Sentiment analysis of online product reviews using NLP

Wainganker Varsha

Presentation Overview

1. **Problem Statement**
2. **Dataset Overview**
3. **Strategy Explained**
4. **Analysis and Findings**
5. **Best Model Identified**
6. **Conclusion**

Data Collection

- This is a list of over 71,045 reviews from 1,000 different products provided by [Datafiniti's Product Database].
- The dataset includes the text and title of the review, the name and manufacturer of the product, reviewer metadata, and more.
- Note that this is a sample of a large data set

Problem Statement

- Purpose of this project is to build a recommendation system that will accurately predict the sentiment , review title, review text as positive or negative
- Also predict the ratings from 1 to 5
- Find the best model with highest performance metric, that will yield appropriate text classification

Business Case

- Technological advances over the past decade have led to the proliferation of consumer review websites such as Walmart, target, where consumers can share experiences about product quality. These reviews provide consumers with information about the goods, and the quality that is observed only after consumption.
- With the click of a button, one can now acquire information from countless other consumers products ranging from food, personal care, medicine, household goods etc.

Dataset Overview

1. Column Definitions
2. Dataset Values
3. Data Analysis

Dataset Values (Breakdown of Data Frame)

```
In [2]: df = pd.read_csv("C:/Users/Varsha/.spyder-py3/DataSetReviews.csv")

In [3]: df.columns
Out[3]:
Index(['id', 'brand', 'categories', 'dateAdded', 'dateUpdated', 'ean', 'keys',
       'manufacturer', 'manufacturerNumber', 'name', 'reviews.date',
       'reviews.dateAdded', 'reviews.dateSeen', 'reviews.didPurchase',
       'reviews.doRecommend', 'reviews.id', 'reviews.numHelpful',
       'reviews.rating', 'reviews.sourceURLs', 'reviews.text', 'reviews.title',
       'reviews.userCity', 'reviews.userProvince', 'reviews.username', 'upc'],
      dtype='object')

In [4]: df.shape
Out[4]: (71044, 25)

In [5]: |
```

Breakdown of Data Frame

```
#combining required features and creating a new dataframe
best_ratings = df2[['id','manufacturerNumber','reviews.rating', 'reviews.title','reviews.text']]

best_ratings.head(4)
```

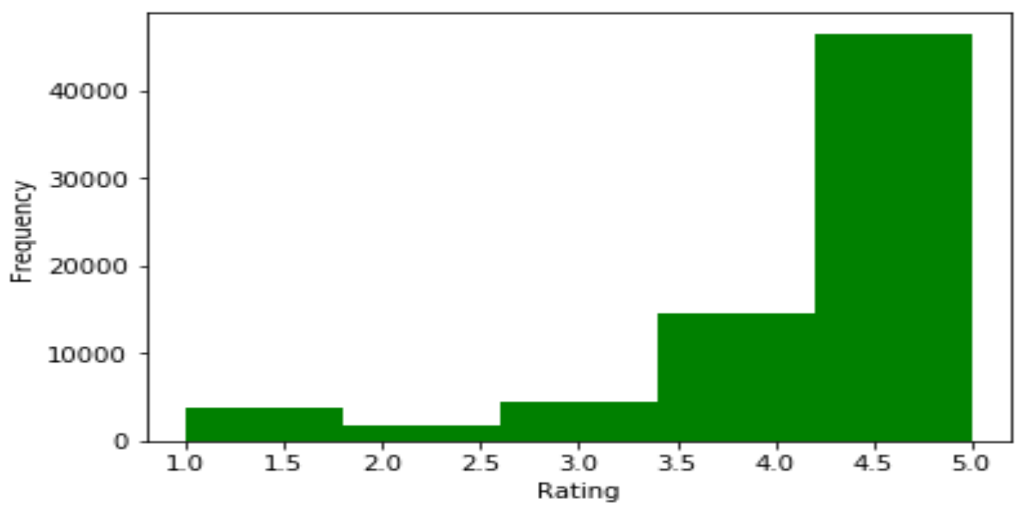
	id	manufacturerNumber	reviews.rating	reviews.title	reviews.text
0	AV13O1A8GV-KLJ3akUyj	14331328	5	Just Awesome	i love this album. it's very good. more to the...
1	AV14LG0R-jtxr-f38QfS	574764	5	Good	Good flavor. This review was collected as part...
2	AV14LG0R-jtxr-f38QfS	574764	5	Good	Good flavor.
3	AV16khLE-jtxr-f38VFn	67981934427	1	Disappointed	I read through the reviews on here before look...

Data Analysis

```
In [9]: #number of rows in dataframe  which are the total number of ratings
   ...: total_num_of_ratings = len(best_ratings)
   ...: print ("total number of user-items ratings is: %d" % total_num_of_ratings)
   ...:
total number of user-items ratings is: 71044

In [10]: #number of unique reviewers
    ...: num_of_users = len(best_ratings['id'].unique())
    ...: print ("number of unique reviewers is: %d" % num_of_users)
    ...:
number of unique reviewers is: 600

In [11]: #number of unique items
    ...: num_of_items = len(best_ratings['manufacturerNumber'].unique())
    ...: print( "number of unique items is: %d" % num_of_items)
    ...:
number of unique items is: 584
```

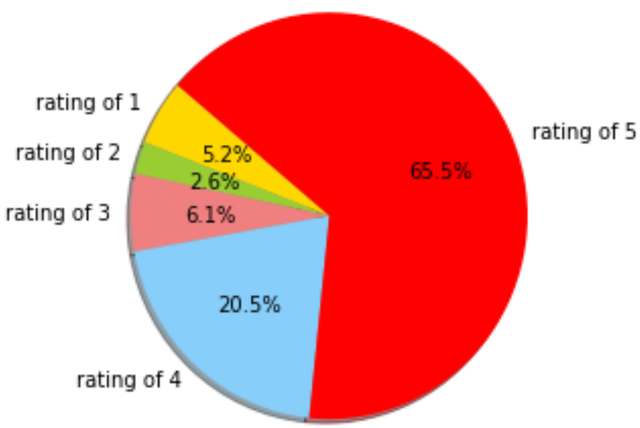
Histogram –Frequency of Rating

```
In [12]: #frequency of each rating in a histogram
    ...: plt.figure(1)
    ...: plt.hist(best_ratings['reviews.rating'].dropna(),5, facecolor='g')
    ...: plt.xlabel('Rating')
    ...: plt.ylabel('Frequency')
    ...: plt.show()
    ...:
```



