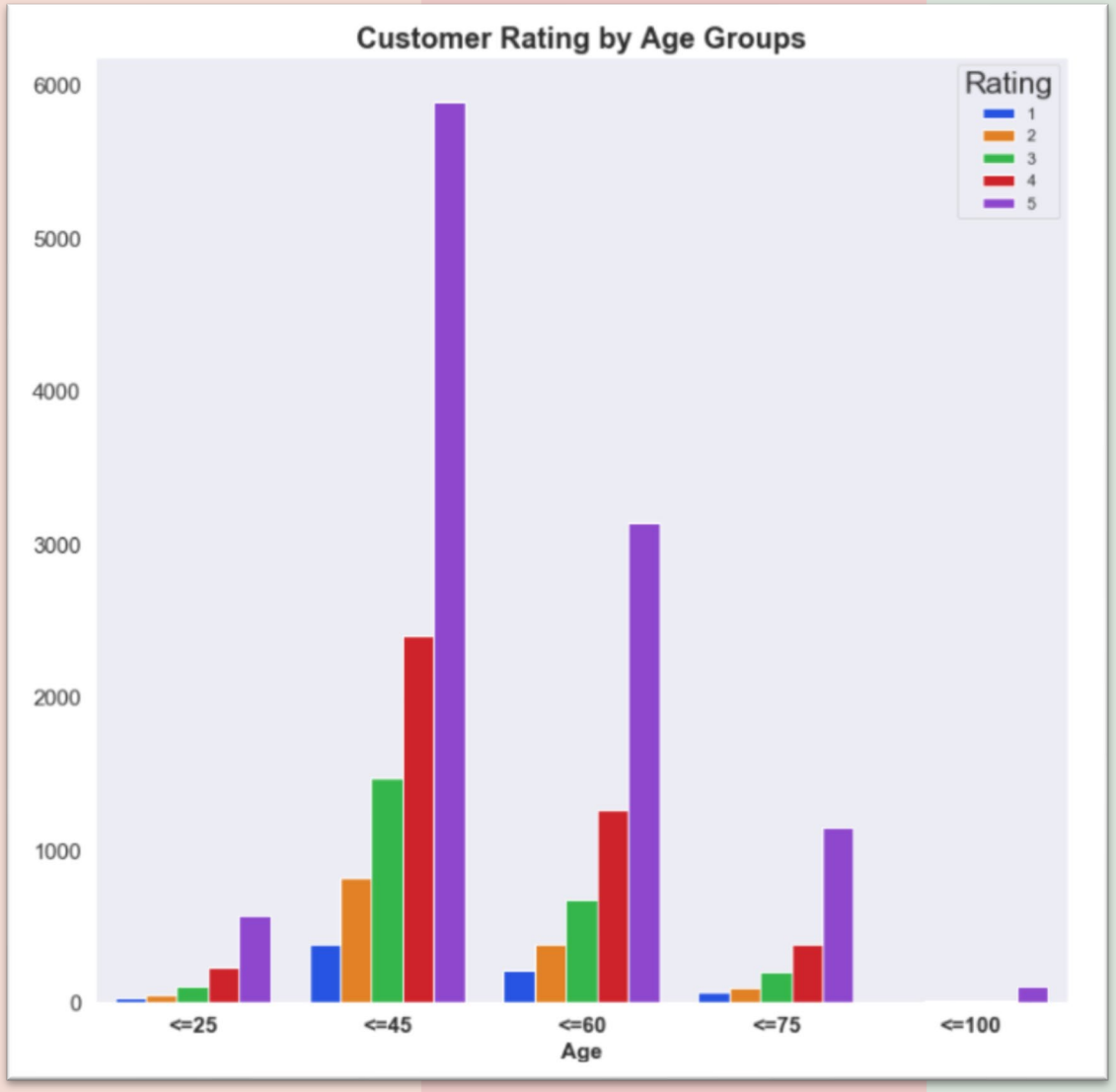
```
# cleaning up Title & Review Text Features
womens_clothing_data['Title'] = womens_clothing_data['Title'].replace('nan',np.NaN)
womens_clothing_data.dropna(subset=['Title'], how='all', inplace = True)
womens_clothing_data['Review Text'] = womens_clothing_data['Review Text'].replace('nan',np.NaN)
womens_clothing_data.dropna(subset=['Review Text'], how='all', inplace = True)

# grabbing the most popular value of important features
mode_division = womens_clothing_data['Division Name'].mode()[0]
mode_department = womens_clothing_data['Department Name'].mode()[0]
mode_class = womens_clothing_data['Class Name'].mode()[0]

# replacing NaN columns with mode value
womens_clothing_data['Division Name'] = womens_clothing_data['Division Name'].fillna(mode_division)
womens_clothing_data['Department Name'] = womens_clothing_data['Department Name'].fillna(mode_division)
womens_clothing_data['Class Name'] = womens_clothing_data['Class Name'].fillna(mode_division)
```

