NUS
National University of Singapore

School of Computing

TBA3102 TEXT ANALYTICS

Author: ANG KUO SHENG CLEMENT

GP04

A0177726R

a. executive summary of project.

This project harness topic modelling techniques by gaining insights from text reviews to improve better web content marketing. The intended audience(s) are for any e-Commerce merchant, web content (Product) developers and marketers pursuing web content marketing efforts to increase organic search engine optimization to improve lead acquisition to conversion. Topic models are extremely useful in summarizing large corpus of text reviews from customers to extract key concepts to create better target ads that highlight the strong selling points and e-Commerce merchants to enhance their web content for organic search optimization.

Topic models approach in this project is using feature engineering such as TD-IDF because Continuous Bag of Words (CBOW) model feature vectors are based on absolute term frequencies, there might be some terms that occur frequently across all documents and these may tend to overshadow other terms in the feature set in a large corpus. Another feature engineering model used is Non-negative Matrix Factorization (NMF) which is Linear-algebraic model that factors high-dimensional vectors into a low-dimensionality representation. In this project, the topic model was built by segregating cleaned raw corpus by different department(s) to ensure topic modelling stays relevant for their product category across each team. Based on the highest coherence score, LDA is the final chosen topic model against NMF and the coherence chart plot was used to identify the optimal number of topics. However, the final number of topics was further reduced until there are minimal repetition of keywords terms across the number of topics. The reason for reducing TOTAL_TOPICS to a minimal can be explained by business instinct to find feature terms for niche topic.

Affinity Propagation algorithm was used such that it identifies 3 distinct cluster labels & hierarchical dendrogram to give an idea of how the topic terms has been clustered across topic documents.

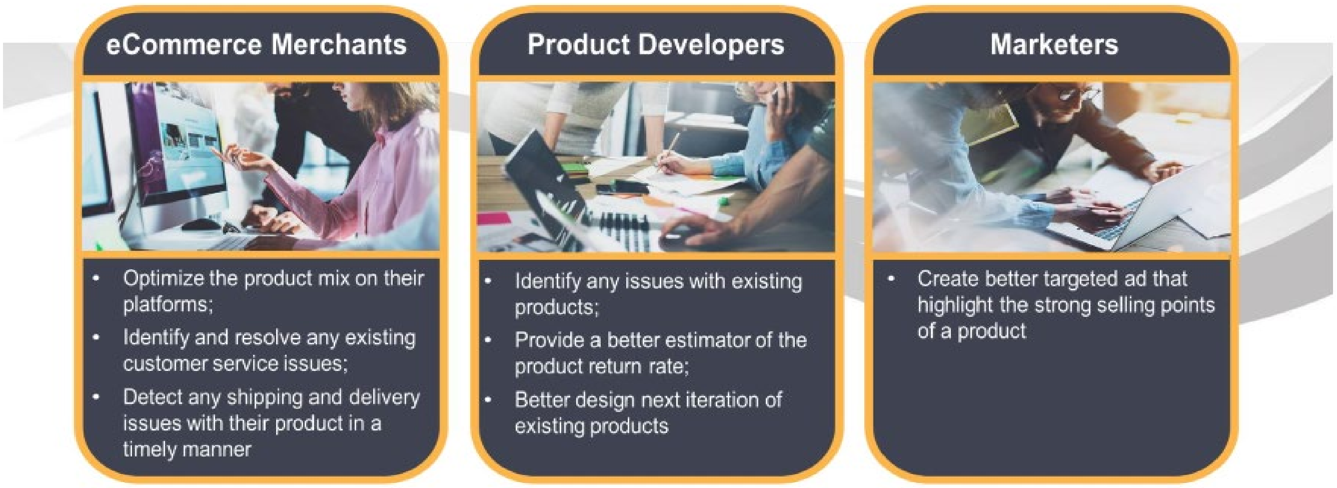
Machine learning model such as Multinominal Naive Bayes, Decision Tree Classifier, Linear SVM and ensemble learning Classifier AdaBoostClassifier, RandomForest Classifier was used to predict sentiments expressed in categorical form of “POSITIVE”, “NEUTRAL”, “NEGATIVE” based on the content of reviews & opinions. Linear SVM is the final chosen model after it has been evaluated in terms of accuracy, precision and recall, support in Classification Report & classification performance between Positive, Neutral and Negative presented in Confusion Matrix. Using unseen observations of Review _Text has also been conducted across all the machine learning Classifiers, LinearSVM was observed to predict sentiment polarity better than other ensemble, supervised & unsupervised machine learning algorithm. Unsupervised learning using lexicon-based model for predicting sentiment polarity, popular model such as SentiWordNet and VADER model were evaluated on their classification performance using Confusion Matrix and Classification Report. VADER model was selected as the final model with higher overall accuracy of 74% to predict sentiments whenever there are unlabeled ratings available.

Project Planning

Suppose your group is an analytics team in a typical real-world organisation. Your group may choose to work on any domain/industry. Propose an analytics project involving unstructured text data in collaboration with one or more other business or functional units in your organisation to solve a real-world business problem in the chosen domain/industry. Prepare a project plan for the Chief Analytics Officer to review, with the aim of getting his approval to initiate the project.

b. Intended audiences and their primary activities.

Based on customer's reviews extracted using web-mining techniques, the objective is to harness topic modelling techniques by identifying key topic terms from customer's reviews for better web content marketing. The intended audience(s) are for any e-Commerce merchant, web content (Product) developers to work in tandem with marketers to enhance their web content marketing efforts and search engine optimization to improve lead acquisition to conversion through topic modelling acquired from customer's text reviews.


eCommerce Merchants
Optimize the product mix on their platforms;
Identify and resolve any existing customer service issues;
Detect any shipping and delivery issues with their product in a timely manner
Product Developers
Identify any issues with existing products;
Provide a better estimator of the product return rate;
Better design next iteration of existing products
Marketers
Create better targeted ad that highlight the strong selling points of a product

By soliciting customer's reviews, we can garner key relevant topic terms to gain insights for marketers & web developers to create better content marketing for target ads. This project encompasses identifying suitable machine learning model that is capable of classifying sentiments based on unseen observation of text reviews or opinions after training & testing based on the existing dataset, in order for marketing & sales team to estimate future sales demand outlook.

c. Business question:

i. State the business question

E-commerce space is extremely competitive and current day consumers want personalized experiences, easy to access product information and exceptional services. In order to deal with high expectations from consumers, online retailers need to leverage customer sentiments or opinions data and actively integrate with marketing analytics using RFM analysis to improve their decision-making in predicting sales or product demand outlook. The purpose is to use the customer's review associated with each clothing category to improve marketing assortment planning for product listing & to improve customer experience by having better understanding what their grievances on product experience are. Based on customer's reviews found in the ecommerce portal to generate topics modelling & document clustering, in this way, we can know about what users are talking about, what they are focusing on, and perhaps where app developers should make progress at. This will couple with marketing analytics in identifying customer purchase patterns and opinions on particular products category, e-commerce retailers can target specific individuals or segments with personalized offers and discounts to boost sales and increase customer loyalty.

With the use of Topic-Based modelling, the marketer can review more vocabulary terms to better engage web content developer to enhance site content.

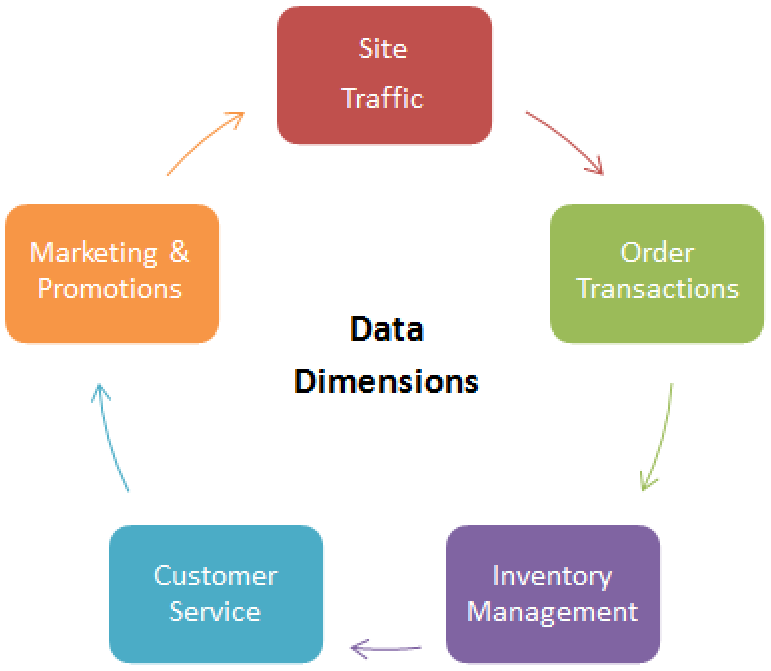
ii. Provide a description of the relevant business challenge/problem.

The online marketing & planning team typically relied on transactional history in online sales channel as the basis in making demand forecast to replenish their stock inventory. The issue of past sales history alone is not justifiable when it can also result in over-stocking and sales discount for clothing has a dire financial loss impact to the firm. Moreover, their online store is bombarded with customer's reviews that is hardly harness in allowing marketing team to learn a great deal about customer sentiment and preferences towards products or services. Marketing team has already been realizing their assortment planning based on customer purchase patterns but find it challenging to solicit & grasp customer's opinions on particular products and their categories. In this competitive E-commerce shopping experience, the marketing team realize the gap in their sales revenue and expenses incurred from e-commerce web content search engine optimization efforts & assortment planning. Hence, they are also unsure what other are avenues promising to enhance their existing content marketing development for each product category.

iii. Identify the affected business processes.

The affected business processes by this topic modelling includes understanding various aspect in data dimensions from website site traffic category management used by marketing and promotions, customer service, inventory management for product listing & assortment planning. This topic modelling is intended to reap significant benefit in strengthening web content marketing for search engine optimization (SEO). The strong argument for topic modeling is a key component of the Hummingbird algorithm to strengthen organic SEO in order to improve search engine page-ranking (SERP).

Content marketing is the best way to improve organic search engine optimization by posting relevant images built from topic modelling and web document clustering. By creating high quality content with images that's focused around targeted keywords and phrases using techniques from topic modelling and document clustering, it can improve visibility on the search engines with higher leads acquisition to conversion rate.



Despite prevalence in landing page conversion rate optimization, a split test can actually be applied to most situations in which we want to optimize performance derived from topic modelling and document clustering. The simplest split test is an A/B test, which tests 2 versions of something against one another using different web document content with different images. Of course, each site and their audience is different. The only way to find out what content online audience responds to best, is to split test it.

d. Data requirement:

i. **State the data that are required.**

The most critical data for this project is the Review Text to identify key topic phrase, which is complemented by data from other fields to provide relevancy & association to the target audience

Field	Description
Clothing ID	Unique identification number of the clothing
Age	Clothing Age
Title	Customer Feedback Summary Description
Review Text	Customer Review Details
Rating	Customer Rating of Products
Recommended IND	Customer's recommendation Indicator
Positive Feedback Count	Customer's positive feedback count
Division Name	Division associated with sales of Product
Department Name	Department associated with sales of Product
Class Name	Class name or category of product

ii. **Provide relevant business and operational definitions of the data.**

Product reviews are multifaceted, and hence the textual content of product reviews is an important determinant of consumers' choices as to how textual data can be used to learn consumers' relative preferences for different product features and also how text can be used for predictive modeling of future changes in sales. Both negative sentiment and positive sentiment of the reviews are individually significant predictors influence in predicting future sales or customer demand which is an important part of Supply Chain Management in managing finished goods inventory stockpoints.