Distribution of Rating

```
In [13]: # percentage of each rating in a pie chart
    ...: plt.figure(2)
    ...: labels = 'rating of 1', 'rating of 2' ,'rating of 3', 'rating of 4','rating of 5'
    ...: colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue','red']
    ...: plt.pie(best_ratings.groupby('reviews.rating').size(),labels=labels, colors=colors,autopct='%1.1f%%',
shadow=True, startangle=140)
    ...: plt.axis('equal')
    ...: plt.show()
    ...:
```




```
In [2]: sum(df['reviews.rating']==1)
Out[2]: 3701

In [3]: sum(df['reviews.rating']==2)
Out[3]: 1833

In [4]: sum(df['reviews.rating']==3)
Out[4]: 4369

In [5]: sum(df['reviews.rating']==4)
Out[5]: 14598

In [6]: sum(df['reviews.rating']==5)
Out[6]: 46543
```

Converting the ratings into binary factors (positive/negative)

```
In [14]:
    ...: #Creating new dataframe with few features, required for our prediction model
    ...: df1 = pd.DataFrame(df, columns = ['reviews.title', 'reviews.rating','reviews.text'])
    ...:
    ...:
    ...: #Function to select values lower than and greater than 3
    ...: def mask_with_values(df1):
    ...:     mask = df1['reviews.rating']!= 3
    ...:     return df1[mask]
    ...:
    ...: df2 =pd.DataFrame(mask_with_values(df1))
    ...:
    ...: #converting numbers to categorical values negative and positive
    ...:
    ...: def partition(x):
    ...:     if x < 3:
    ...:      return 'negative'
    ...:     return 'positive'
    ...:
    ...: df2['reviews.rating'] = df2['reviews.rating'].map(partition)
    ...: df2['reviews.title'] = df2['reviews.title']
    ...: df2['reviews.text']  = df2['reviews.text']
    ...:
    ...: #Final data frame with reviews identified as positive or negative
    ...: df3=pd.DataFrame(df2 , columns= ['reviews.rating','reviews.title', 'reviews.text'])
```

Before and after conversion

```
df1.head(4)
```

	reviews.rating	reviews.title	reviews.text
0	5	Just Awesome	i love this album. it's very good. more to the...
1	5	Good	Good flavor. This review was collected as part...
2	5	Good	Good flavor.
3	1	Disappointed	I read through the reviews on here before look...

```
df3.head(4)
```

	reviews.rating	reviews.title	reviews.text
0	positive	Just Awesome	i love this album. it's very good. more to the...
1	positive	Good	Good flavor. This review was collected as part...
2	positive	Good	Good flavor.
3	negative	Disappointed	I read through the reviews on here before look...

Text Processing

1. Tokenization
2. Stemming
3. Stop words removal
4. Count Vectorization
5. Term Frequency –Inverse Document Frequency

Step 1 : Tokenization

- **Tokenization** is a method of breaking up a piece of text into many pieces, such as sentences and words

```
In [4]: #Tokenization example
   ...: import nltk
   ...: from nltk.tokenize import word_tokenize
   ...: tokens = nltk.word_tokenize("According to a study, 84% of vegetarians in
America eventually go back to eating meat: source 9gag")
   ...: print(tokens)
   ...:
['According', 'to', 'a', 'study', ',', '84', '%', 'of', 'vegetarians', 'in',
'America', 'eventually', 'go', 'back', 'to', 'eating', 'meat', ':', 'source',
'9gag']
```

Step 2 : Stemming

- Stemmers remove morphological affixes from words, leaving only the word stem.

```
In [7]: #Example for stemming
   ...: #Importing the package
   ...: from nltk.stem.porter import PorterStemmer
   ...: #Function call
   ...: stemmer = PorterStemmer()
   ...: plurals = ['caresses', 'flies', 'dies', 'mules', 'denied',  'died',         'agreed',
'owned', 'humbled', 'sized','meeting', 'stating', 'siezing','sensational', 'traditional',
'reference', 'colonizer','plotted']
   ...: singles = [stemmer.stem(plural) for plural in plurals]
   ...: print(' '.join(singles))
   ...:
caress fli die mule deni die agre own humbl size meet state siez sensat tradit refer colon plot
```

Step 3 : Count Vectorization

The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called **feature extraction (or vectorization).**

- The *CountVectorizer* provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.
- How to use it??
- Create an instance of the *CountVectorizer* class.
- Call the ***fit()*** function in order to learn a vocabulary from one or more documents.
- Call the ***transform()*** function on one or more documents as needed to encode each as a vector.
- An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

Example of CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

In [66]: # list of text documents
    ...: text = ["The quick brown fox jumped over the lazy dog."]
    ...: # create the transform
    ...: count_vect = CountVectorizer()
    ...: # tokenize and build vocab
    ...: count_vect.fit(text)
    ...: # summarize
    ...: print(count_vect.vocabulary_)
    ...: # encode document
    ...: vector = count_vect.transform(text)
    ...: # summarize encoded vector
    ...: print(vector.shape)
    ...: print(type(vector))
    ...: print(vector.toarray())
    ...:
{'the': 7, 'quick': 6, 'brown': 0, 'fox': 2, 'jumped': 3, 'over': 5, 'lazy': 4, 'dog': 1}
(1, 8)
<class 'scipy.sparse.csr.csr_matrix'>
[[1 1 1 1 1 1 1 2]]
```

```
In [68]: text2 = ["the cute little puppy"]
    ...: count_vect.fit(text2)
    ...: print(count_vect.vocabulary_)
    ...: vector = count_vect.transform(text2)
    ...: print(vector.toarray())
    ...:
{'the': 3, 'cute': 0, 'little': 1, 'puppy': 2}
[[1 1 1 1]]
```

Step 4 :Term frequency – inverse document frequency

- **Term Frequency**: This summarizes how often a given word appears within a document.
- **Inverse Document Frequency**: This downscales words that appear a lot across documents.