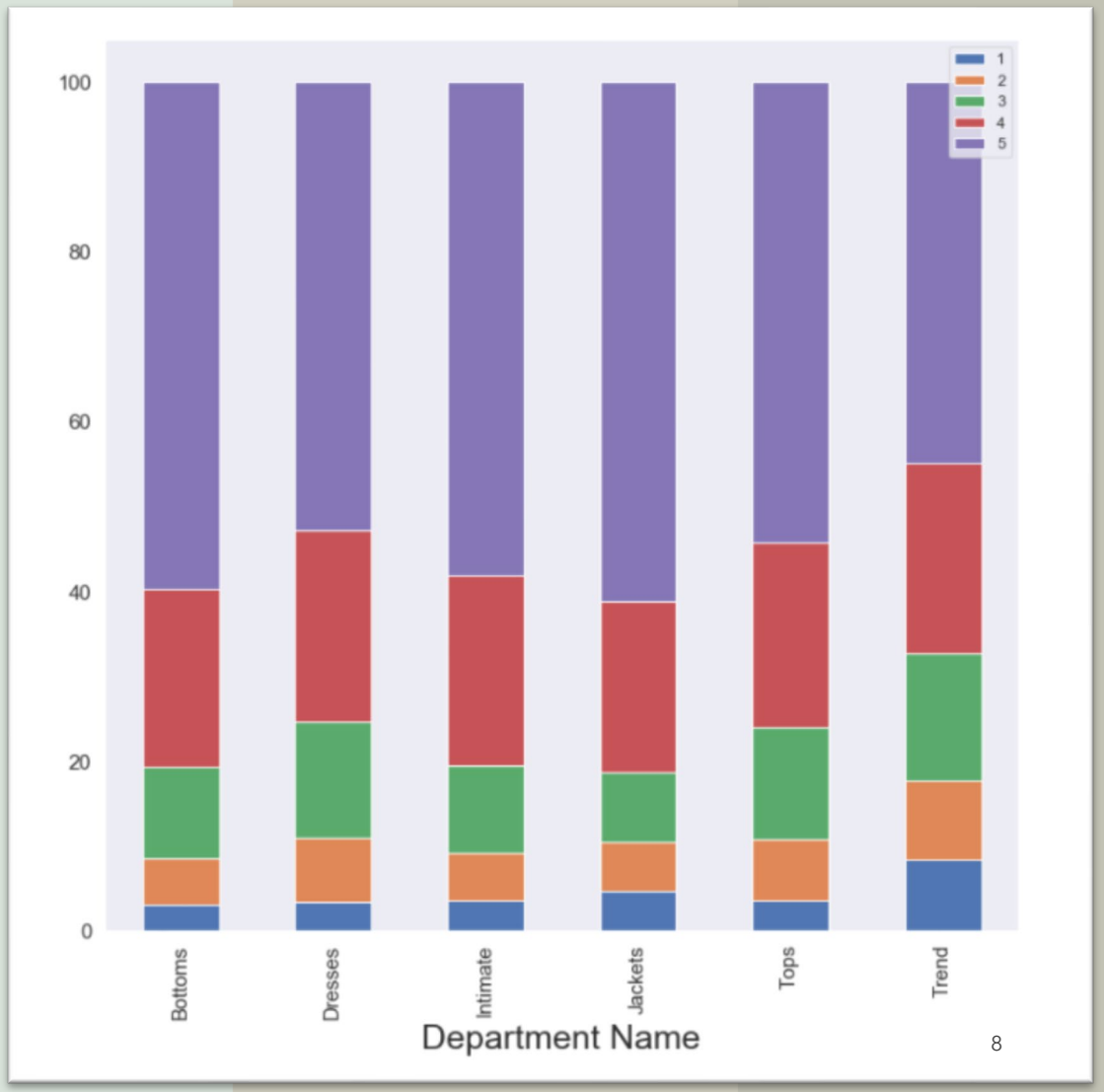
```
# calculating sentiment feature
SentAnalyzer = SentimentIntensityAnalyzer()
womens_clothing_data['Title'] = womens_clothing_data['Title'].astype(str)
womens_clothing_data['Polarity Score'] = womens_clothing_data['Title'].apply(lambda x:SentAnalyzer.polarity_scores(x)['compound'])
womens_clothing_data['Sentiment'] = ''
womens_clothing_data.loc[womens_clothing_data['Polarity Score'] > 0,'Sentiment'] = 'Positive'

womens_clothing_data.loc[womens_clothing_data['Polarity Score'] < 0,'Sentiment'] = 'Negative'
womens_clothing_data.loc[womens_clothing_data['Polarity Score'] == 0,'Sentiment'] = 'Neutral'
womens_clothing_data.head()
```

```
womens_clothing_data.to_csv("Womens_Clothing_Data_For_Tableau.csv", index=False, encoding='utf-8')
```