Rating: Quantitative measure of customer's satisfaction level of the purchase transaction with respect to the Clothing ID.

Feedback Count: Quantitative measure on the number of customers providing their feedback from the survey made after their purchase transaction.

Review Text: Voice of Customer (VoC) or qualitative information of customer's satisfaction level with respect to the Clothing ID of the purchase transaction.

Product Reviews can be useful to explore unpromising avenues in ways to improve web content for new visitors & ads re-targetting to serve online visitors who have already visited any e-tailers website before.

iii. Data collection plan for existing and/or new data.

Data collection for existing data is performed using web mining techniques to retrieve customer's reviews, ratings, feedback_counts from e-commerce online store to form the existing data. The data solicitation plan for new data such as store's geographic location & regional/country sales revenue would be beneficial for performing market segmentation and sales revenue generation for different geography.

e. Data analysis plan:

i. Explain in detail the type of analysis work that will be undertaken in order to answer the business question.

The type of analysis to identify important domain keywords in woman clothing fashion encompass topic modelling and document clustering. Firstly, each document is preprocessed through tokenization, lemmatization, spelling checks, removal of html tags & special characters as well as deletion of unsuitable terms using stop word list. Hence, stop word list plays a very important role to basically exclude or remove common words that don't really add a lot of feature terms value desirable for web content and marketing objectives. If don't remove the word "the" from customer's review corpus for example, it will likely show up in every topic generated, such that there's no ways for me to define TdifVectorizor (min_df or max_df) in order to achieve features vocabulary with high relevancy to woman clothing fashion business terms.

Latent Dirichlet Allocation (LDA) is used to exploit hidden topics distribution information that are present in customer's review corpus. LDA is a Bayesian generative model that learns topic representation of each document and the words associated to each topic that best fit the corpus. This model is categorized as unsupervised learning, making the number and the content of topics difficult to control. However, supervised topic models such as discriminative LDA, L-LDA requires metadata of the document (such as labels) in order to match the topics to the labels in the document.

Term frequency-inverse topic frequency (TF-ITF) method is used for constructing a supervised topic model, in which the weight of each topic term has the ability to distinguish topics. Non-negative Matrix Factorization (NMF), is also used to find topics in text. The mathematical basis underpinning NMF is quite different from LDA. NMF sometimes produces more meaningful topics for smaller datasets.

Both algorithms are able to return the documents that belong to a topic in a corpus and the words that belong to a topic. LDA is based on probabilistic graphical modeling while NMF relies on linear algebra. Both algorithms take as input a bag of words matrix (i.e., each document represented as a row, with each columns containing the count of words in the corpus). The aim of each algorithm is then to produce 2 smaller matrices; a document to topic matrix and a word to topic matrix that when multiplied together reproduce the bag of words matrix with the lowest error.

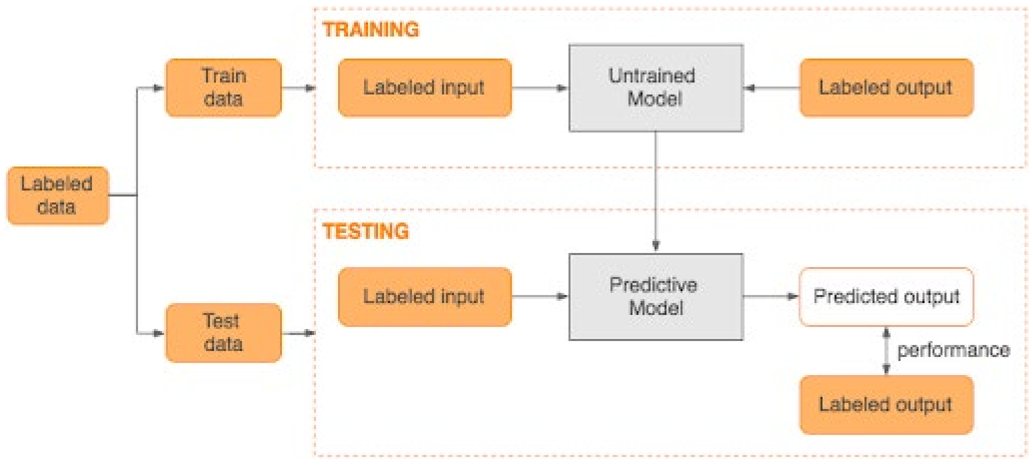
Both NMF and LDA are not able to automatically determine the number of topics and this must be specified to ensure each topic terms are unique after performing the topic modelling. This is because topic modelling is meaningless when there's repetition of words terms in a few other topics. For example, number of topics defined in LatentDirichletAllocation(n_components = 10) is also dependent on the amount of vocabulary term generated from TfidfVectorizer, depending on value of this parameter min_df and max_df.

ii. **Provide the details for the prediction involved.**

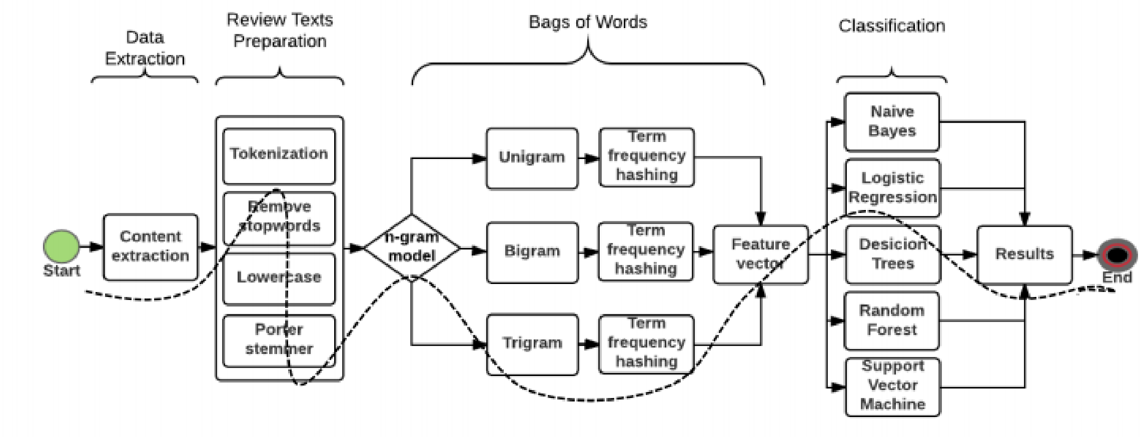
The typical prediction involved in the e-commerce space retailer or merchants comprises of customer retention and churn model based on implicit feedbacks such as past purchase transactions recency, frequency, monetary_value and explicit feedbacks such as text reviews and satisfaction rating level. For any e-tailers, this kind of prediction capability is also extremely important in order to manage the supply chain efficiently as well as ensure customer satisfaction, as such this project investigates the efficacy of various modeling techniques, namely, regression analysis, decision-tree analysis. The scope of this project include using text reviews from explicit feedbacks to predict satisfaction level or sentiments polarity expressed in categorical scale of "POSITIVE", "NEUTRAL" and "NEGATIVE".

Satisfaction rating level extracted from the web mining process is firstly converted from ordinal scale of 1-5 into categorical format as "POSITIVE", "NEUTRAL" and "NEGATIVE" before it is split into training and test data for predicting sentiments using various supervised, unsupervised and ensemble machine learning algorithm within classification model.

All reviews with rating 1 and 2 are marked as 'negative' reviews, reviews with rating 3 is marked as 'neutral' and reviews with rating 4 and 5 are marked as 'positive'.



The machine learning models that will be evaluated include traditional supervised multinomial Naïve Bayes models, Support Vector Machines (SVM), unsupervised machine model include DecisionTree with gini criterion, and Ensemble model such as AdaBoost and RandomForest for predicting sentiment polarity.



Workflow model for review processing

Leverage on pipeline module from Scikit-Learn to build this machine learning pipeline which combines CountVectorizer, TfidTransformer and various supervised, unsupervised and ensemble machine model(s) as mentioned earlier.

```
from sklearn.pipeline import Pipeline
clf_linearSVC_pipe = Pipeline([("vect", CountVectorizer()), ("tfidf",
TfidfTransformer()), ("clf_linearSVC", LinearSVC())])
```

GridSearchCV was also used for tuning the hyper-parameters of an estimator for each of the supervised, unsupervised and ensemble machine model.

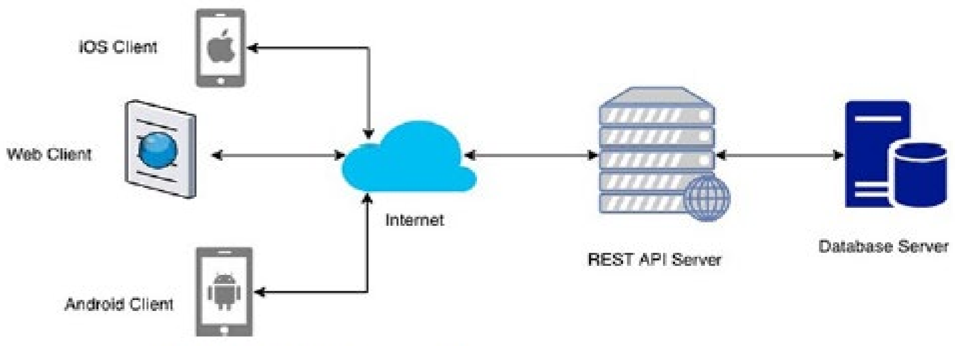
```
from sklearn.model_selection import GridSearchCV
parameters = {'vect__ngram_range': [(1, 1), (1, 2), (1, 3), (1, 4)],
             'tfidf__use_idf': (True, False) }
gs_clf_LinearSVC_pipe = GridSearchCV(clf_linearSVC_pipe, parameters, n_jobs=-1)
gs_clf_LinearSVC_pipe = gs_clf_LinearSVC_pipe.fit(X_train, Sentiment_train)
```

f. Type and format of deliverables:

i. State what are the reporting, dashboarding and modelling output that would be created from the data analysis work.

The reporting analysis is viewable on web browser, which is supported by frameworks, including Flask, Tornado, Pyramid, and Django. The web-based dashboards is supported by Bokeh which supports interactive data visualization. Bokeh allows developer to build Powerful Data Applications with a wide array of widgets, plot tools, and UI events that can trigger real Python callbacks, the Bokeh server is the bridge that lets apps developer connect these tools to rich, interactive visualizations in the browser. Alternatively, building of dynamic, interactive data visualizations on web browser can be achieved leveraging on D3.js which is a Javascript library.

Deploying Machine Learning output for this topic modelling will be delivered as a REST (representational state transfer) RESTful web services using Flask micro-framework in order to expose it to external users. This allows them to use the model output or prediction without having to access the underlying model by making use of Flask to deploy the model as a RESTful service which can be integrated with MySQL database server.



REST architecture diagram

```
from flask import (
    Flask,
    render_template
)

# Create the application instance
app = Flask(__name__, template_folder="templates")

# Create a URL route in our application for "/"
@app.route('/')
def home():
    """
    This function just responds to the browser ULR
    localhost:5000/

    :return:        the rendered template 'home.html'
    """
    return render_template('topic_modelling.html')

       if __name__ == '__main__':
       app.run(debug=True)
```

ii. List the frequency and periods when the analysis should be delivered.

The analysis presented on web dashboard are on weekly periodic interval since there shouldn't be much changes on daily basis. Customer reviews text corpus & ratings collated through web mining techniques before performing topic modelling on weekly basis should be adequate given the scale of customer base. For small customer base, the text corpus from customer's review could be performed on monthly period for topic modelling analysis.

g. Outline the initial presentation plan as well as the review and follow-up process.