```
In [69]:
    ...: from sklearn.feature_extraction.text import TfidfVectorizer
    ...: # list of text documents
    ...: text = ["The quick brown fox jumped over the lazy dog.",
    ...:         "The dog.",
    ...:         "The fox"]
    ...: # create the transform
    ...: vectorizer = TfidfVectorizer()
    ...: # tokenize and build vocab
    ...: vectorizer.fit(text)
    ...: # summarize
    ...: print(vectorizer.vocabulary_)
    ...: print(vectorizer.idf_)
    ...: # encode document
    ...: vector = vectorizer.transform([text[0]])
    ...: # summarize encoded vector
    ...: print(vector.shape)
    ...: print(vector.toarray())
    ...:
{'the': 7, 'quick': 6, 'brown': 0, 'fox': 2, 'jumped': 3, 'over': 5, 'lazy': 4, 'dog': 1}
[ 1.69314718  1.28768207  1.28768207  1.69314718  1.69314718  1.69314718
  1.69314718  1.        ]
(1, 8)
[[ 0.36388646  0.27674503  0.27674503  0.36388646  0.36388646  0.36388646
   0.36388646  0.42983441]]
```

Text processing on the Train and Test Data

```
In [1]: #Splitting the data set into train and test
   ...: X_train, X_test, y_train, y_test = train_test_split(df3['reviews.text'],
df3['reviews.rating'],test_size=0.25, random_state=42)
   ...:
   ...:
   ...: #Function for stemming the data/ text
   ...: stemmer = PorterStemmer()
   ...: from nltk.corpus import stopwords
   ...:
   ...: def stem_tokens(tokens, stemmer):
   ...:     stemmed = []
   ...:     for item in tokens:
   ...:         stemmed.append(stemmer.stem(item))
   ...:     return stemmed
   ...:
   ...: #Function for tokenization of the text
   ...:
   ...: def tokenize(text):
   ...:     tokens = nltk.word_tokenize(text)
   ...:     tokens = [word for word in tokens if word not in stopwords.words('english')]
   ...:     stems = stem_tokens(tokens, stemmer)
   ...:     return ' '.join(stems)
   ...:
   ...: intab = string.punctuation
   ...: outtab = "                                "
   ...: trantab = str.maketrans(intab, outtab)
```

Stops words hold importance in reviews.
So they were not removed, as accuracy reduced.

Training and Testing Modules

```
In [1]: #--- Training Data set---------
   ...:
   ...: corpus = []
   ...: for text in X_train:
   ...:    text = str(text).lower()
   ...:    text = text.translate(trantab)
   ...:    text=tokenize(text)
   ...:    corpus.append(text)
   ...:
   ...: count_vect = CountVectorizer()
   ...: X_train_counts = count_vect.fit_transform(corpus)
   ...: count_vect.get_feature_names()
   ...:
   ...: tfidf_transformer = TfidfTransformer()
   ...: X_train_tfidf = tfidf_transformer.fit_transform(X_train_count
   ...:
   ...: #--- Test Data set-------------------
   ...:
   ...: test_set = []
   ...: for text in X_test:
   ...:     text = str(text).lower()
   ...:     text = text.translate(trantab)
   ...:     text=tokenize(text)
   ...:     test_set.append(text)
   ...:
   ...: X_new_counts = count_vect.transform(test_set)
   ...: X_test_tfidf = tfidf_transformer.transform(X_new_counts)
```

```
In [4]: print(count_vect.vocabulary_)
{'bought': 1962, 'becaus': 1600, 'we': 15107, 'realli': 11101, 'enjoy': 4739, '
'day': 3644, 'movi': 8978, 'not': 9361, 'fan': 5131, 'of': 9503, 'resurg': 1147
'better': 1700, 'than': 13711, 'salon': 11829, 'for': 5492, 'fraction': 5571, '
'durabl': 4457, 'and': 968, 'love': 8214, 'them': 13741, 'have': 6484, 'been':
10684, 'mani': 8383, 'year': 15572, 'it': 7377, 'wa': 14969, 'superior': 13355,
9589, 'market': 8426, 'just': 7585, 'new': 9223, 'tube': 14262, 'two': 14316, '
9293, 'longer': 8155, 'hold': 6716, 'my': 9083, 'hair': 6352, 'in': 7047, 'plac
8923, 'wateri': 15087, 'will': 15300, 'buy': 2240, 'as': 1209, 'mix': 8797, 'wi
'sprayer': 12902, 'bottl': 1957, 'washer': 15062, 'mah': 8336, 'pickumup': 1021
'none': 9318, 'so': 12639, 'when': 15204, 'get': 5879, 'bug': 2158, 'gut': 6323
12901, 'down': 4323, 'real': 11088, 'good': 6041, 'wait': 14992, 'few': 5254, '
'wiper': 15337, 'usual': 14693, 'wash': 15057, 'most': 8942, 'dem': 3809, 'en':
13836, 'anoth': 1021, 'dose': 4306, 'sometim': 12699, 'too': 14040, 'wallyworld
715, 'store': 13122, 'brand': 2006, 'later': 7846, 'tri': 14185, 'name': 9110,
'definit': 3761, 'say': 11923, 'that': 13720, 'truli': 14236, 'quilt': 10957, '
10378, 'soft': 12656, 'touch': 14091, 'absorb': 531, 'leav': 7918, 'me': 8523,
'like': 8030, 'singl': 12405, 'pli': 10356, 'tissu': 13961, 'out': 9748, 'there
4239, 'care': 2390, 'scratchi': 11996, 'kind': 7682, 'best': 1690, 'way': 15100
904, 'review': 11525, 'collect': 2961, 'part': 9955, 'promot': 10724, 'long': 8
'bath': 1529, 'am': 908, 'happi': 6429, 'forget': 5512, 'peac': 10020, 'mind':
12972, 'white': 15234, 'cloth': 2867, 'put': 10874, 'pod': 10391, 'let': 7972,
'loader': 8128, 'deterg': 3924, 'dispens': 4172, 'but': 2221, 'doesn': 4252, 'g
332, 'isnt': 7369, 'notic': 9372, 'anim': 996, 'compar': 3038, 'live': 8108, 'a
'own': 9843, 'one': 9598, 'funniest': 5734, 'sequel': 12128, 'seri': 12135, 'da
'eye': 5059, 'within': 15362, 'olay': 9552, 'total': 14081, 'effect': 4571, 'ma
'did': 3985, 'reduc': 11225, 'fine': 5305, 'line': 8053, 'make': 8354, 'look':
```

Text after Transformation

Classification works by learning from **labeled feature sets**, or training data, to later classify an **unlabeled feature set**

A **feature set** is basically a key-value mapping of feature names to feature values

```
   ...: df3 = pd.DataFrame({'Before': X_train, 'After': corpus})
   ...: print(df3.head(5))
   ...:
   ...: prediction = dict()
                                                   After  \
62822  bought becaus we realli enjoy the first indepe...
70569  these nail are better than salon nail for a fr...
21609  i have been use thi product for mani year and ...
957    i mix it with water in a zep sprayer bottl my ...
45328  after use store brand for year and later tri t...