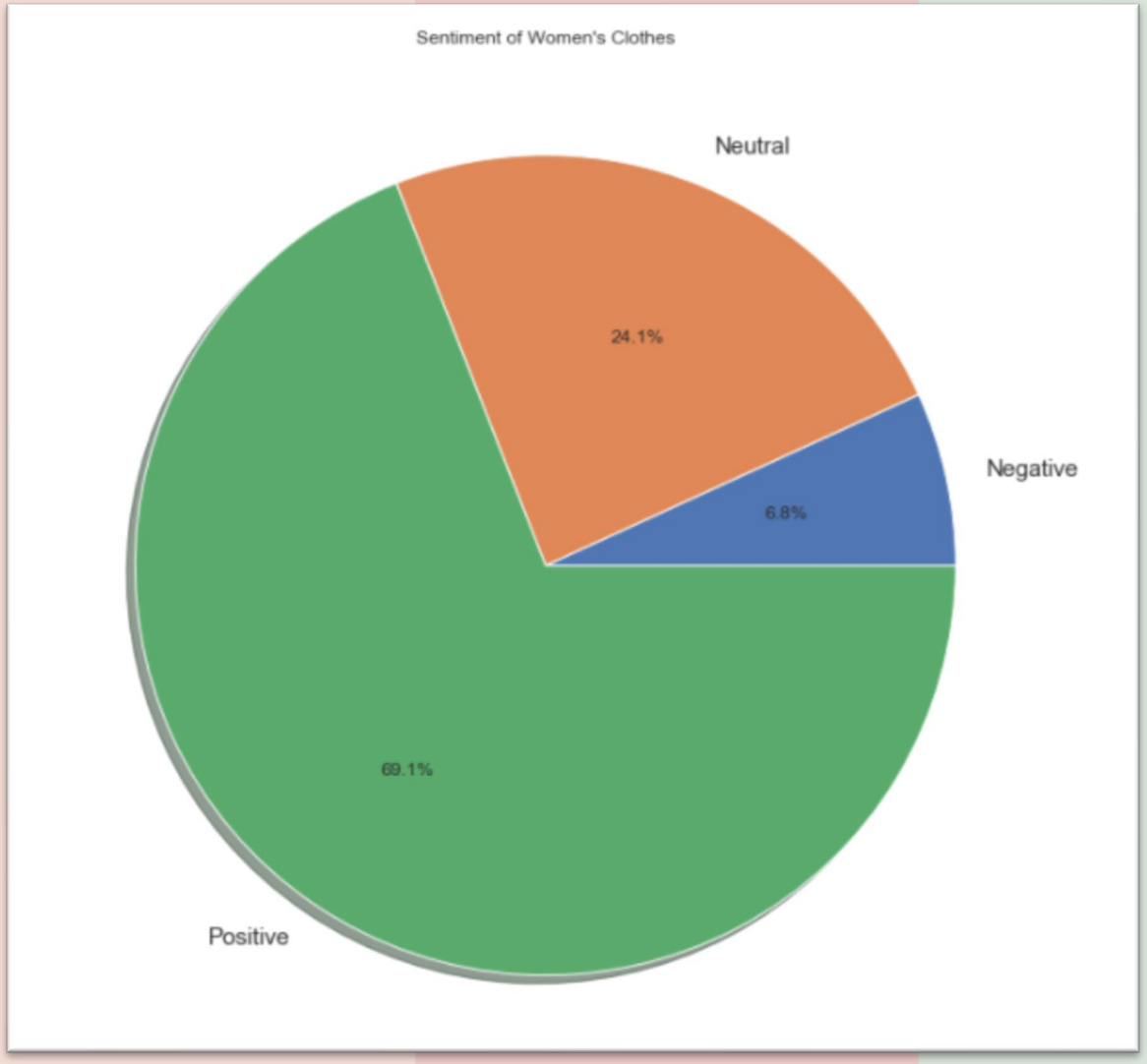
Data Visualizations

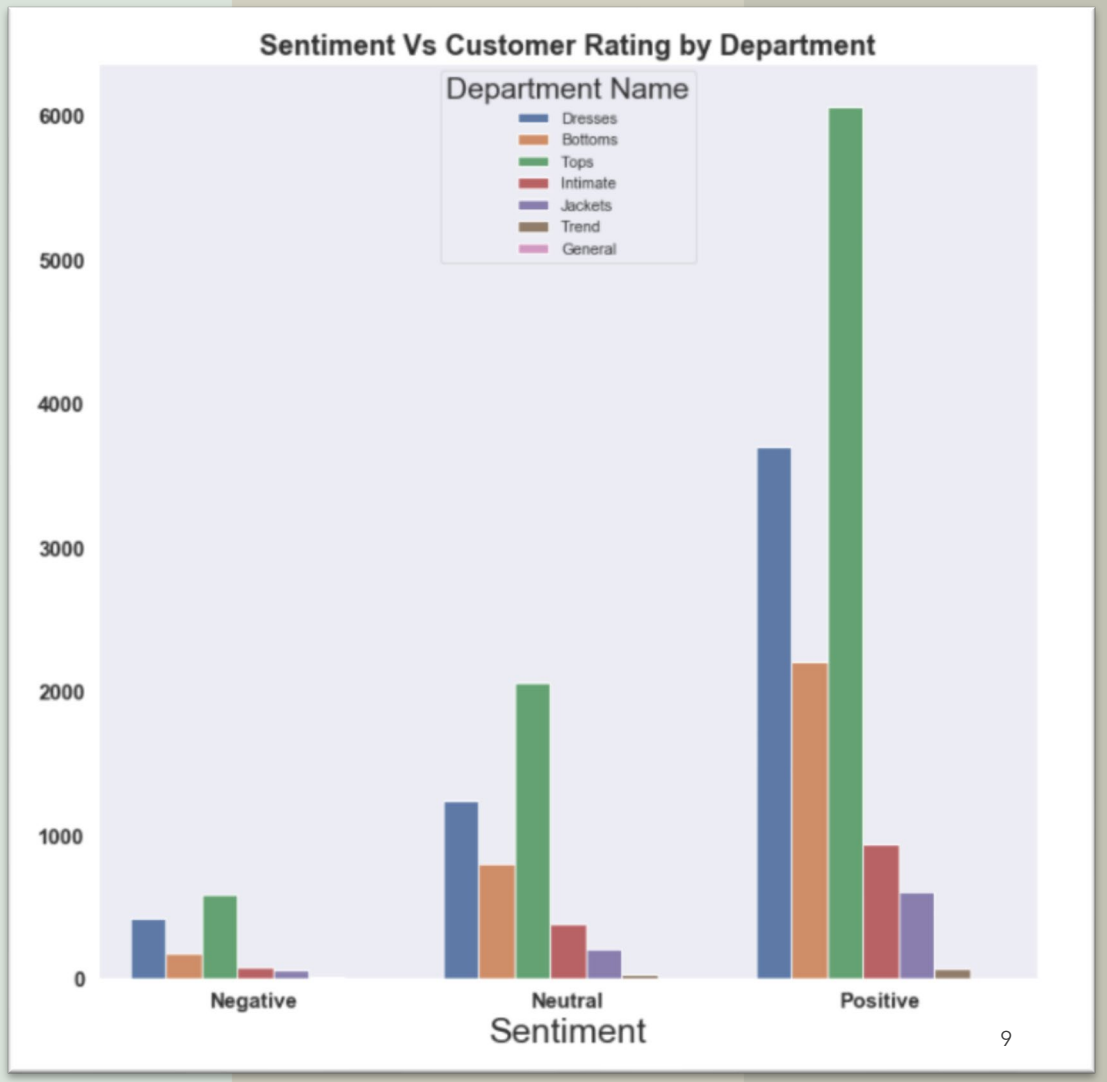
Ratings by Age Group & Department





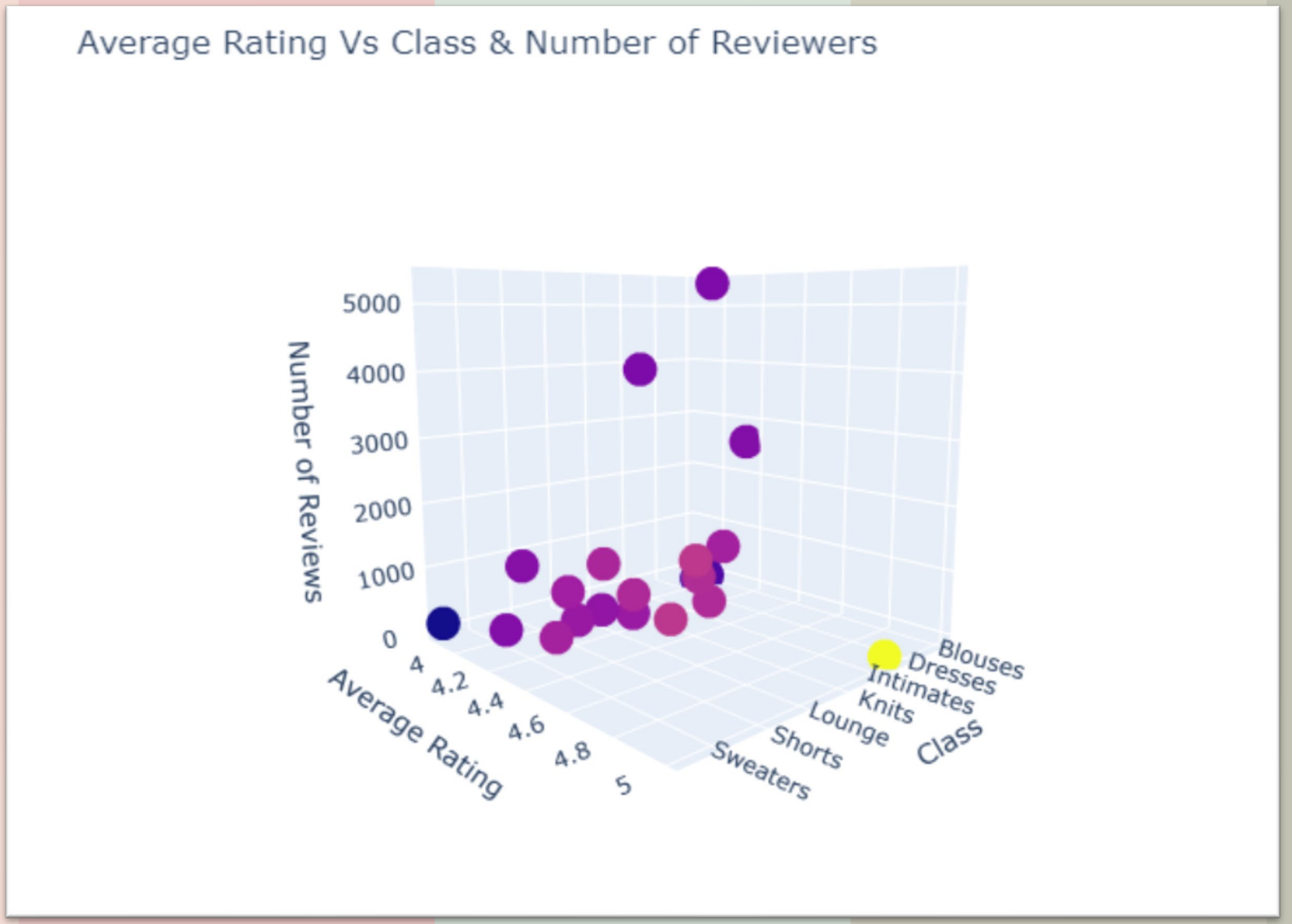
Sentiment Allocation & 'Sentiment v. Rating' by Dept.