This topic modelling project will initially be presentable by hosting on Github or https://colab.research.google.com/ with each subsequent review and development enhancement changes on the prediction modelling progressing from Linear SVM (Support Vector Machine) to Deep Learning which overcome bag-of-words limitation of loses ordering of words and doesn't consider the semantic relationship between words.

The use of traditional supervised machine learning models such as Linear SVM to perform sentiment analysis on training dataset consisting of ratings that are already pre-labelled with the sentiments is good to have but not essential. This is because in unsupervised learning Lexicon-Based Model, it can determine the sentiments even when there is no labeled sentiment data (ie: rating). Without the presence of label data, we consider the unsupervised lexicon-based models that can classify the exact sentiment polarity of a document considering other factors like presence of negation parameters, surrounding words, overall context, phrases before aggregating overall sentiment polarity scores to decide the final sentiment score. Unsupervised sentiment analysis models have detailed information pertaining to subjective words, phrases including sentiment, mood, polarity, objectivity, subjectivity that should be evaluated to compare against traditional supervised machine learning model such as LinearSVM.

The follow-up process includes implementing Convolution Neural Network because of its efficiency in representing words in vector space by predicting the current word given context and by predicting surrounding words given current word. The sentiment analysis prediction modelling development version will then be released to production use as a business intelligence dashboard or platform to obtain precise prediction of product demand integrating it with Marketing Analytics using RFM (Recency, Frequency, Monetary Value) analysis.

Project Implementation

Suppose your group's project plan has been approved by the Chief Analytics Officer. Implement the project and document the tasks and results for each of the following steps in the text analytics process:

a. Construction of text corpus – Include the crawling of data from the Internet (if applicable) and ingestion of data into a suitable data storage engine such as a relational database or csv file.

The process of extracting customer's reviews and other information as shown in (d) is performed using Python with Beautiful soup, request and Selenium package libraries to scrap content of the web documents. Requests for fetching web pages and BeautifulSoup for parsing HTML pages. For customer's text review or comments, comments are embedded as an iframe element by JavaScript. In order to harvest the comments, we will need to automate the browser and interact with the DOM interactively. One of the best tools for dynamic scraping is Selenium.

Web content mining can be harness to support a web enabled electronic business to improve on marketing, customer support and sales operations. The combination of Beautiful Soup and Selenium will do the job of dynamic scraping. Beautiful Soup is a very powerful library that makes web scraping by traversing the DOM (document object model) of a web page. Selenium automates web browser interaction from python when the data is rendered by JavaScript links can be made available by automating the button clicks with Selenium and then can be extracted by Beautiful Soup

Selenium setup:

```
from selenium import webdriver
       driver_exe = 'C:/Users/kuosh/Downloads/chromedriver_win32/chromedriver.exe'
        driver = webdriver.Chrome(driver_exe)
        driver.implicitly_wait(2)
        driver.get(product_url)
```

Web Scraping Setup:

```
import requests
from bs4 import BeautifulSoup
```

```
user_agent = '( Mozilla/5.0 Intel Mac OS X 10_8_2 AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/77.0.3865.90 Safari/537.36'
headers = {'User-Agent': user_agent}
html_soup = BeautifulSoup(r.text, 'html.parser')
    for amazon in html_soup.find_all('div', class_="sg-row"):
        links = amazon.find_all('a', class_='a-link-normal a-text-normal')
        for link in links:
            product_url = link['href']
```

b. Text preprocessing – Understand and clean the text data.

The Data pre-processing comprises of these steps:

- ❒ Tokenization: Split the text into words. Lowercase all words so that it aggregate raw text corpus into similar or common words which is useful when processing the subsequent stopword list and spelling-checker.
- ❒ Words that have fewer than 2 characters are removed since it does not provide much value for topic modelling or features term.
- ❒ Build stopwords list to remove common vocabulary and English words that does not provide any value for features engineering related to context of woman clothing or fashion.
- ❒ Use SpellChecker instead of Textblob. Spellchecker has been tested and verified by myself that it has better & higher accuracy than Textblob which often mis-corrected. Spelling-checker is important to correct and remove unwanted errors.
- ❒ Remove any digit or numeric characters since it has little value or relevance to topic modelling & bring any insights to marketing business context.
- ❒ Remove html tag characters since it has little value or relevance to topic modelling & bring any insights to marketing business context. HTML tags, which do not add much value when analyzing text for topic modelling, hence this function helps to remove such unnecessary content.

- Removing accented characters: In text corpus that is dealt with in this woman's clothing review in English Language, there are accented characters/letters that needs to be converted and standardized into ASCII characters. A simple example is converting é to e

c. **Feature engineering – Represent the unstructured text data in a structured format using various feature engineering models.**

There are some potential problems that might arise with Bag of Words (BOW) model when it is used on large corpora of text data. Since BOW feature vectors are based on absolute term frequencies, there might be some terms that occur frequently across all documents and these may tend to overshadow other terms in the feature set. For words that don't occur as frequently, it might be more interesting and effective to use combination of two metrics, term frequency (tf) and inverse document frequency (idf) to identify as features relevant to specific business domain arena (ie: fashion clothing).
"Bag-of-words was popular for many years but it has two major flaws as it loses ordering of words and doesn't consider the semantic relationship between words. For example, words such as "bad", "worst", and "Las Vegas" are equally distant despite the fact that "bad" should be semantically closer to "worst" than "Las Vegas". Although bag-of-n-grams somewhat considers word order in short context, it suffers from data sparsity and high dimensionality" [1].

Inverse document frequency denoted by idf is the inverse of the document frequency for each term and is computed by dividing the total number of documents in our corpus by the document frequency for each term and then applying logarithmic scaling to the result.

Before fitting into the NMF & LDA model, term frequency-inverse document frequency (tf-idf) features was extracted from the data sets with the function call, TfidfVectorizer. This function converts a collection of documents to a matrix of tf-idf features.

```
# Build feature matrix for a unigram for a cleaned normalized corpus

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(analyzer='word', min_df=0.1, max_df=0.8,
smooth_idf=True)
Bottom_tfidf_features = vectorizer.fit_transform(Bottoms_df['cleaned_words'])
```

```
Bottom_vocabulary = np.array(vectorizer.get_feature_names())
```

Depending on the size of the vocabulary that is required for business objectives in the respective departments (ie: Bottoms, Intimate, Top, Dresses etc.), TfidfVectorizer min_df or max_df shall be adjusted accordingly. For this project, min_df was set to 0.07 and approximately 50 vocabulary words are generated.

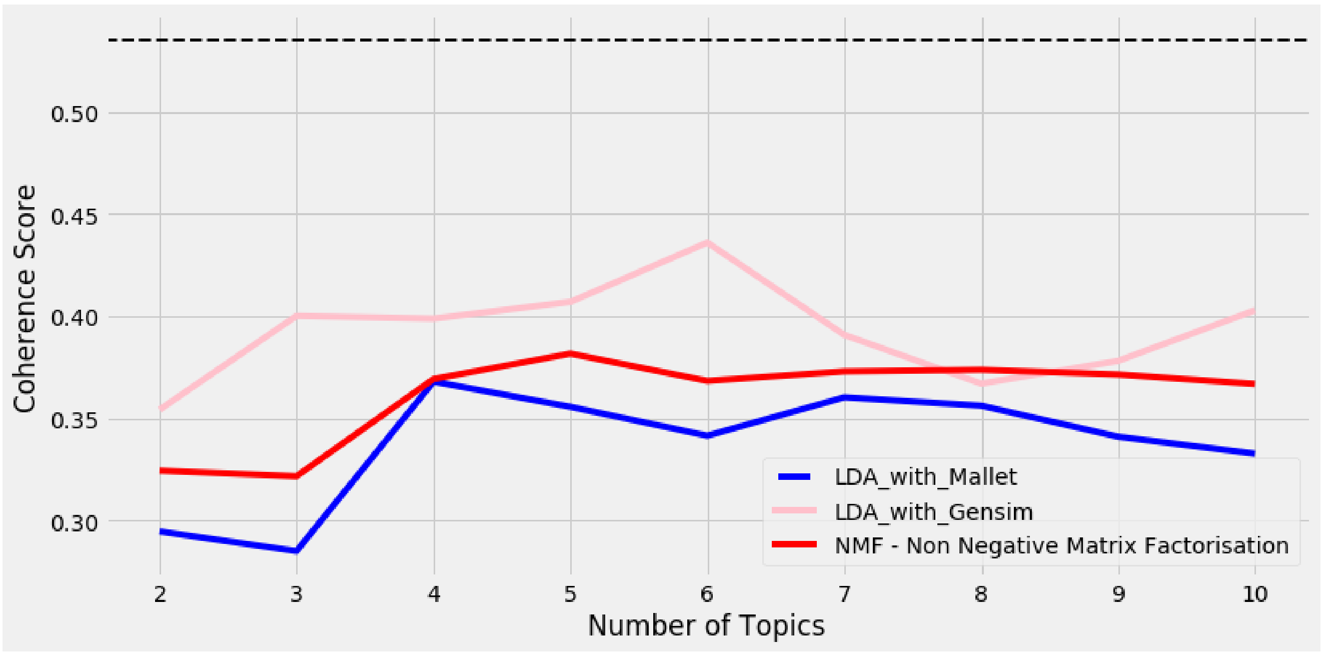
While TF-IDF is effective methods for extracting features from text, due to the inherent nature of the corpus being just a bag of unstructured words. To ensure topic modelling stays relevant and precise for each product category with respect to department or even brand labels, the raw corpus data was sliced before applying TfidfVectorizer and LDA / NMF topic modelling.

```
Bottoms_df = wc_reviews_cleaned.loc[wc_reviews_cleaned['Department Name'] ==
'Bottoms']
Dresses_df = wc_reviews_cleaned.loc[wc_reviews_cleaned['Department Name'] ==
'Dresses']
Intimate_df = wc_reviews_cleaned.loc[wc_reviews_cleaned['Department Name'] ==
'Intimate']
Jackets_df = wc_reviews_cleaned.loc[wc_reviews_cleaned['Department Name'] ==
'Jackets']
Tops_df = wc_reviews_cleaned.loc[wc_reviews_cleaned['Department Name'] == 'Tops']
Trend_df = wc_reviews_cleaned.loc[wc_reviews_cleaned['Department Name'] == 'Trend']
```

LDA, or Latent Derelicht Analysis is a probabilistic model, and to obtain cluster assignments, it uses two probability values: P(word | topics) and P(topics | documents). These values are calculated based on an initial random assignment, after which they are repeated for each word in each document, to decide their topic assignment. In an iterative procedure, these probabilities are calculated multiple times, until the convergence of the algorithm.

To build an LDA model, we would require to find the optimal number of topics to be extracted from the caption dataset. We can use the coherence score of the LDA model to identify the optimal number of topics. Approach to finding the optimal number of topics is to build many LDA models with different values of number of topics (k) and pick the one that gives the highest coherence value. Choosing a 'k' that marks the end of a rapid growth of topic coherence usually offers meaningful and interpretable topics. Picking an even higher value can sometimes provide more granular sub-topics.

When seeing the same keywords being repeated in multiple topics, it's probably a sign that the 'k' is too large. The compute_coherence_values() (see below) trains multiple LDA models and provides the models and their corresponding coherence scores.