                                                  Before
62822  Bought because we really enjoyed the first ind...
70569  These nails are better than salon nails for a ...
21609  I have been using this product for many years ...
957    I mix it with water in a Zep sprayer bottle. M...
45328  After using store brands for years and later t...
```

Strategy Overview

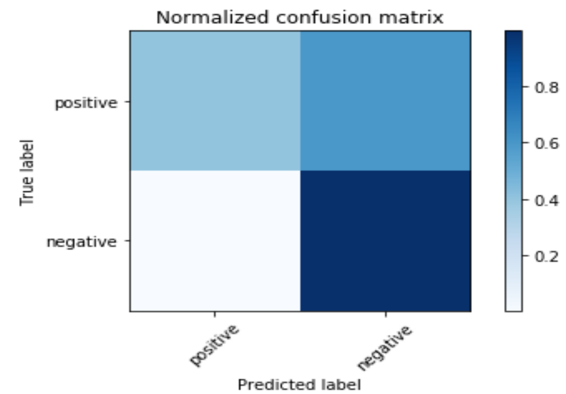
- Predicting sentiment as positive/negative on reviews title and reviews text
- Predicting ratings from 1 to 5 (very poor, poor, average, good, very good)
- Five different models implemented as follows:
- Naïve Bayes – Multinomial
- Naïve Bayes – Bernoulli
- SVC – Support Vector Classification
- Logistic Regression
- Decision Tree Classifier
- Score Measure used is Accuracy and AUC

```
In [14]: #---------Naive Bayes Multinomial Model-------------------------
    ...:
    ...: from sklearn.naive_bayes import MultinomialNB
    ...: model = MultinomialNB().fit(X_train_tfidf, y_train)
    ...: prediction['Multinomial'] = model.predict(X_test_tfidf)
    ...:
```

```
In [15]: #------Accuarcy Score ---------------
    ...:
    ...: print("Accuracy score for multinomial model")
    ...: accuracy_score(y_test,prediction['Multinomial'])
    ...:
Accuracy score for multinomial model
Out[15]: 0.94721903496587956
```

The confusion matrix

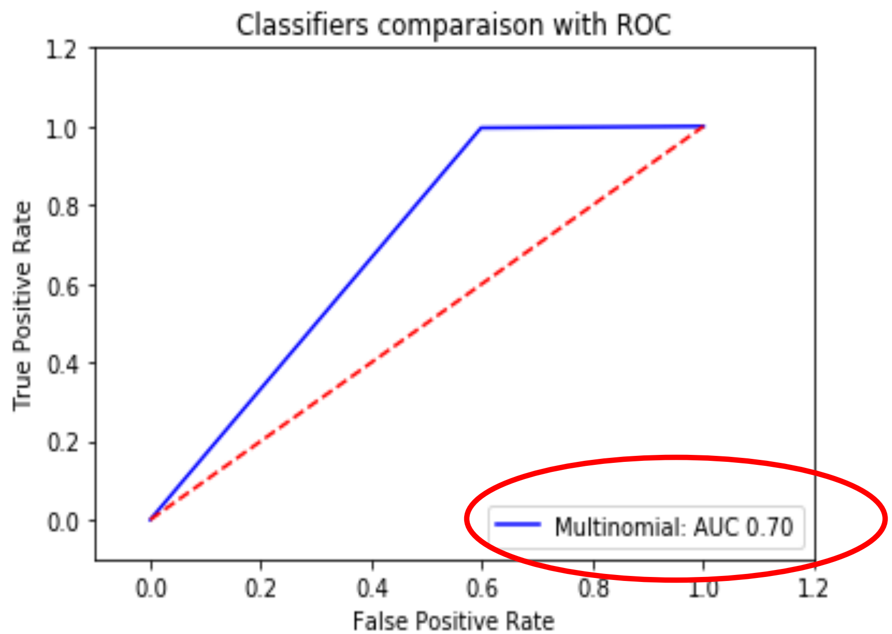
	precision	recall	f1-score	support
positive	0.90	0.40	0.56	1360
negative	0.95	1.00	0.97	15199
avg / total	0.94	0.95	0.94	16559



Naïve Bayes Multinomial Analysis

ROC

Area under the curve

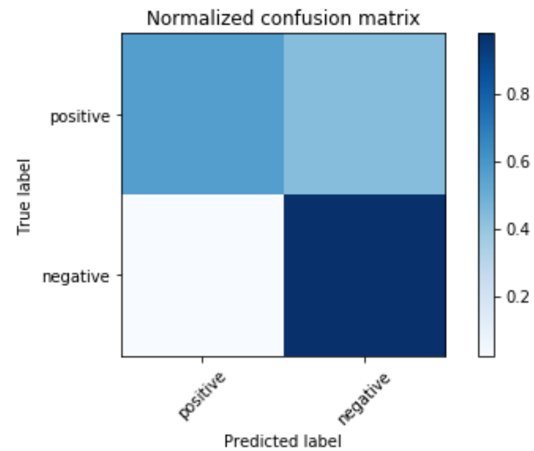


```
In [18]: #---------Naive Bayes Bernoulli Model--------------------------
    ...:
    ...: from sklearn.naive_bayes import BernoulliNB
    ...: model = BernoulliNB().fit(X_train_tfidf, y_train)
    ...: prediction['Bernoulli'] = model.predict(X_test_tfidf)
    ...:
```