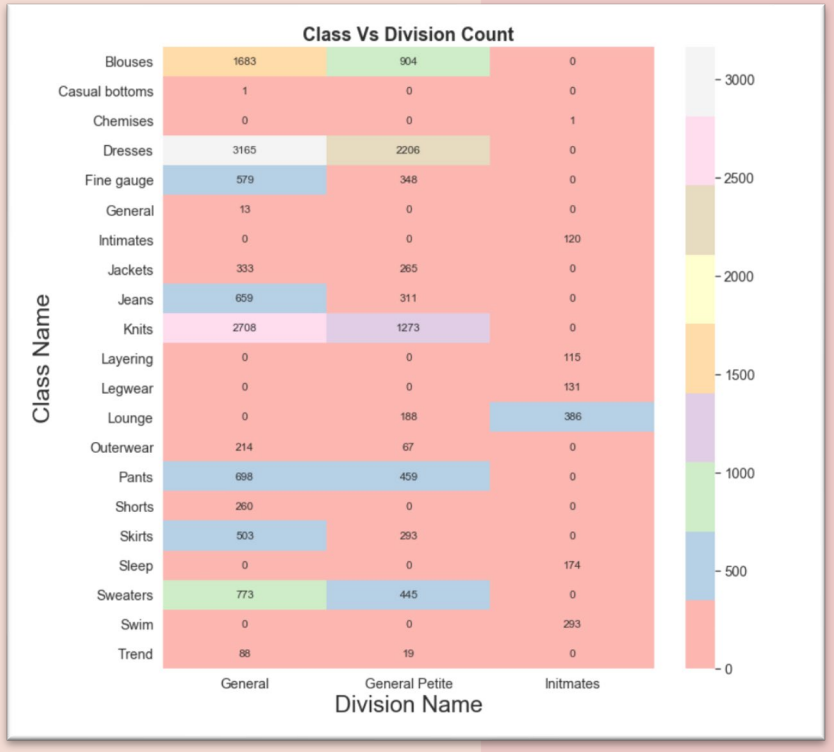


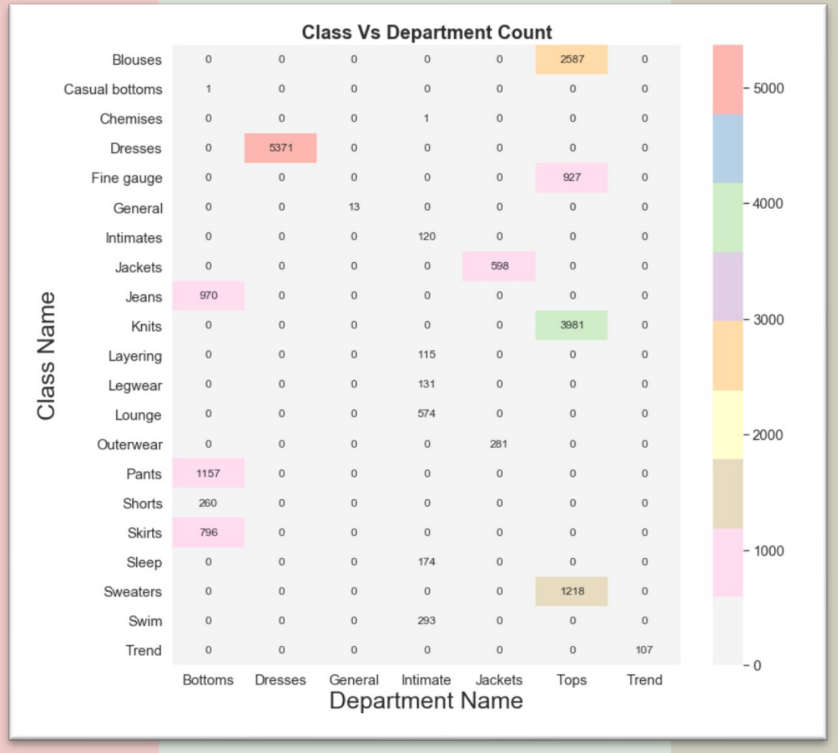
Relationship Between Rating, Class Name and Age

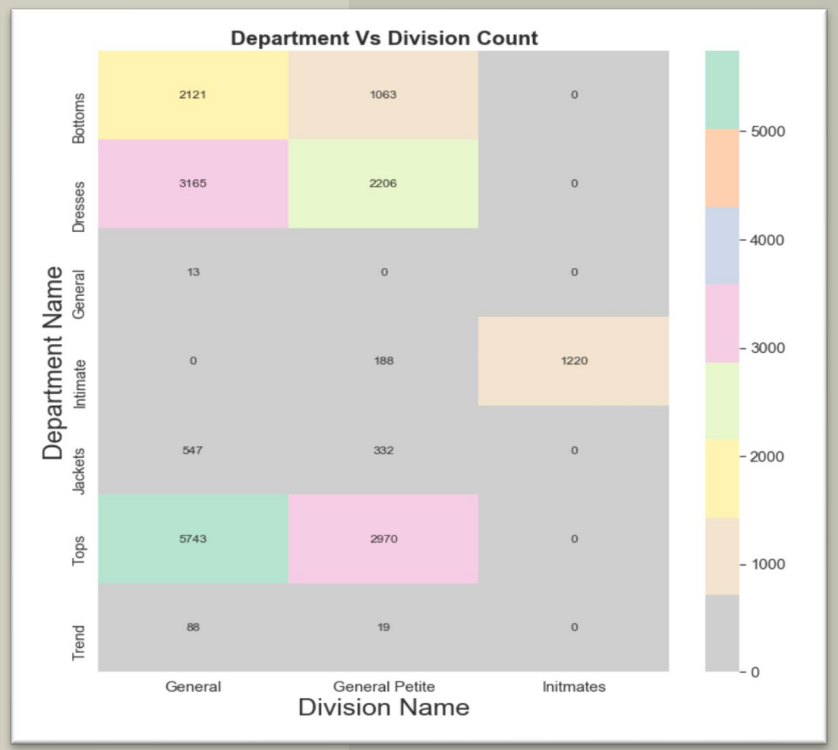
Average Rating v. Class & Number of Reviews