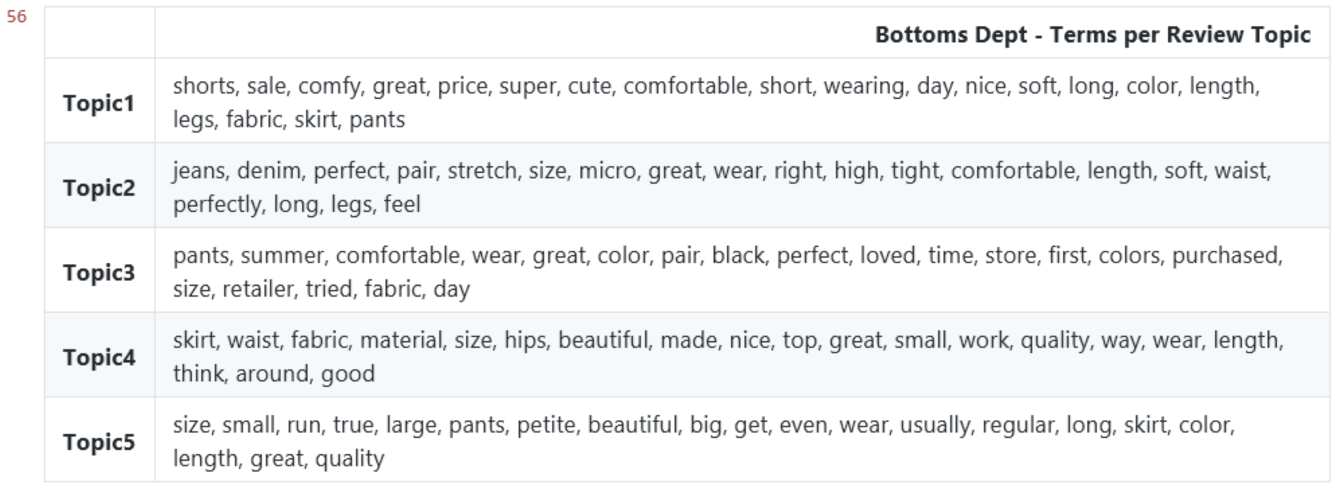


The LatentDirichletAllocation (n_components = TOTAL_TOPICS) is manually refined from the number of optimal topics of 6 derived from highest coherence score to a smaller scale upon each observation to ensure there's no repetition of topics terms. The reason is for reducing TOTAL_TOPICS to a minimal can be explained by any business instinct to find a niche topic that is unique and relevant. To ensure topic terms are unique and relevant for each department could be different, hence the parameter settings for LatentDirichletAllocation(n_components =TOTAL_TOPICS) could vary from each dataframe associated with each department.

```
from sklearn.decomposition import LatentDirichletAllocation

lda_model = LatentDirichletAllocation(n_components =TOTAL_TOPICS, max_iter=500,
max_doc_update_iter=50, doc_topic_prior=None,learning_method='online',
batch_size=128, learning_offset=50., topic_word_prior=None, random_state=42,
n_jobs=16)
```

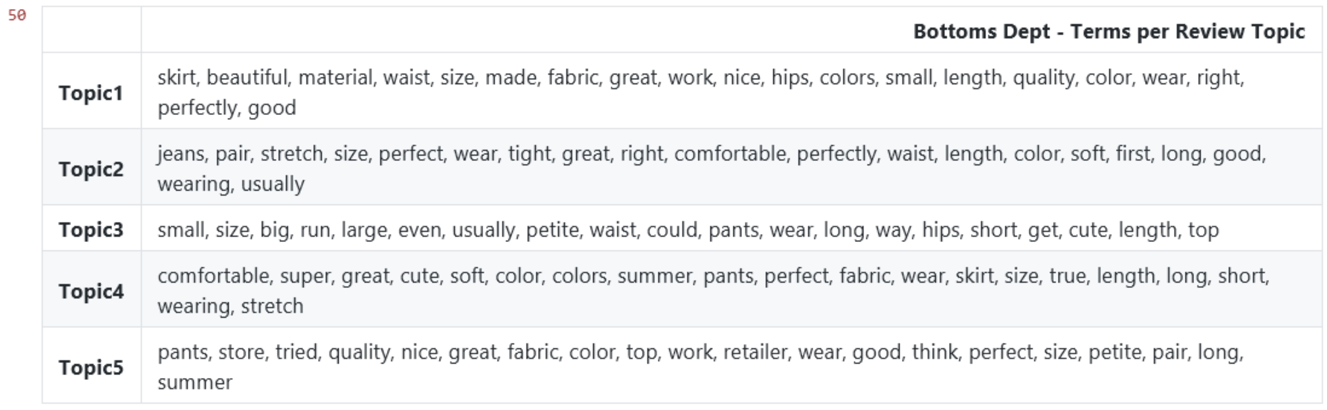
```
lda_document_topics = lda_model.fit_transform(tfidf_features
lda_topic_terms = lda_model.components_
```

When TfidfVectorize: `min_df=0.05 (75 word terms)`


56

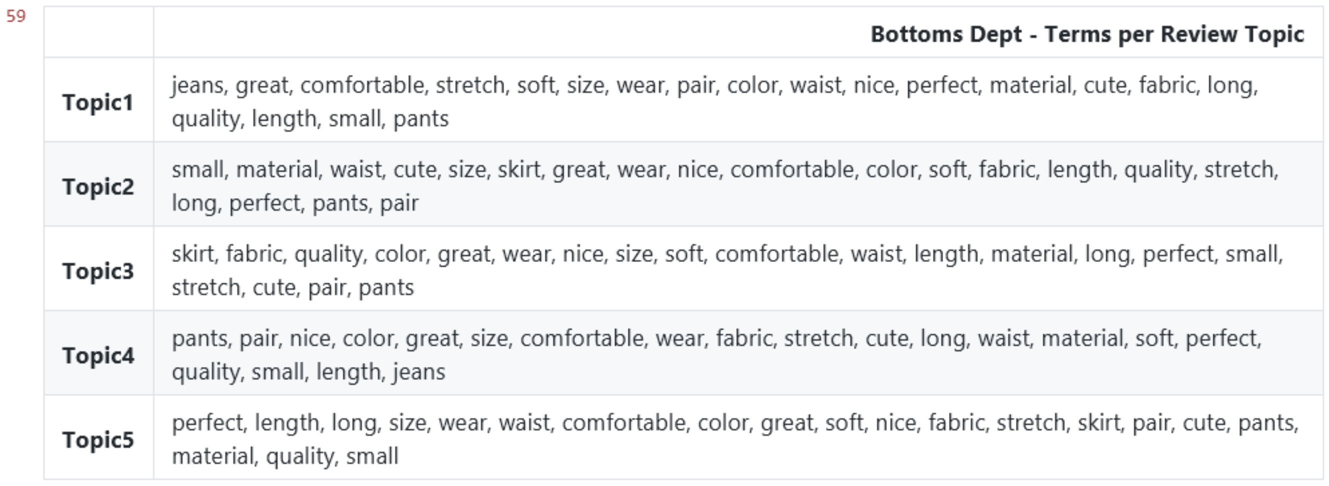
	Bottoms Dept - Terms per Review Topic
Topic1	shorts, sale, comfy, great, price, super, cute, comfortable, short, wearing, day, nice, soft, long, color, length, legs, fabric, skirt, pants
Topic2	jeans, denim, perfect, pair, stretch, size, micro, great, wear, right, high, tight, comfortable, length, soft, waist, perfectly, long, legs, feel
Topic3	pants, summer, comfortable, wear, great, color, pair, black, perfect, loved, time, store, first, colors, purchased, size, retailer, tried, fabric, day
Topic4	skirt, waist, fabric, material, size, hips, beautiful, made, nice, top, great, small, work, quality, way, wear, length, think, around, good
Topic5	size, small, run, true, large, pants, petite, beautiful, big, get, even, wear, usually, regular, long, skirt, color, length, great, quality

When TfidfVectorize: `min_df=0.07 (50 word terms)`


50

	Bottoms Dept - Terms per Review Topic
Topic1	skirt, beautiful, material, waist, size, made, fabric, great, work, nice, hips, colors, small, length, quality, color, wear, right, perfectly, good
Topic2	jeans, pair, stretch, size, perfect, wear, tight, great, right, comfortable, perfectly, waist, length, color, soft, first, long, good, wearing, usually
Topic3	small, size, big, run, large, even, usually, petite, waist, could, pants, wear, long, way, hips, short, get, cute, length, top
Topic4	comfortable, super, great, cute, soft, color, colors, summer, pants, perfect, fabric, wear, skirt, size, true, length, long, short, wearing, stretch
Topic5	pants, store, tried, quality, nice, great, fabric, color, top, work, retailer, wear, good, think, perfect, size, petite, pair, long, summer

When TfidfVectorize: `min_df=0.1 (21 word terms)`


59

	Bottoms Dept - Terms per Review Topic
Topic1	jeans, great, comfortable, stretch, soft, size, wear, pair, color, waist, nice, perfect, material, cute, fabric, long, quality, length, small, pants
Topic2	small, material, waist, cute, size, skirt, great, wear, nice, comfortable, color, soft, fabric, length, quality, stretch, long, perfect, pants, pair
Topic3	skirt, fabric, quality, color, great, wear, nice, size, soft, comfortable, waist, length, material, long, perfect, small, stretch, cute, pair, pants
Topic4	pants, pair, nice, color, great, size, comfortable, wear, fabric, stretch, cute, long, waist, material, soft, perfect, quality, small, length, jeans
Topic5	perfect, length, long, size, wear, waist, comfortable, color, great, soft, nice, fabric, stretch, skirt, pair, cute, pants, material, quality, small

Non-negative Matrix Factorization is Linear-algebraic model, that factors high-dimensional vectors into a low-dimensionality representation. Similar to Principal component analysis (PCA), NMF takes advantage of the fact that the vectors are non-negative. By factoring them into the lower-dimensional form, NMF forces the coefficients to also be non-negative.

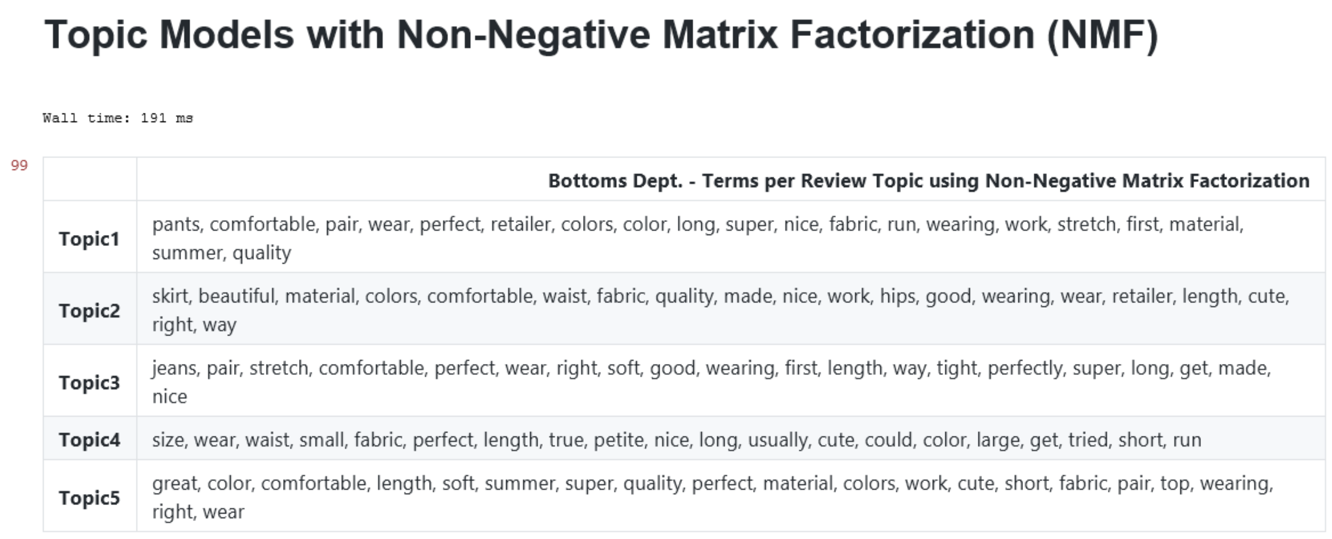
Given the original matrix A, we can obtain two matrices W and H, such that A= WH. NMF has an inherent clustering property, such that W and H represent the following information about A:

A (Document-word matrix) — input that contains which words appear in which documents.