```
In [19]: #------Accuarcy Score ---------------
    ...:
    ...: print("Accuracy score for Bernoulli model")
    ...: accuracy_score(y_test,prediction['Bernoulli'])
    ...:
Accuracy score for Bernoulli model
Out[19]: 0.94395796847635727
```

The confusion matrix

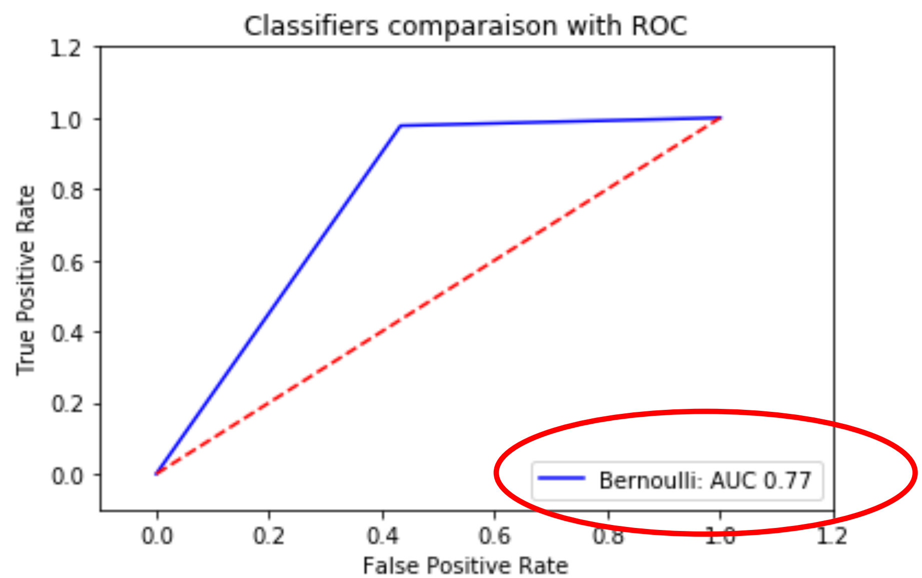
	precision	recall	f1-score	support
positive	0.69	0.57	0.62	1360
negative	0.96	0.98	0.97	15199
avg / total	0.94	0.94	0.94	16559



Naïve Bayes Bernoulli Analysis

ROC

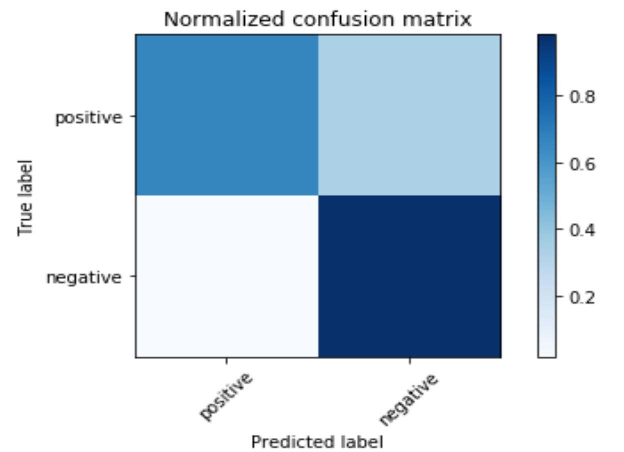
Area under the curve



```
In [2]: #----------Logistic Regression Model---------------------
   ...:
   ...: from sklearn import linear_model
   ...: logreg = linear_model.LogisticRegression(C=1e5)
   ...: logreg.fit(X_train_tfidf, y_train)
   ...: prediction['Logistic'] = logreg.predict(X_test_tfidf)
   ...:
```

```
In [3]: #------Accuarcy Score ---------------
   ...:
   ...: print("Accuracy score for Logistic  model")
   ...: accuracy_score(y_test,prediction['Logistic'])
   ...:
Accuracy score for Logistic  model
Out[3]: 0.95609638263180141
```

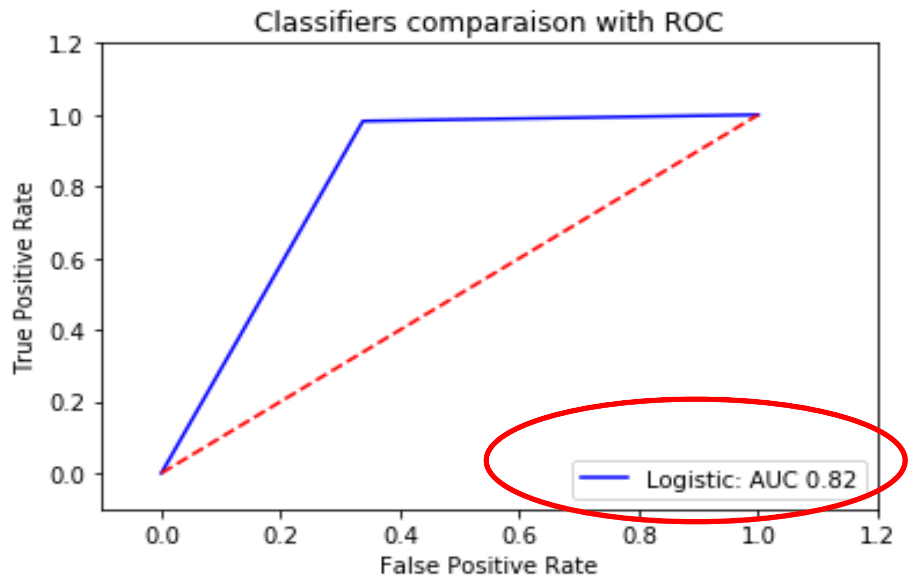
	precision	recall	f1-score	support
positive	0.77	0.66	0.71	1360
negative	0.97	0.98	0.98	15199
avg / total	0.95	0.96	0.95	16559



Logistic Regression Analysis

ROC

Area under the curve

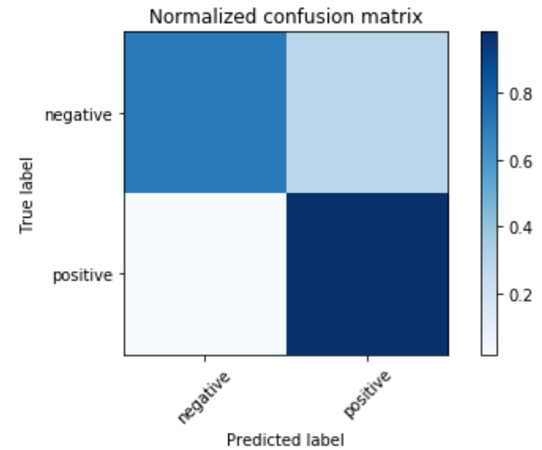