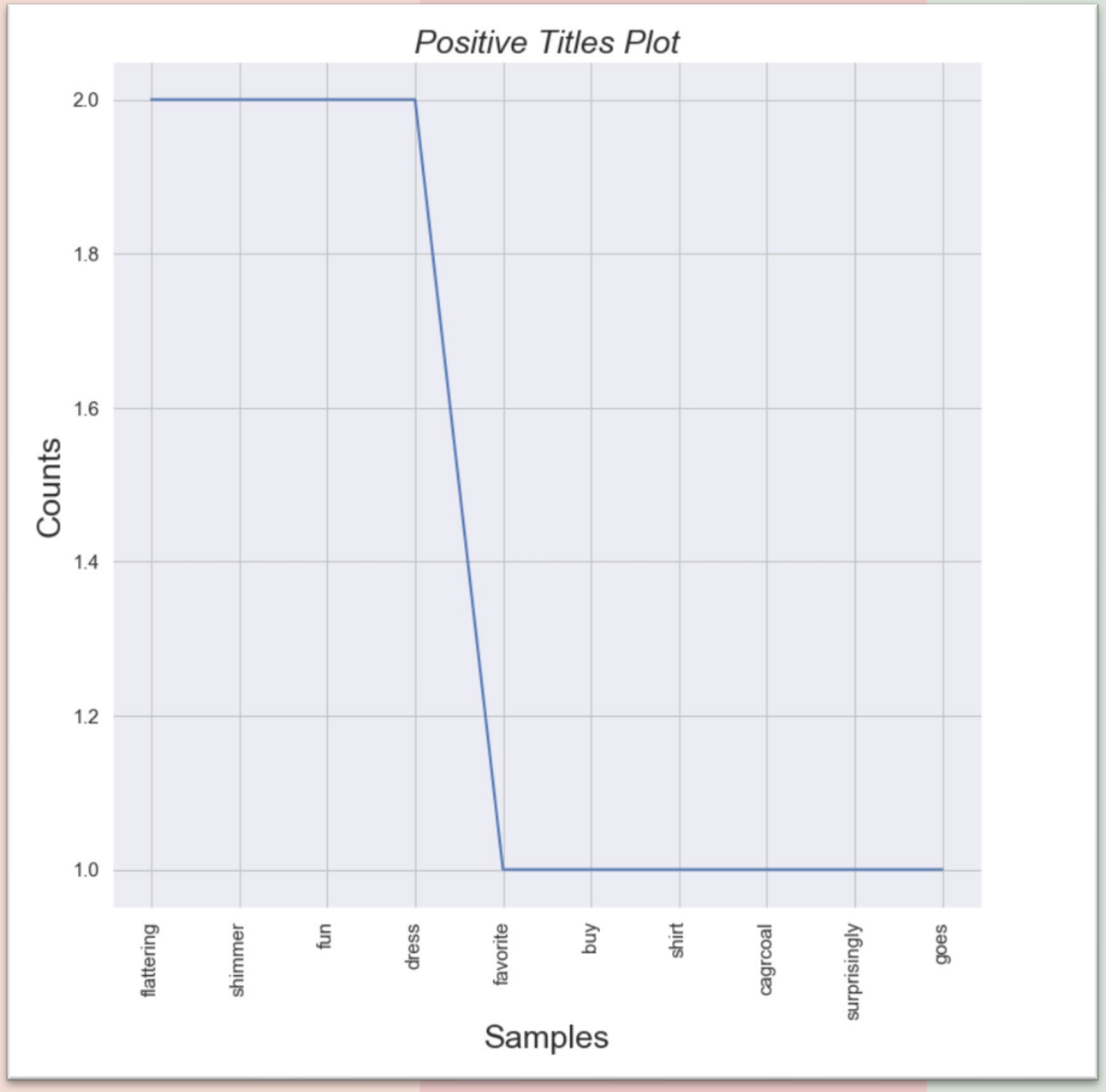
Heatmaps


Class Vs Division Count
Class Name
Blouses 1683 904 0
Casual bottoms 1 0 0
Chemises 0 0 1
Dresses 3165 2206 0
Fine gauge 579 348 0
General 13 0 0
Intimates 0 0 120
Jackets 333 265 0
Jeans 659 311 0
Knits 2708 1273 0
Layering 0 0 115
Legwear 0 0 131
Lounge 0 188 386
Outerwear 214 67 0
Pants 698 459 0
Shorts 260 0 0
Skirts 503 293 0
Sleep 0 0 174
Sweaters 773 445 0
Swim 0 0 293
Trend 88 19 0
General
General Petite
Initmates
Division Name
3000
2500
2000
1500
1000
500
0