W (Basis vectors) — the topics (clusters) discovered from the documents.

H (Coefficient matrix) — the membership weights for the topics in each document.

W and H is being calculated by optimizing over an objective function (like the EM algorithm), updating both W and H iteratively until convergence.

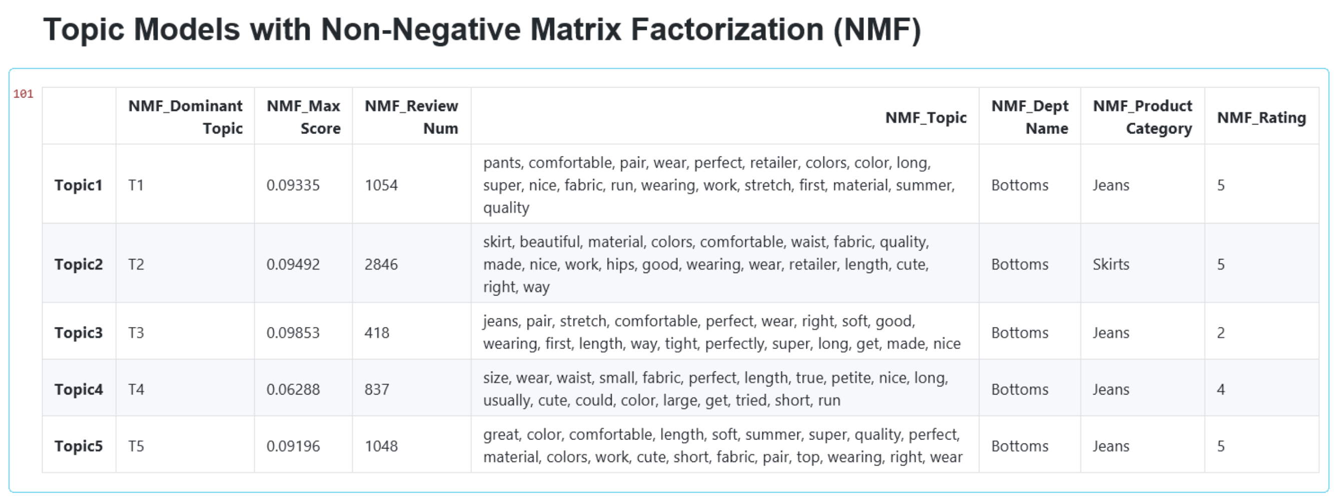


Topic Models with Non-Negative Matrix Factorization (NMF)

```
Wall time: 191 ms
```

99

	Bottoms Dept. - Terms per Review Topic using Non-Negative Matrix Factorization
Topic1	pants, comfortable, pair, wear, perfect, retailer, colors, color, long, super, nice, fabric, run, wearing, work, stretch, first, material, summer, quality
Topic2	skirt, beautiful, material, colors, comfortable, waist, fabric, quality, made, nice, work, hips, good, wearing, wear, retailer, length, cute, right, way
Topic3	jeans, pair, stretch, comfortable, perfect, wear, right, soft, good, wearing, first, length, way, tight, perfectly, super, long, get, made, nice
Topic4	size, wear, waist, small, fabric, perfect, length, true, petite, nice, long, usually, cute, could, color, large, get, tried, short, run
Topic5	great, color, comfortable, length, soft, summer, super, quality, perfect, material, colors, work, cute, short, fabric, pair, top, wearing, right, wear



Topic Models with Non-Negative Matrix Factorization (NMF)

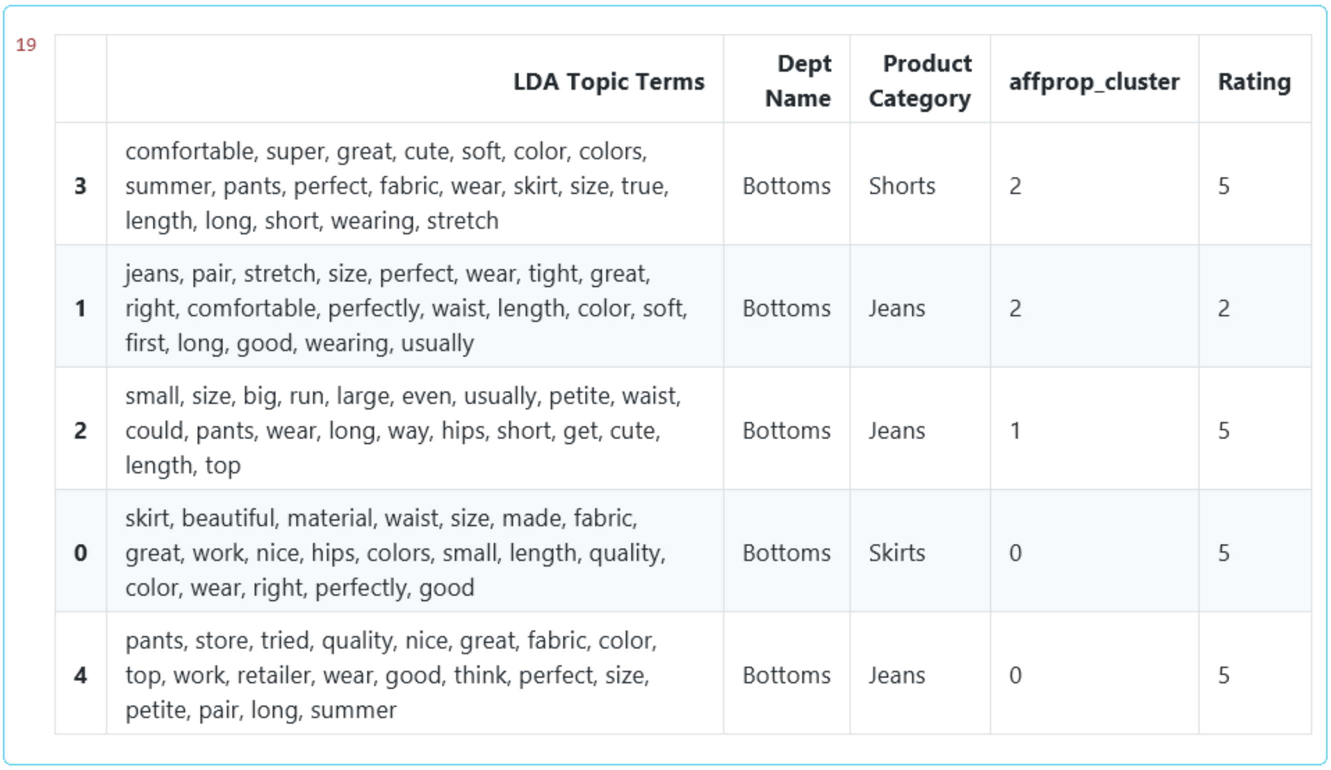
101

	NMF_Dominant Topic	NMF_Max Score	NMF_Review Num	NMF_Topic	NMF_Dept Name	NMF_Product Category	NMF_Rating
Topic1	T1	0.09335	1054	pants, comfortable, pair, wear, perfect, retailer, colors, color, long, super, nice, fabric, run, wearing, work, stretch, first, material, summer, quality	Bottoms	Jeans	5
Topic2	T2	0.09492	2846	skirt, beautiful, material, colors, comfortable, waist, fabric, quality, made, nice, work, hips, good, wearing, wear, retailer, length, cute, right, way	Bottoms	Skirts	5
Topic3	T3	0.09853	418	jeans, pair, stretch, comfortable, perfect, wear, right, soft, good, wearing, first, length, way, tight, perfectly, super, long, get, made, nice	Bottoms	Jeans	2
Topic4	T4	0.06288	837	size, wear, waist, small, fabric, perfect, length, true, petite, nice, long, usually, cute, could, color, large, get, tried, short, run	Bottoms	Jeans	4
Topic5	T5	0.09196	1048	great, color, comfortable, length, soft, summer, super, quality, perfect, material, colors, work, cute, short, fabric, pair, top, wearing, right, wear	Bottoms	Jeans	5

Next, document clustering is leveraged using these distance similarity (ie: cosine similarity) to identify how similar a text document is to any other document(s) using the topics terms generated from LDA or NMF. There are several similarity and distance metrics that are used to compute document similarity which include cosine distance/similarity, Euclidean distance, manhattan distance, BM25 similarity, jaccard distance, and so on. In our analysis, we use perhaps the most popular and widely used similarity metrics—cosine similarity before it is applied to affinity propagation document clustering for information retrieval and detecting patterns from data collections.

```
from collections import Counter
from sklearn.cluster import AffinityPropagation
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim_features = cosine_similarity(tfidf_features)

ap = AffinityPropagation(max_iter=1000)
ap.fit(cosine_sim_features)
res = Counter(ap.labels_)
res.most_common(10)

wc_reviews_topic['affprop_cluster'] = ap.labels_
filtered_clusters = [item[0] for item in res.most_common(8)]
filtered_df =
wc_reviews_topic[wc_reviews_topic['affprop_cluster'].isin(filtered_clusters)]

Woman_Clothings_clusters = (filtered_df[['LDA Topic Terms','Dept Name','Product
Category','affprop_cluster', 'Rating']]
                  .sort_values(by=['affprop_cluster', 'Product Category'],
                               ascending=False)
                  .groupby('affprop_cluster').head(20))
Woman_Clothings_clusters = Woman_Clothings_clusters.copy(deep=True)
Woman_Clothings_clusters
```

Affinity Propagation - for information retrieval and detecting patterns from data collections.


19

	LDA Topic Terms	Dept Name	Product Category	affprop_cluster	Rating
3	comfortable, super, great, cute, soft, color, colors, summer, pants, perfect, fabric, wear, skirt, size, true, length, long, short, wearing, stretch	Bottoms	Shorts	2	5
1	jeans, pair, stretch, size, perfect, wear, tight, great, right, comfortable, perfectly, waist, length, color, soft, first, long, good, wearing, usually	Bottoms	Jeans	2	2
2	small, size, big, run, large, even, usually, petite, waist, could, pants, wear, long, way, hips, short, get, cute, length, top	Bottoms	Jeans	1	5
0	skirt, beautiful, material, waist, size, made, fabric, great, work, nice, hips, colors, small, length, quality, color, wear, right, perfectly, good	Bottoms	Skirts	0	5
4	pants, store, tried, quality, nice, great, fabric, color, top, work, retailer, wear, good, think, perfect, size, petite, pair, long, summer	Bottoms	Jeans	0	5

Affinity propagation is good method for solving various clustering problems and generates that for consistently found clusters with lesser error than those establish by other methods. Since its easiness, broad applicability, and performance, considers affinity propagation will prove to be of wide value in science and engineering

We can clearly see from Affinity Propagation algorithm, it has identified 3 distinct cluster labels assigned to them. This give us a good idea of how we can use topic terms in each cluster topic documents collectively due to its similarity. It can only be taken as a reference for any A/B experiments on site traffic landing page, otherwise business judgment and domain expertise is still required when applying these document clustering for other purposes.