```
In [2]: #------Decision Tree Classifier Model-------
   ...:
   ...: from sklearn.tree import DecisionTreeClassifier
   ...: model = DecisionTreeClassifier()
   ...: model.fit(X_train_tfidf, y_train)
   ...: prediction['DTC'] = model.predict(X_test_tfidf)
   ...:
```

```
In [3]: #------Accuarcy Score ---------------
   ...:
   ...: print("Accuracy score for Decision Tree Classifier  model")
   ...: accuracy_score(y_test,prediction['DTC'])
   ...:
Accuracy score for Decision Tree Classifier  model
Out[3]: 0.95863276767920769
```

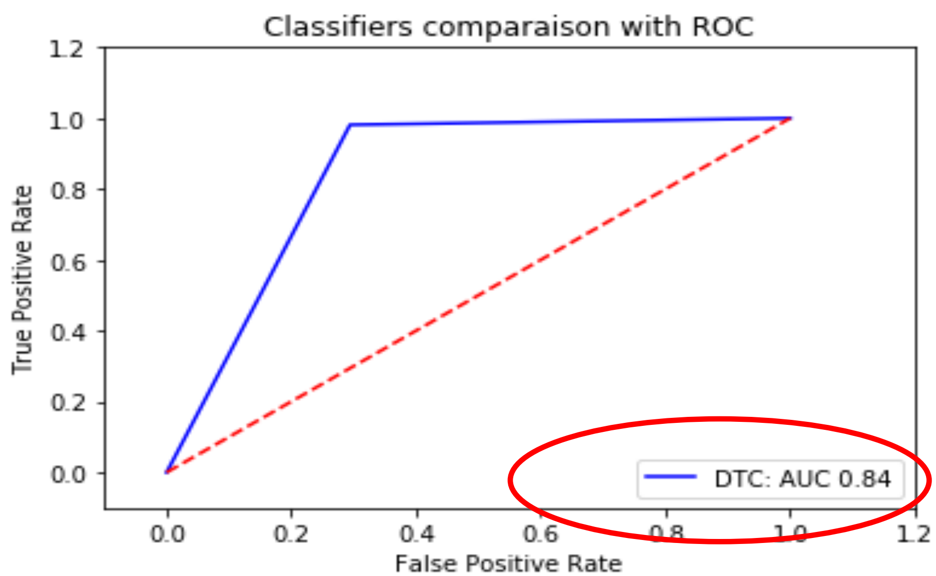
The confusion matrix

	precision	recall	f1-score	support
positive	0.77	0.71	0.74	1360
negative	0.97	0.98	0.98	15199
avg / total	0.96	0.96	0.96	16559



Decision Tree Analysis

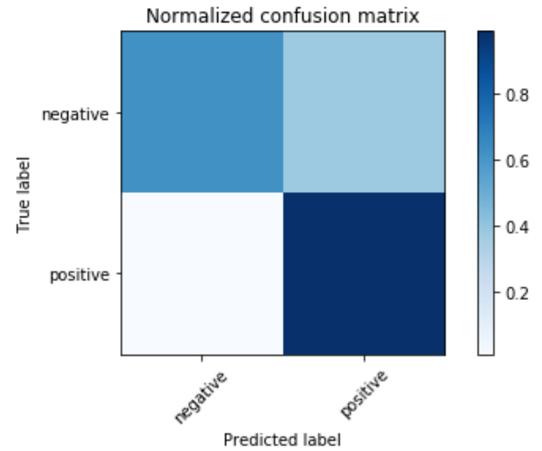
ROC Area under the curve



```
In [2]: #------------------Svc Model--------------------------
   ...: from nltk.classify.scikitlearn import SklearnClassifier
   ...: from sklearn.svm import LinearSVC
   ...: model =LinearSVC().fit(X_train_tfidf, y_train)
   ...: prediction['svc']= model.predict(X_test_tfidf)
   ...:
```

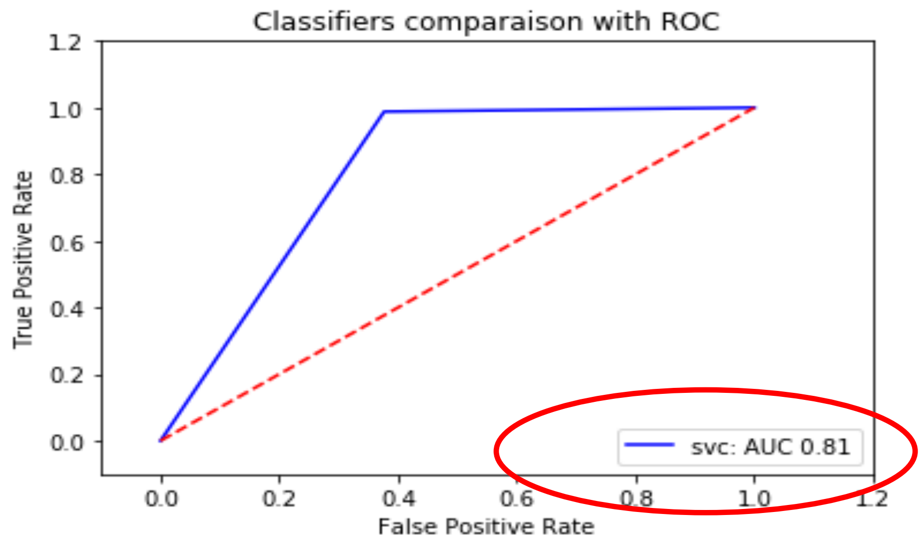
```
In [3]: #------Accuarcy Score ---------------
   ...:
   ...: print("Accuracy score for SVC  model")
   ...: accuracy_score(y_test,prediction['svc'])
   ...:
Accuracy score for SVC  model
Out[3]: 0.9580288664774434
```