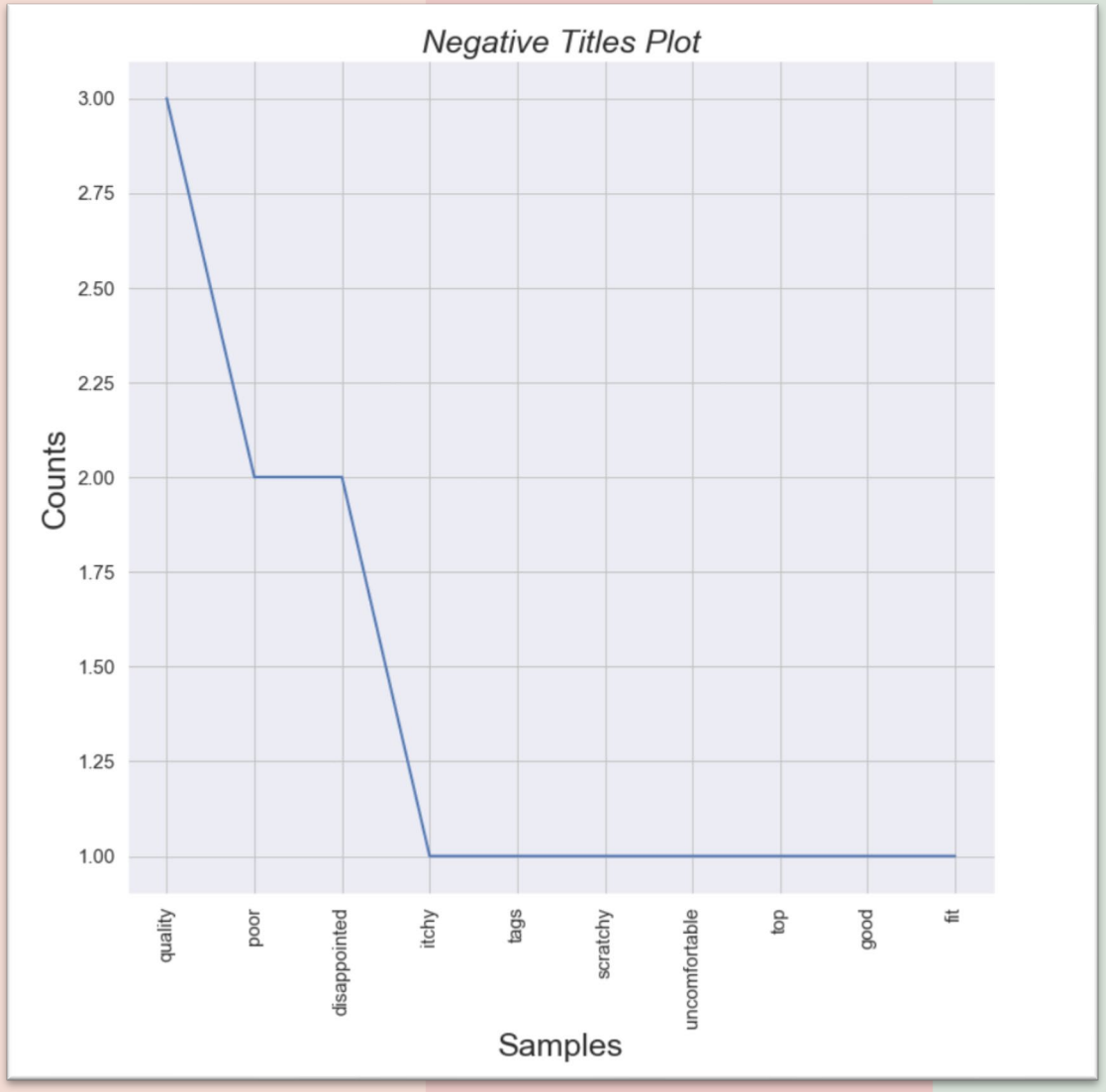

Class Vs Department Count
Class Name
Blouses 0 0 0 0 0 2587 0
Casual bottoms 1 0 0 0 0 0 0
Chemises 0 0 0 1 0 0 0
Dresses 0 5371 0 0 0 0 0
Fine gauge 0 0 0 0 0 927 0
General 0 0 13 0 0 0 0
Intimates 0 0 0 120 0 0 0
Jackets 0 0 0 0 598 0 0
Jeans 970 0 0 0 0 0 0
Knits 0 0 0 0 0 3981 0
Layering 0 0 0 115 0 0 0
Legwear 0 0 0 131 0 0 0
Lounge 0 0 0 574 0 0 0
Outerwear 0 0 0 0 281 0 0
Pants 1157 0 0 0 0 0 0
Shorts 260 0 0 0 0 0 0
Skirts 796 0 0 0 0 0 0
Sleep 0 0 0 174 0 0 0
Sweaters 0 0 0 0 0 1218 0
Swim 0 0 0 293 0 0 0
Trend 0 0 0 0 0 0 107
Bottoms
Dresses
General
Intimate
Jackets
Tops
Trend
Department Name
5000
4000
3000
2000
1000
0

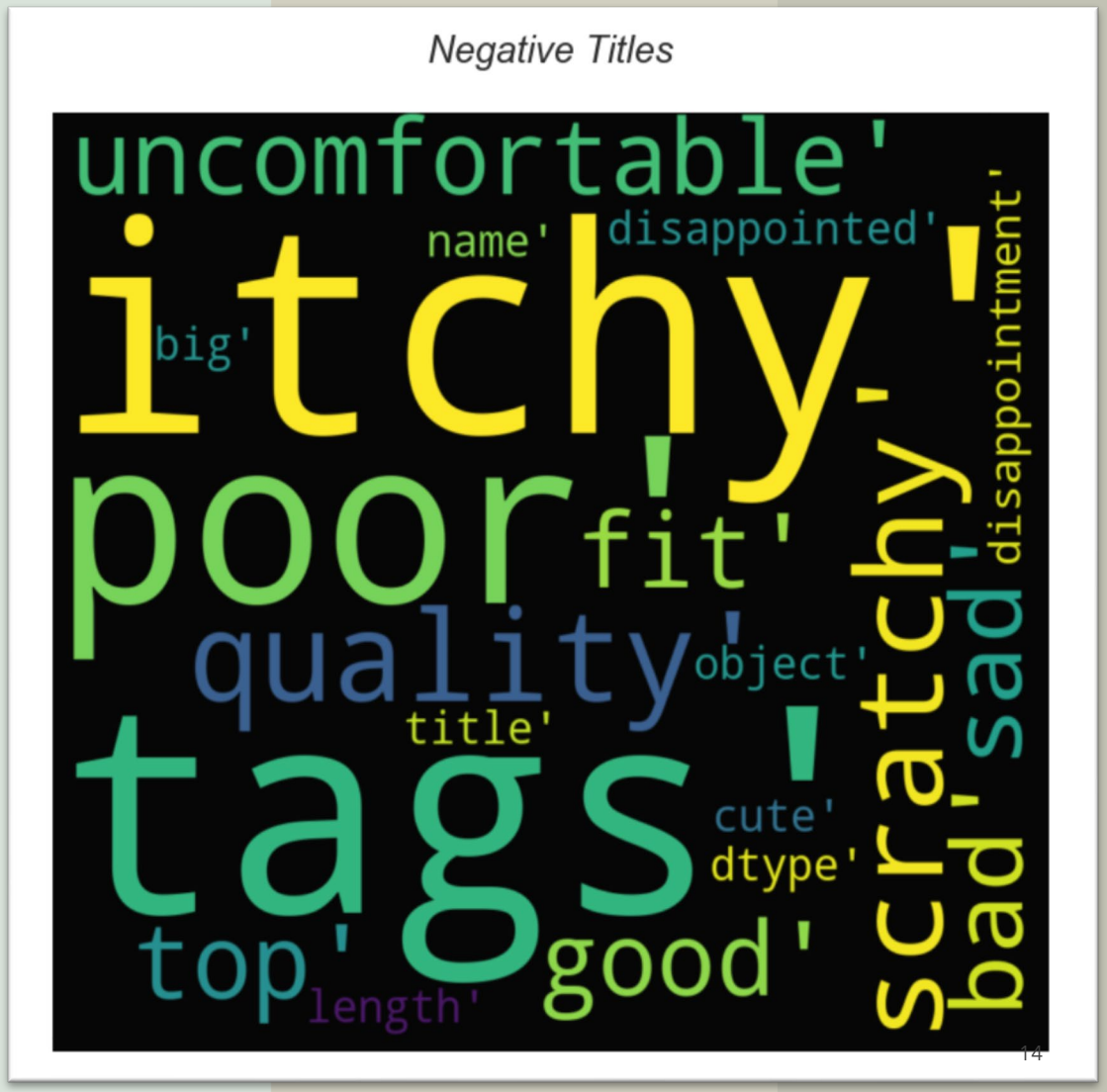

Department Vs Division Count
Department Name
Bottoms 2121 1063 0
Dresses 3165 2206 0
General 13 0 0
Intimate 0 188 1220
Jackets 547 332 0
Tops 5743 2970 0
Trend 88 19 0
General
General Petite
Initmates
Division Name
5000
4000
3000
2000
1000
0