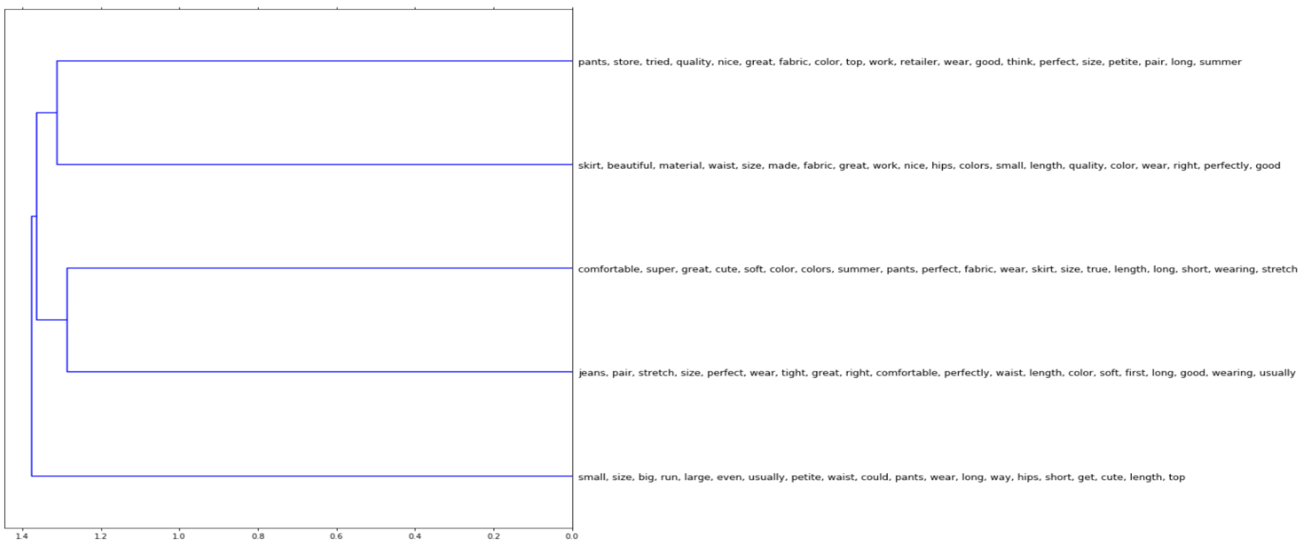
Cosine similarity is used to calculate the linkage matrix before it is used to develop hierarchical structure as a dendrogram.

To work with Ward Clustering Algorithm, the following steps were performed:

- Prepare a cosine distance matrix to calculate ward linkage_matrix
- Ward's minimum variance method is used as our linkage criterion to minimize total within-cluster variance
- Plot hierarchical structure as a dendrogram.

```
### Calculate Linkage Matrix using Cosine Similarity


def ward_hierarchical_clustering(feature_matrix):

    cosine_distance = 1 - cosine_similarity(feature_matrix)
    linkage_matrix = ward(cosine_distance)
    return linkage_matrix

### Plot Hierarchical Structure as a Dendrogram - version 1


def plot_hierarchical_clusters(linkage_matrix, data, figure_size=(8,12)):
    # set size
    fig, ax = plt.subplots(figsize=figure_size)
    wc_review_topic_clusters = data['Topic'].values.tolist()
    # plot dendrogram
    ax = dendrogram(linkage_matrix, orientation="left",
labels=wc_review_topic_clusters)
    plt.tick_params(axis= 'x',
                    which='both',
                    bottom='off',
                    top='off',
                    labelbottom='off')
    plt.tight_layout()
    plt.savefig(path + 'ward_hierachical_clusters.png', dpi=300)



### Plot Hierarchical Structure as a Dendrogram - version 2


def plot_ward_hierarchical_clusters(data, figure_size=(8,12)):
    from scipy.cluster.hierarchy import ward, dendrogram, linkage
    from sklearn.metrics.pairwise import cosine_similarity

    similarity_matrix = cosine_similarity(tfidf_features)
    Z = linkage(similarity_matrix, 'ward')
    fig, ax = plt.subplots(figsize=figure_size)
    plt.title("Bottoms Dept - Hierarchical Structure as a Dendrogram")
    wc_review_topic_clusters = data['LDA Topic Terms'].values.tolist()
    # plot dendrogram
    ax = dendrogram(Z, orientation="left", labels=wc_review_topic_clusters)
    plt.tick_params(axis= 'x',
                    which='both',
                    bottom='off',
                    top='off',
                labelbottom='off')
    plt.tight_layout()
    plt.savefig(path + 'ward_hierachical_clusters.png', dpi=300)
```

Below is the hierarchical dendrogram that can be applied across when performing documents or topics clustering



d. Actual text analysis using machine learning:

i. Train models using various supervised learning, unsupervised learning, reinforcement learning, and deep learning algorithms.

Models used for training is to determine sentiments polarity. It includes supervised learning algorithm such as Linear Support Vector Machine (SVM), Decision Tree Classifier & ensemble learning Classifier AdaBoostClassifier, RandomForest Classifier, based on predictor such as review text or opinions. To ensure sentiment polarity rating prediction do not end up in floating values, the submitted numerical ratings are label encoded by transforming them into categorical data expressed as “Positive”, “Neutral” and “Negative”.

A comparative approach towards sentiment analysis of product reviews or opinions include using Lexicon-Based model which is unsupervised learning method that works well without the presence of labeled data on sentiment response from ratings.

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

def sentiments(rating):
    if (rating == 5) or (rating == 4):
        return "Positive"
    elif rating == 3:
        return "Neutral"
    elif (rating == 2) or (rating == 1):
        return "Negative"


X = Bottoms_df.drop(columns=['Rating'])
Y = Bottoms_df['Rating']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y)

# Add sentiments to the data
Y_train['Sentiment'] = Y_train.apply(sentiments)
Y_test['Sentiment'] = Y_test.apply(sentiments)

# Prepare data
X_train = X_train['cleaned_words']
X_test = X_test['cleaned_words']
Sentiment_train = Y_train["Sentiment"]
Sentiment_test = Y_test["Sentiment"]
```

Machine learning pipeline is used to help automate machine learning workflows operate by enabling the feature engineering of CountVectorizor, tfidf before applying various machine learning algorithm with different hyper-parameters to achieve an outcome, whether opinions from the reviews sentiment is predicted to be positive, neutral or negative. Python sklearn Pipeline can be set up to encapsulate the fit/transform/predict functionality to streamline predictor and transformer to build the prediction model. Pipeline allows the use of grid search over a set of parameters for each step of its meta-estimator.

```
**Support Vector Machine Classifier**

from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
clf_linearSVC_pipe = Pipeline([("vect", CountVectorizer()), ("tfidf",
TfidfTransformer()), ("clf_linearSVC", LinearSVC())])
clf_linearSVC_pipe.fit(X_train, Sentiment_train)

predictedLinearSVC = clf_linearSVC_pipe.predict(X_test)
print("Support Vector Machine Classifier:{:.3f}".format(np.mean(predictedLinearSVC
== Sentiment_test)))


**Decision Tree Classifier**

from sklearn.tree import DecisionTreeClassifier
clf_decisionTree_pipe = Pipeline([("vect", CountVectorizer()), ("tfidf",
TfidfTransformer()), ("clf_decisionTree", DecisionTreeClassifier(criterion='gini',
splitter='best'))])
```

```
clf_decisionTree_pipe.fit(X_train, Sentiment_train)

predictedDecisionTree = clf_decisionTree_pipe.predict(X_test)
print("Decision Tree Classifier:{:.3f}".format(np.mean(predictedDecisionTree ==
Sentiment_test)))

**Random Forest Classifier**


from sklearn.ensemble import RandomForestClassifier
clf_randomForest_pipe = Pipeline([("vect", CountVectorizer()), ("tfidf",
TfidfTransformer()), ("clf_randomForest",
RandomForestClassifier(criterion='gini'))])
clf_randomForest_pipe.fit(X_train, Sentiment_train)

predictedRandomForest = clf_randomForest_pipe.predict(X_test)
print("Random Forest Classifier :{:.3f}".format(np.mean(predictedRandomForest ==
Sentiment_test)))


**Ada Boost Tree Classifier**


from sklearn.ensemble import AdaBoostClassifier
clf_AdaBoostClassifier_pipe = Pipeline([("vect", CountVectorizer()), ("tfidf",
TfidfTransformer()), ("clf_AdaBoost", AdaBoostClassifier())])
clf_AdaBoostClassifier_pipe.fit(X_train, Sentiment_train)

predicted_AdaBoostClassifier = clf_AdaBoostClassifier_pipe.predict(X_test)
print("AdaBoostClassifier :{:.3f}".format(np.mean(predicted_AdaBoostClassifier ==
Sentiment_test)))
```

Here, Grid Search of the best parameters is performed on a grid of possible values, instead of tweaking the parameters of various components of the chain (ie. use_idf in tfidftransformer). Grid search will be performed using LinearSVC, RandomForest & AdaBoost classifier pipeline, parameters and cpu core maximization.

```
from sklearn.model_selection import GridSearchCV

parameters = {'clf_AdaBoost__n_estimators': list(range(5, 30, 40)),
              'clf_AdaBoost__algorithm': ('SAMME', 'SAMME.R')}
gs_clf_AdaBoost_pipe = GridSearchCV(clf_AdaBoostClassifier_pipe, parameters, cv=5)
gs_clf_AdaBoost_pipe = gs_clf_AdaBoost_pipe.fit(X_train, Sentiment_train)
```

Apart from the use of traditional supervised learning model such as LinearSVM, DecisionTree or ensemble learning model, a comparative study is done by using Python nltk unsupervised lexicons-based library models such as SentiWordNet & VADER lexicon. The virtue of using unsupervised lexicons-based model is that it does not require the presence of label dataset (ie: ratings) in order to perform sentiments classification. This unsupervised lexicons-based model makes sentiments prediction simpler when ordinal data (ie: ratings) is not available. The inputs of opinions or text reviews is only required for unsupervised lexicons-based model after performing necessary text preprocessing steps to remove special & unwanted html tag characters.

The reason for evaluating lexicon-based model primarily include its ability in considering factors like presence of negation parameters, surrounding words, overall context, phrases before aggregating them to determine sentiment polarity. Since we have labeled data of sentiments derived from ratings, we then use confusion matrix table and classification report to evaluate sentiment ratings values for these woman clothing's match against lexicon-mode based predicted sentiment values.

One of the parameter such as threshold value =0.05 is used to categorize the sentiment aggregate score computed by lexicon-based mode into 3 categories (ie: "Positive", "Neutral" & "Negative" in similar scale as the actual sentiment rating. In this way, it ensures standardization of scale necessary for evaluating performance of classification model using confusion matrix table and classification report.

```
for idx, (review, actual_sentiment) in enumerate(zip(X,Sentiments)):
    print('REVIEW:', review)
    print('Rating Sentiment:', actual_sentiment)
    sentiwordnet_predicted_sentiment= analyze_sentiment_sentiwordnet_lexicon(review,
threshold=0.05, verbose=True )
    print('Predict Sentiwordnet Sentiment:', sentiwordnet_predicted_sentiment)
```

```
def sentiment_scoring(agg_score, threshold):
    if agg_score >= threshold:
        return 'Positive'
    elif agg_score >= 0 or agg_score <= threshold:
        return "Neutral"
    else:
        return "Negative"
```

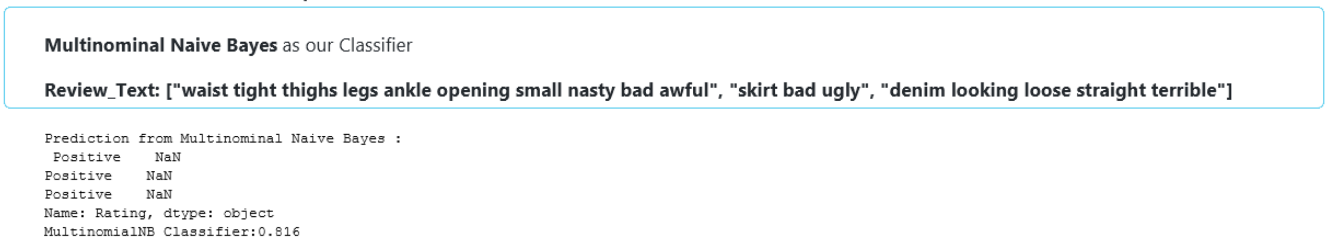
This threshold value settings is standardized across the SentiWordNet & VADER lexicon-based model to ensure consistent comparison and evaluation from the confusion matrix table and classification report.