```
The confusion matrix
             precision    recall  f1-score   support

   positive       0.82      0.62      0.71      1360
   negative       0.97      0.99      0.98     15199

avg / total       0.96      0.96      0.96     16559
```

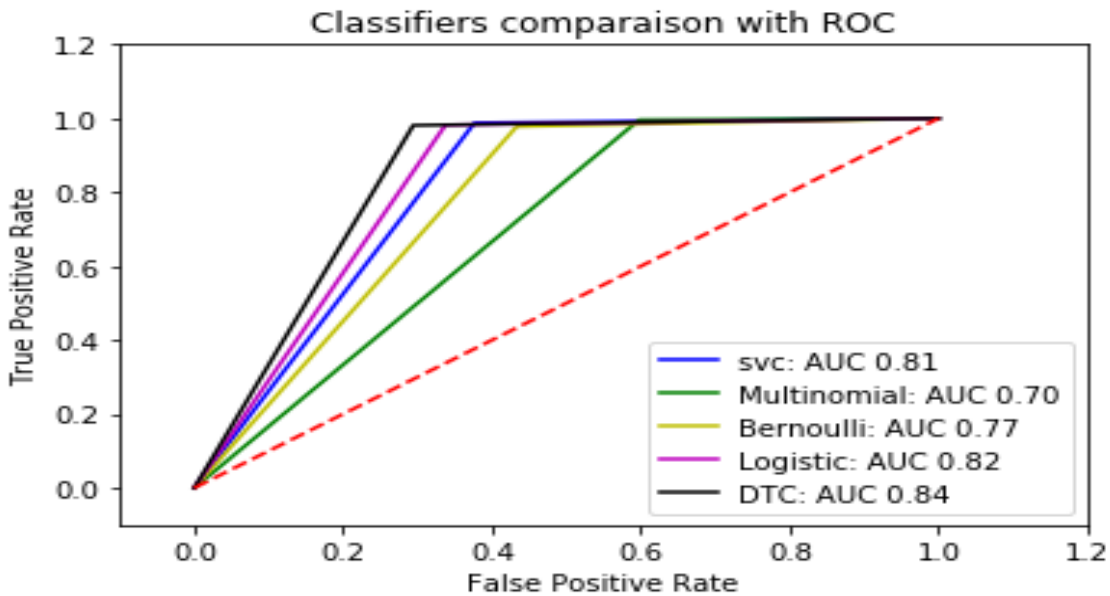


SVC Analysis

ROC Area under the curve



Comparison of the metrics Receiver Operator Characteristic – Area Under the Curve



Word cloud for Reviews - Title

```
show_wordcloud(cluster1["reviews.titleClean"][0], title = "Review Score negative")
```

```
show_wordcloud(cluster1["reviews.titleClean"][1], title = "Review Score positive")
```

Review Score negative

Review Score positive

Analysis of Review Text - Accuracy Score

```
In [11]: #------Accuarcy Score --------------
    ...:
    ...: print("Accuracy score for multinomial model")
    ...: accuracy_score(y_test,prediction['Multinomial'])
    ...:
Accuracy score for multinomial model
Out[11]: 0.92892082855244884
```

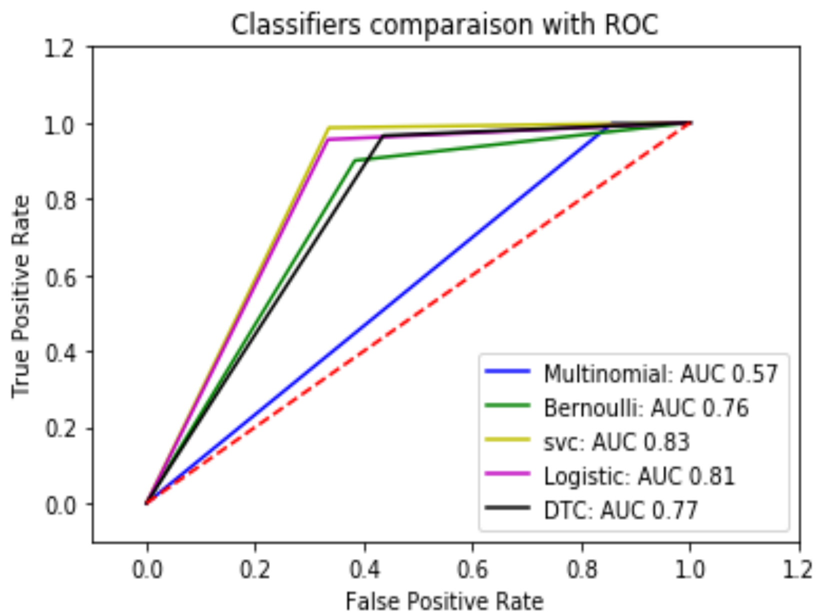
```
In [17]: #------Accuarcy Score --------------
    ...:
    ...: print("Accuracy score for Decision Tree Classifier  model")
    ...: accuracy_score(y_test,prediction['DTC'])
    ...:
Accuracy score for Decision Tree Classifier  model
Out[17]: 0.93514101093061175
```

```
In [13]: #------Accuarcy Score ---------------
    ...:
    ...: print("Accuracy score for Bernoulli model")
    ...: accuracy_score(y_test,prediction['Bernoulli'])
    ...:
Accuracy score for Bernoulli model
Out[13]: 0.87710610544114986
```

```
In [15]: #------Accuarcy Score ---------------
    ...:
    ...: print("Accuracy score for Logistic  model")
    ...: accuracy_score(y_test,prediction['Logistic'])
    ...:
Accuracy score for Logistic  model
Out[15]: 0.93187994444108946
```

```
In [19]: #------Accuarcy Score --------------
    ...:
    ...: print("Accuracy score for SVC  model")
    ...: accuracy_score(y_test,prediction['svc'])
    ...:
Accuracy score for SVC  model
Out[19]: 0.96014252068361616
```

Reviews Text – Area Under the Curve (ROC)



Word Cloud – Reviews Text