Positive Title Plot & Wordcloud

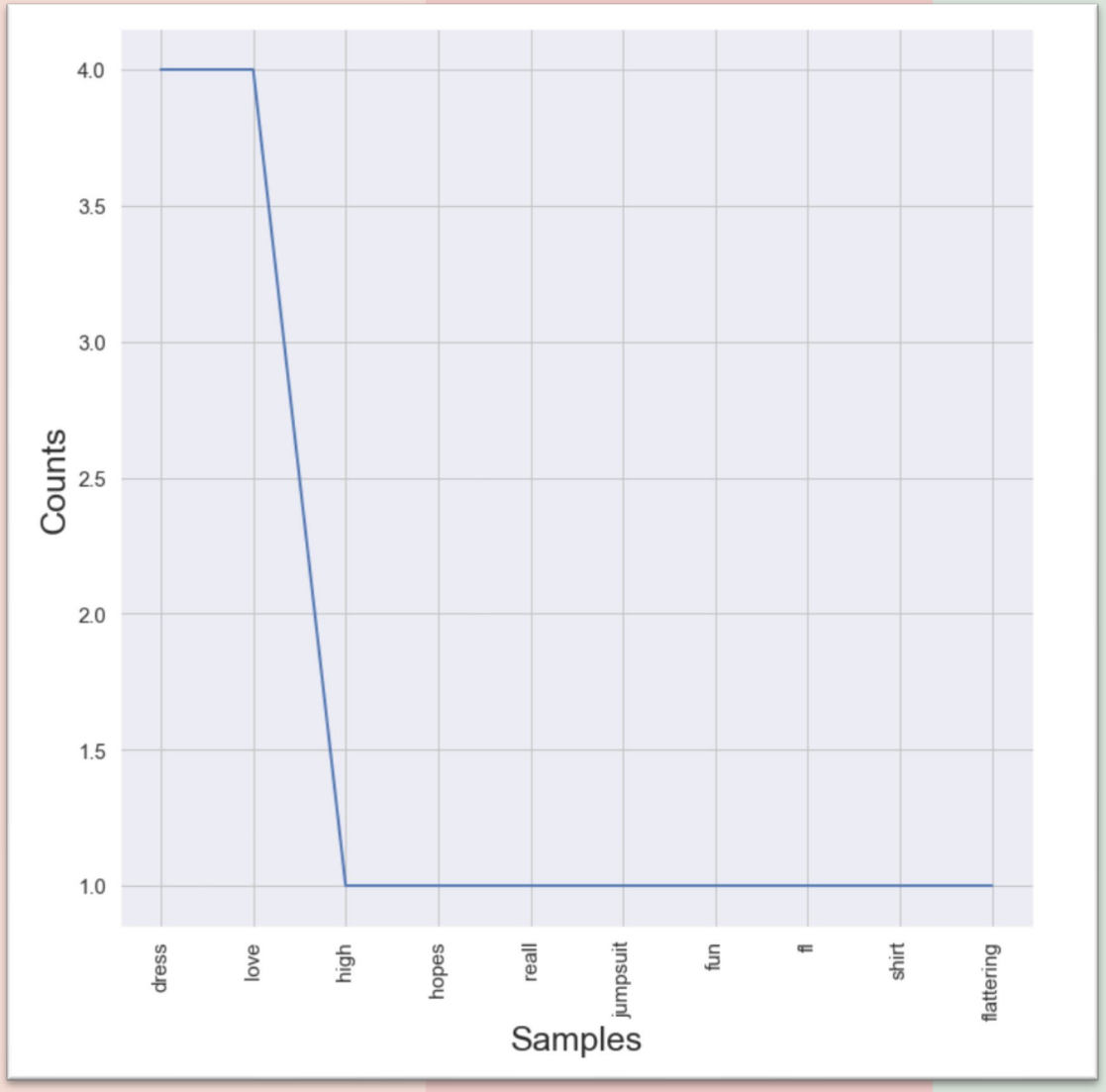


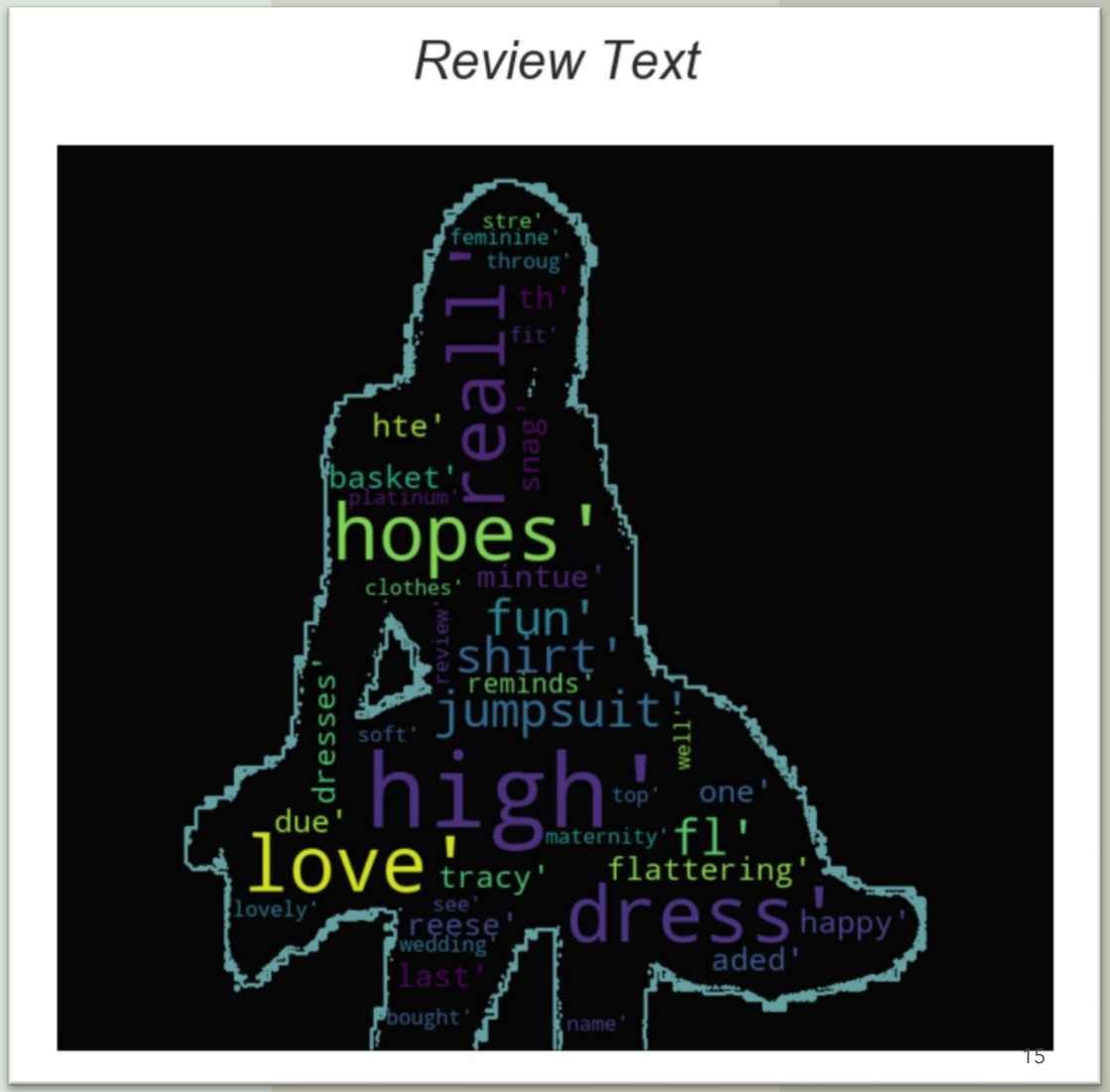
Negative Title Plot & Wordcloud





Review Text Plot & Wordcloud





We will feed our dataset into various ML models (e.g. Naive Bayes, Logistic Regression etc.) and compare which model can most accurately predict consumer sentiment. Based on the review text, we want the model to predict which customer is most likely to make a purchase.