```
for idx, (review, actual_sentiment) in enumerate(zip(X,Sentiments)):
    print('REVIEW:', review)
    print('Rating Sentiment:', actual_sentiment)
    predicted_sentiments= analyze_sentiment_vader_lexicon(review, threshold=0.05,
verbose=True)
    print('Predict Sentiment:', predicted_sentiments)
```

ii. Evaluate the performance of the models.

Linear Support Vector Machines is observed to very suitable for classification by measuring extreme values between classes, to differentiate the worst case scenarios such that it can classify between Positive, Neutral and Negative correctly even with accuracy about 83%, unlike ensemble learning model such as RandomForest, AdaBoost & Multinominal Naive Bayes which has poor prediction accuracy on the sentiment even from evaluation test from human observation. As shown below on the evaluation test conducted, Review_Text containing words such as ‘awful’, ‘bad’, ‘ugly’, ‘terrible’ has been predicted incorrectly by RandomForest, Multinominal Naive Bayes & AdaBoost classify as “Positive’ sentiment,.

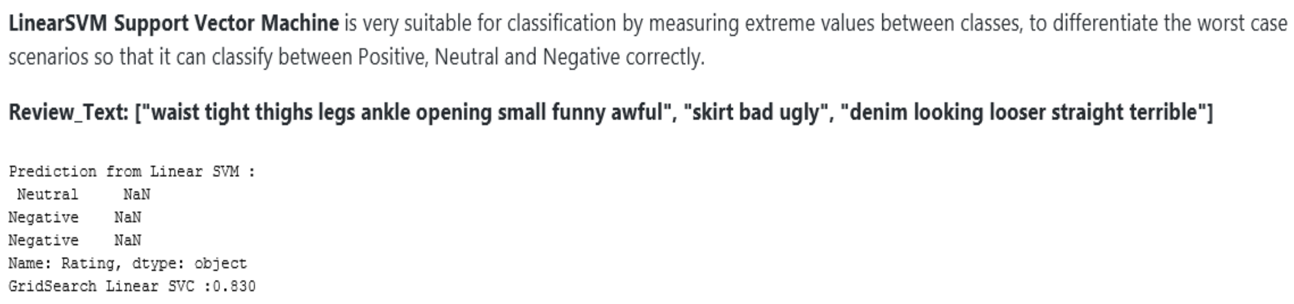
Regardless of whether ngram_range or unigram, whenever there are words with contrasting sentiments such as “pretty loose but terrible”, Linear Support Vector Machines prediction will be incorrect even with the same accuracy values. Hence, this supervised learning model limitation of not considering surrounding words & overall context can be overcome by lexicon-based model such as SentiWordNet & VADER.



Multinominal Naive Bayes as our Classifier

Review_Text: ["waist tight thighs legs ankle opening small nasty bad awful", "skirt bad ugly", "denim looking loose straight terrible"]

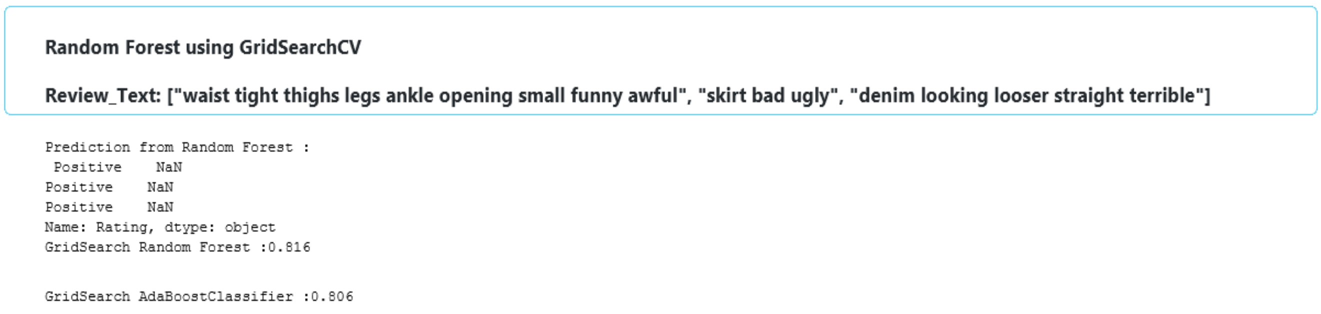
```
Prediction from Multinominal Naive Bayes :
 Positive    NaN
Positive    NaN
Positive    NaN
Name: Rating, dtype: object
MultinomialNB Classifier:0.816
```



LinearSVM Support Vector Machine is very suitable for classification by measuring extreme values between classes, to differentiate the worst case scenarios so that it can classify between Positive, Neutral and Negative correctly.

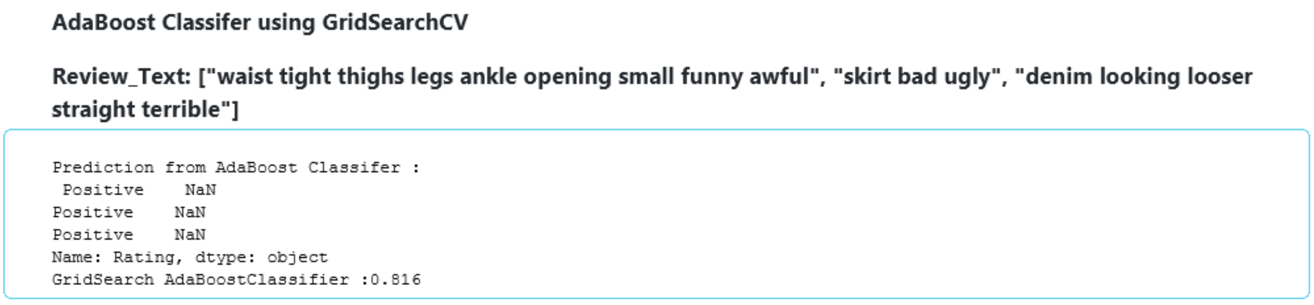
Review_Text: ["waist tight thighs legs ankle opening small funny awful", "skirt bad ugly", "denim looking looser straight terrible"]

```
Prediction from Linear SVM :
 Neutral     NaN
Negative    NaN
Negative    NaN
Name: Rating, dtype: object
GridSearch Linear SVC :0.830
```


Random Forest using GridSearchCV

Review_Text: ["waist tight thighs legs ankle opening small funny awful", "skirt bad ugly", "denim looking looser straight terrible"]

```
Prediction from Random Forest :
 Positive    NaN
Positive    NaN
Positive    NaN
Name: Rating, dtype: object
GridSearch Random Forest :0.816

GridSearch AdaBoostClassifier :0.806
```

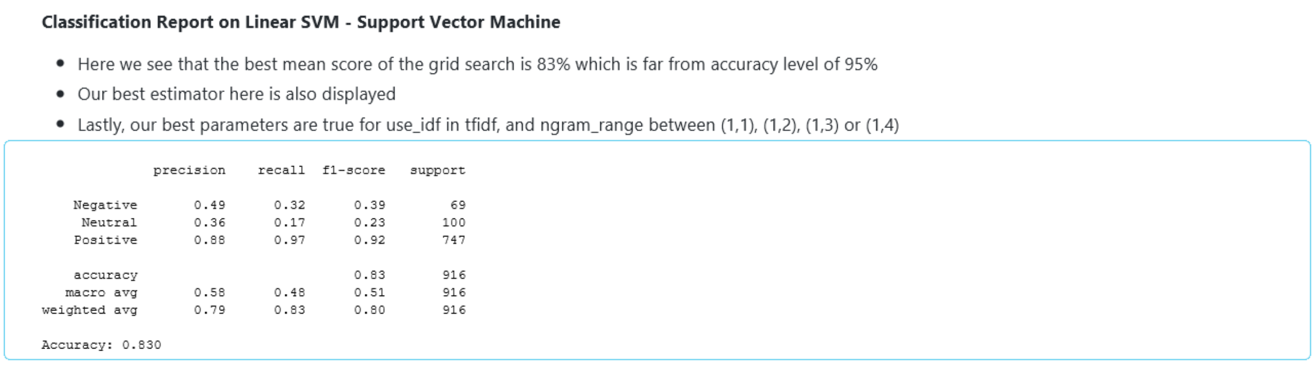

AdaBoost Classifer using GridSearchCV

Review_Text: ["waist tight thighs legs ankle opening small funny awful", "skirt bad ugly", "denim looking looser straight terrible"]

```
Prediction from AdaBoost Classifer :
 Positive    NaN
Positive    NaN
Positive    NaN
Name: Rating, dtype: object
GridSearch AdaBoostClassifier :0.816
```

The classification report below only display the best estimator - Linear SVM (Support Vector Machine) with best mean score of 83% even with Gridsearch which is far from accuracy level of 95%.

Below is the summary of the classification report:


Classification Report on Linear SVM - Support Vector Machine

- Here we see that the best mean score of the grid search is 83% which is far from accuracy level of 95%
- Our best estimator here is also displayed
- Lastly, our best parameters are true for use_idf in tfidf, and ngram_range between (1,1), (1,2), (1,3) or (1,4)

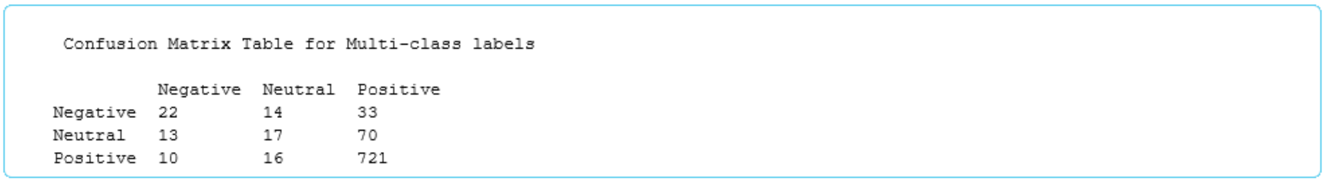
```
              precision    recall  f1-score   support

    Negative       0.49      0.32      0.39        69
     Neutral       0.36      0.17      0.23       100
    Positive       0.88      0.97      0.92       747

    accuracy                           0.83       916
   macro avg       0.58      0.48      0.51       916
weighted avg       0.79      0.83      0.80       916

Accuracy: 0.830
```

- ❒ Precision: determines how many objects selected were correct
- ❒ Recall: tells you how many of the objects that should have been selected were actually selected
- ❒ F1 score measures the weights of recall and precision (1 means precision and recall are equally important, 0 otherwise)
- ❒ Support is the number of occurrences of each class

The results in this analysis confirms our previous data exploration analysis, where the data are very skewed to the positive reviews as shown by the higher support counts in the classification report.

Despite of the fact that Neutral and Negative results are not very strong predictors in this data set, it still shows 83% accuracy level in predicting the sentiment analysis, which we tested and worked very well when inputting unseen text (Review_Test).

Finally, the overall result from Confusion Matrix shows the performance of classification model (or "classifier") for Linear Support Vector Model on a set of test data for which the true values are known.



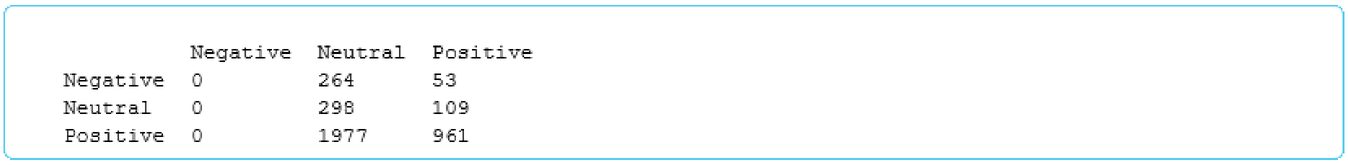
```
 Confusion Matrix Table for Multi-class labels

          Negative  Neutral  Positive
Negative  22        14       33
Neutral   13        17       70
Positive  10        16       721
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Sentiment_test, predictedGS_clf_LinearSVC_pipe)

print(" Confusion Matrix Table for Multi-class labels\n")
cm_df = pd.DataFrame(cm, index = ['Negative','Neutral','Positive'], columns =
['Negative','Neutral','Positive'])
print(cm_df)
```

We see that Positive sentiment prediction can be daunting with neutral and negative ratings being misclassified with count of 70 and 33 respectively. However, based on the overall number of significant positive sentiment at a score 747, this misclassification error of 17% are considered significant depending on the application if this happens to be implemented in recommender system.

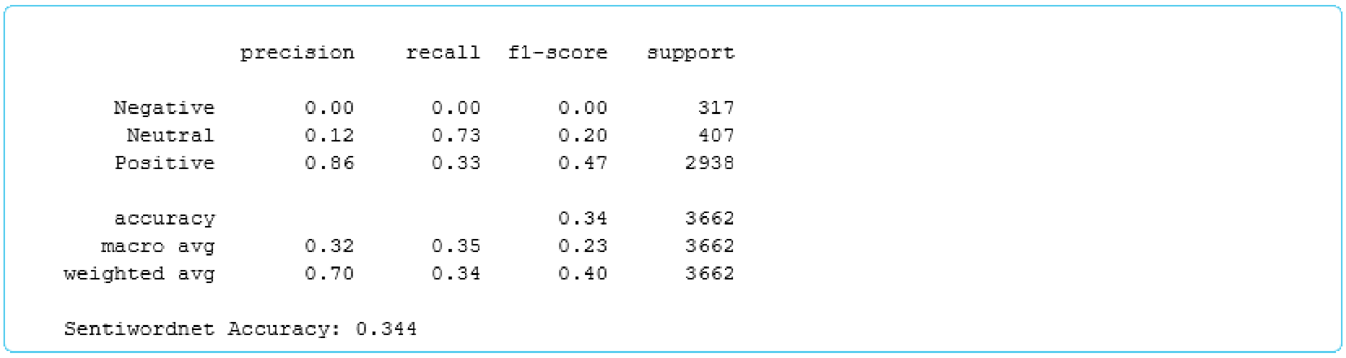
The evaluation in the section below describes the performance of SentiWordNet & VADER lexicon-based model.

Analyze_SentiWordNet Confusion Matrix - performance of classification model Report



```
          Negative  Neutral  Positive
Negative  0         264      53
Neutral   0         298      109
Positive  0         1977     961
```

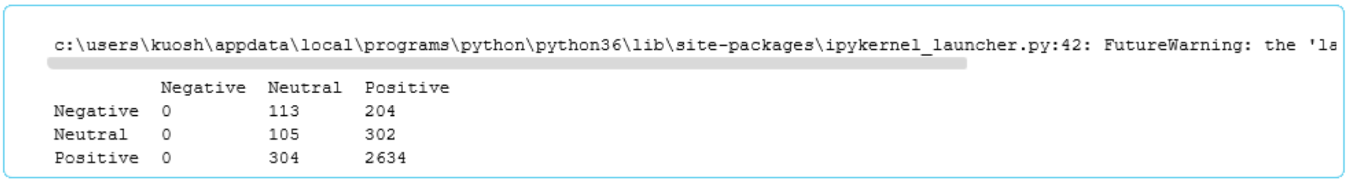
Analyze_SentiWordNet Classification Report



```
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00       317
     Neutral       0.12      0.73      0.20       407
    Positive       0.86      0.33      0.47      2938

    accuracy                           0.34      3662
   macro avg       0.32      0.35      0.23      3662
weighted avg       0.70      0.34      0.40      3662

Sentiwordnet Accuracy: 0.344
```

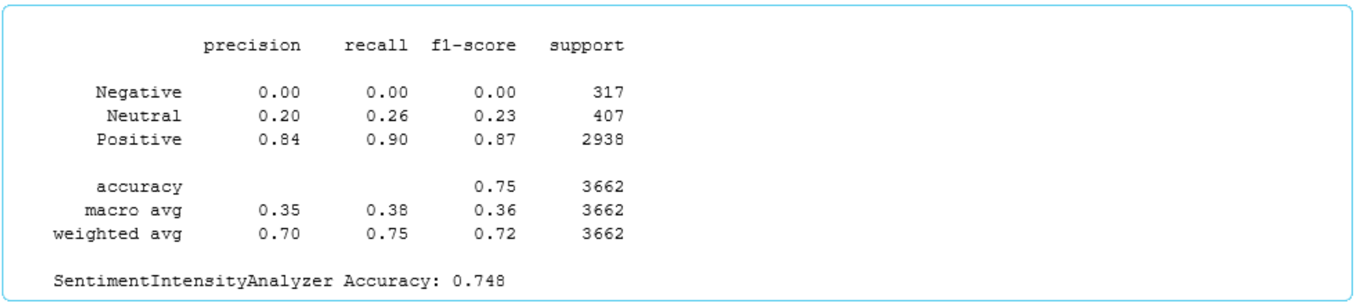
Based on the threshold settings applied the same to both models, the classification report shows that VADER has a better accuracy performance with lower probabilities of false negative and false positive. The confusion matrix table shows the count of true positive, true negative, false negative and false positive from the classification. Hence, VADER model will be preferred choice in predicting unseen observations.

Analyze_Sentiment_vader_lexicon Confusion Matrix - Misclassification Report



```
c:\users\kuosh\appdata\local\programs\python\python36\lib\site-packages\ipykernel_launcher.py:42: FutureWarning: the 'la

         Negative  Neutral  Positive
Negative  0         113       204
Neutral   0         105       302
Positive  0         304       2634
```

Analyze_Sentiment_vader_lexicon - Classification Report



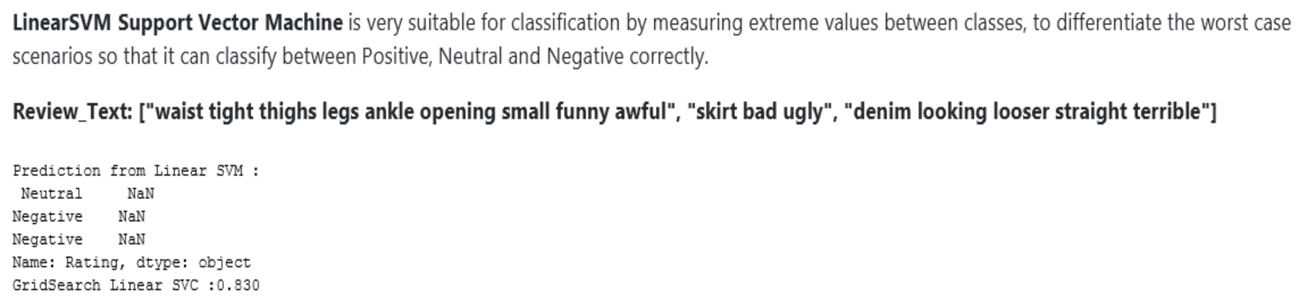
```
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00       317
     Neutral       0.20      0.26      0.23       407
    Positive       0.84      0.90      0.87      2938

    accuracy                           0.75      3662
   macro avg       0.35      0.38      0.36      3662
weighted avg       0.70      0.75      0.72      3662

SentimentIntensityAnalyzer Accuracy: 0.748
```

iii. **Deploy the final chosen model to predict new unseen observations.**

The "Review_text" list is created to test and predict new unseen observations since the final chosen model - LinearSVM is able to classify between Positive, Neutral and Negative correctly. Regardless of the ngram_range, as long as there are no words with contrasting sentiments LinearSVM will not result in wrong classification with false negative or false positive. In other words, whenever Negative and Positive sentiment words co-exist, the LinearSVM classifier will result in True Negative instead of False Negative. This misclassification error could be the result of positively skewed dataset as observed in exploratory analysis on the labeled rating dataset. LinearSVM is deployed in conjunction with GridSearchCV so that it exhaustively generates candidates from a grid of parameter values specified with the param_grid parameter. Hyper-parameters are parameters that are not directly learnt within estimators.

Hence, it is recommended to search the hyper-parameter space for the best cross validation score. Any parameter provided when constructing an estimator may be optimized in this manner using GridSearchCV.

```
from sklearn.model_selection import GridSearchCV
parameters = {'vect__ngram_range': [(1, 1), (1, 2), (1, 3), (1, 4)],
             'tfidf__use_idf': (True, False),
             }
gs_clf_LinearSVC_pipe = GridSearchCV(clf_linearSVC_pipe, parameters, n_jobs=-1)
gs_clf_LinearSVC_pipe = gs_clf_LinearSVC_pipe.fit(X_train, Sentiment_train)
Review_text = ["waist tight thighs legs pretty ankle opening small bad awful",
            "skirt bad looking ugly",
            "denim looking loose straight terrible"]

print("Prediction from Linear SVM : \n",
Sentiment_train[gs_clf_LinearSVC_pipe.predict(Review_text)])
predictedGS_clf_LinearSVC_pipe = gs_clf_LinearSVC_pipe.predict(X_test)
print("GridSearch Linear SVC :{:.3f}".format(np.mean(predictedGS_clf_LinearSVC_pipe
== Sentiment_test)))
```



LinearSVM Support Vector Machine is very suitable for classification by measuring extreme values between classes, to differentiate the worst case scenarios so that it can classify between Positive, Neutral and Negative correctly.

Review_Text: ["waist tight thighs legs ankle opening small funny awful", "skirt bad ugly", "denim looking looser straight terrible"]

```
Prediction from Linear SVM :
 Neutral      NaN
Negative    NaN
Negative    NaN
Name: Rating, dtype: object
GridSearch Linear SVC :0.830
```

LinearSVM model is able to predict the unseen review text list sentiment polarity as “Neutral”, “Negative”, “Negative”.

For the Unsupervised lexicon-based model given that threshold value settings applied are the same., VADER lexicon-based model will be adopted instead of SentiWordNet due to higher overall accuracy.

Reference

[1] Nishit Shrestha and Fatma Nasoz, “DEEP LEARNING SENTIMENT ANALYSIS OF AMAZON.COM REVIEWS AND RATINGS,” International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI), Vol.8, No.1, February 2019 [Access on 29-Apr-2020]

[2]T. Pranckevičius and V. Marcinkevičius, "Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification," Baltic Journal of Modern Computing, vol. 5, (2), pp. 221-232, 2017. Available: http://libproxy1.nus.edu.sg/login?url=https://search-proquest-com.libproxy1.nus.edu.sg/docview/1924818053?accountid=13876. DOI: http://dx.doi.org.libproxy1.nus.edu.sg/10.22364/bjmc.2017.5.2.05. [Access on 30-Apr-2020]