ML Data Prep: 'preproc()' function

1. Drop stop words from the 'Review Text' column data.
2. Tokenize the data using the RegexpTockenizer().
3. Lemmatize the data.

The 'preproc()' function we created applies all three of the above steps to each customer's 'Review Text' and allows us to output a new 'revised_text' object.

The new 'revised_text' data, along with 'sentiment_class' data is then separated into training and testing buckets (at an 80-20% split)

```
# cleaning up review text
sw = set(stopwords.words('english'))

def preproc(data):
    #converting all to lowercase
    data = data.lower()
    #Tokenize
    words = RegexpTokenizer(r'[a-z]+').tokenize(data)
    #Deleting stopwords
    words = [w for w in words if not w in sw]


    #Lemmatizing
    for pos in [wordnet.NOUN, wordnet.VERB, wordnet.ADJ, wordnet.ADV]:
        words = [WordNetLemmatizer().lemmatize(x, pos) for x in words]
    return " ".join(words)
```

```
womens_clothing_data['Revised Text'] = womens_clothing_data['Review Text'].apply(preproc)
```

```
womens_clothing_data[['Revised Text', 'Title', 'Sentiment', 'Rating', 'Positive Feedback Count', 'Polarity Score']]
```

	Revised Text	Title	Sentiment	Rating	Positive Feedback Count	Polarity Score
2	high hope dress really want work initially ord...	Some major design flaws	Neutral	3	0	0.0000
3	love love love jumpsuit fun flirty fabulous ev...	My favorite buy!	Positive	5	0	0.5093
4	shirt flat due adjustable front tie perfect le...	Flattering shirt	Positive	5	6	0.3182
5	love tracy reese dress one petite foot tall us...	Not for the very petite	Neutral	2	4	0.0000
6	aded basket hte last mintue see would look lik...	Cagrcoal shimmer fun	Positive	5	1	0.5106
...	...	...	...	...	...	...
23481	happy snag dress great price easy slip flat cu...	Great dress for many occasions	Positive	5	0	0.6249
23482	remind maternity clothe soft stretchy shiny ma...	Wish it was made of cotton	Positive	3	0	0.4019
23483	fit well top see never would work glad able tr...	Cute, but see through	Positive	3	1	0.2500
23484	buy dress wed summer cute unfortunately fit pe...	Very cute dress, perfect for summer parties an...	Positive	3	2	0.8737
23485	dress lovely platinum feminine fit perfectly e...	Please make more like this one!	Positive	5	22	0.6581

ML Data Prep: TD-IDF Matrix

The last step before running our 'revised_text' data through each of the 5 ML models is to scale it based on feature vectors… in other words, identify and weigh the words found in multiple documents so that the model can give prominence to words with more meaning.

```
# extracting feature
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(encoding='latin-1')
X_train_counts = count_vect.fit_transform(train_revised_text)
X_train_counts.shape
```

```
(15740, 8701)
```

```
# testing token count with 'dress'
count_vect.vocabulary_.get('dress')
```

```
2278
```

```
# transforming count matrix to a normalized tf or tf-idf representation
# tf = term-frequency, if-idf = term-frequency times inverse document-frequency
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer(use_idf=True)
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

Machine Learning Outcomes: Train, Test & Predict (x5):

- Logistic Regression
- Support Vector Machines
- Naive Bayes
- Random Forest
- Neural Network

Naive Bayes Accuracy: 69.63%

```
# training our model
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train_tfidf, train_sentiment_class)
```

```
# testing our model
from sklearn.metrics import accuracy_score
X_test_counts = count_vect.transform(test_revised_text)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
predito = clf.predict(X_test_tfidf)

# calculate & print accuracy score
gaussian_acc = accuracy_score(test_sentiment_class, predito)
print(gaussian_acc)
```

```
0.6963151207115629
```

Random Forest Accuracy: 69.56%

```
# training our model
from sklearn.ensemble import RandomForestClassifier
ran = RandomForestClassifier(n_estimators=50)
ran.fit(X_train_tfidf, train_sentiment_class)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=50,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
# testing our model
X_test_counts = count_vect.transform(test_revised_text)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
predito = ran.predict(X_test_tfidf)

# calculate & print accuracy score
ran_acc = accuracy_score(test_sentiment_class, predito)
print(ran_acc)
```

```
0.6955527318932656
```

Support Vector Machine Accuracy: 69.71%

```
# training our model
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train_tfidf, train_sentiment_class)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
# testing our model
X_test_counts = count_vect.transform(test_revised_text)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
predito = svm.predict(X_test_tfidf)

# calculate & print accuracy score
svm_acc = accuracy_score(test_sentiment_class, predito)
print(svm_acc)
```

```
0.6970775095298603
```

Neural Network Accuracy: 64.37%

```
# training our model
from sklearn.neural_network import MLPClassifier
nn = MLPClassifier()
nn.fit(X_train_tfidf, train_sentiment_class)
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
# testing our model
X_test_counts = count_vect.transform(test_revised_text)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
predito = nn.predict(X_test_tfidf)

# calculate & print accuracy score
nn_acc = accuracy_score(test_sentiment_class, predito)
print(nn_acc)
```

```
0.643710292249047
```

Logistic Regression Accuracy: 69.86%

```
# training our model
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train_tfidf, train_sentiment_class)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
# testing our model
X_test_counts = count_vect.transform(test_revised_text)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
predito = lr.predict(X_test_tfidf)

# calculate & print accuracy score
lr_acc = accuracy_score(test_sentiment_class, predito)
print(lr_acc)
```

```
0.6986022871664549
```

Analysis

1. Millennials proved to be the most outspoken with their product reviews, followed by the Generation X. Interesting enough, the youngest age group, Generation Z (who allegedly spend most of their time of smartphones), contributed the least amount of reviews. Due to extremely low participation, we have omitted the "Silent Generation" (i.e. age 76 years or older).
2. The majority of the reviews in this dataset are incredibly positive (4 & 5), and we see that same trend across all age groups. This e-commerce retailer seem to offer a very well-rounded portfolio that is appealing to most age groups.
3. Looking at various visualizations between Department, Division and Class Name, consumers left the most reviews about categories like Tops, Dresses and Knits, while leaving far fewer about sleepwear, swimwear, casual bottoms etc.
4. Our Sentiment Analysis demonstrates that the consumers usually have a strong sentiment one way or the other and for this dataset, less than 25% of the reviewers expressed a neutral sentiment.
5. While our dataset does not provide any information on actual sales conversions, by looking at other consumer behavior datasets we can confidently assume that customers with positive sentiment are likely to make (or have made) an actual purchase.
6. We have also processed our dataset through five different machine learning models to project consumer's sentiment based on their review text. For our dataset Logistic Regression was the most accurate model and Neural Network was the least.

Next Steps

For our own learning and self-improvement we would like to continue to work on following areas beyond this course:

1. Implement Feature Engineering and Hyper Parameterization to fine tune our models.
2. Identify datasets that provide review information along with sales conversion and abandon cart information so that we can provide more accurate predictions.
3. Convert our Jupyter Notebook into a Python app with Bootstrap / ES6 frontend and deploy it on Heroku, AWS or similar platform.