```
show_wordcloud(cluster1["reviews.textClean"][0], title = "Review Score negative")

show_wordcloud(cluster1["reviews.textClean"][1], title = "Review Score positive")
```

Review Score negative

Review Score positive

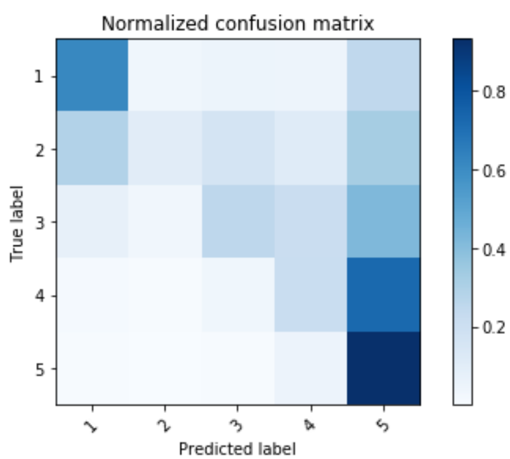
Multi-Label classification

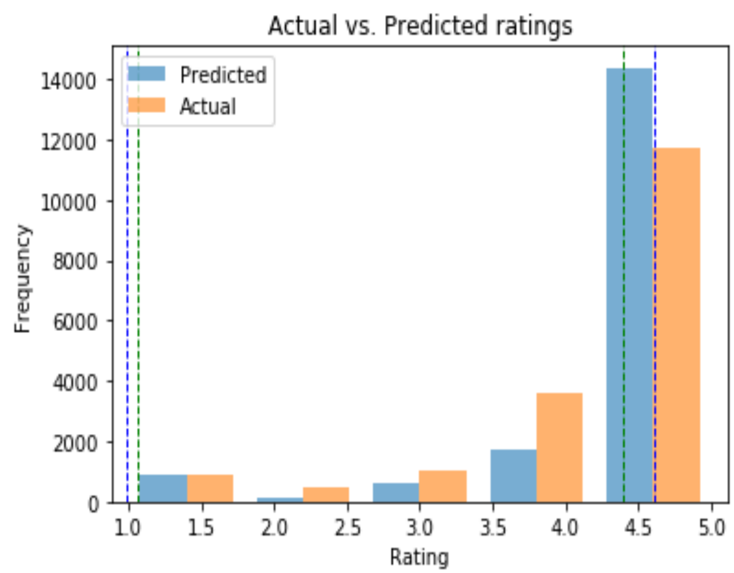
```
In [23]: accuracy_score(y_test,prediction['Multinomial'])
Out[23]: 0.6949496086932O421

In [24]: accuracy_score(y_test,prediction['Bernoulli'])
Out[24]: 0.6871234727774З369

In [25]: accuracy_score(y_test,prediction['svc'])
Out[25]: 0.7098699397556443

In [26]: accuracy_score(y_test,prediction['Logistic'])
Out[26]: 0.7007488317O992619

In [27]: accuracy_score(y_test,prediction['DTC'])
Out[27]: 0.6897134170373290２
```

Confusion Matrix for SVC – Best Model

confusion matrix

	precision	recall	f1-score	support
Rating1	0.63	0.62	0.62	917
Rating2	0.31	0.10	0.16	440
Rating3	0.47	0.26	0.33	1053
Rating4	0.45	0.22	0.29	3607
Rating5	0.76	0.93	0.84	11744
avg / total	0.66	0.71	0.67	17761

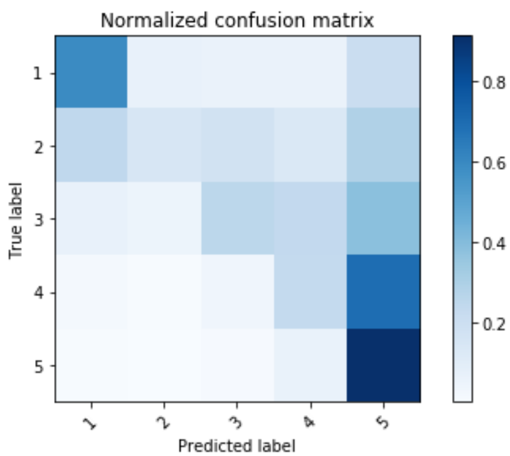


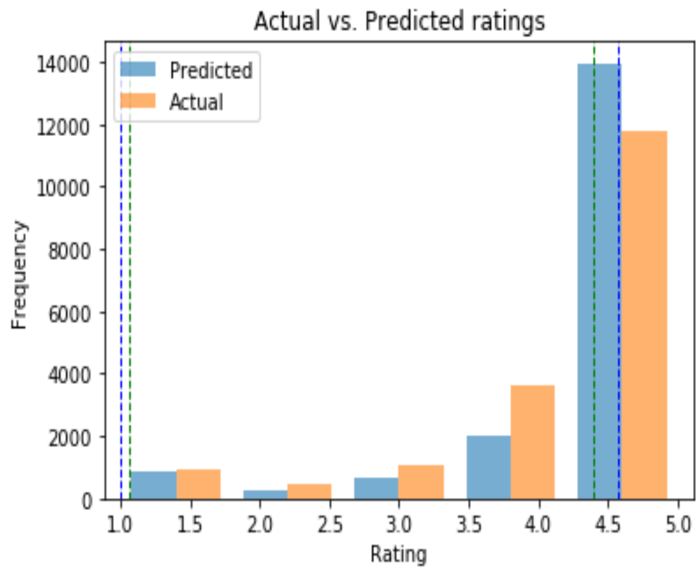


Confusion Matrix for Logistic – Best Model

confusion matrix

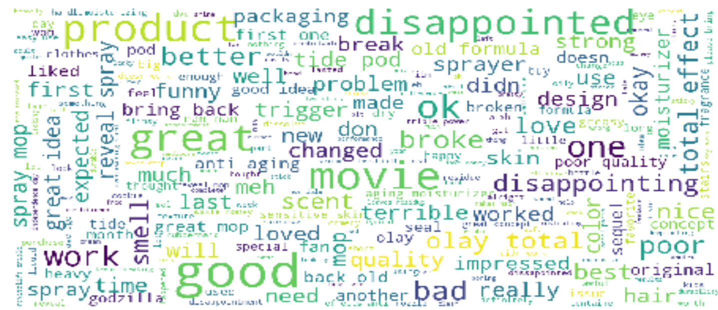
	precision	recall	f1-score	support
Rating1	0.62	0.60	0.61	917
Rating2	0.26	0.15	0.19	440
Rating3	0.40	0.26	0.31	1053
Rating4	0.43	0.24	0.30	3607
Rating5	0.77	0.91	0.83	11744
avg / total	0.66	0.70	0.67	17761



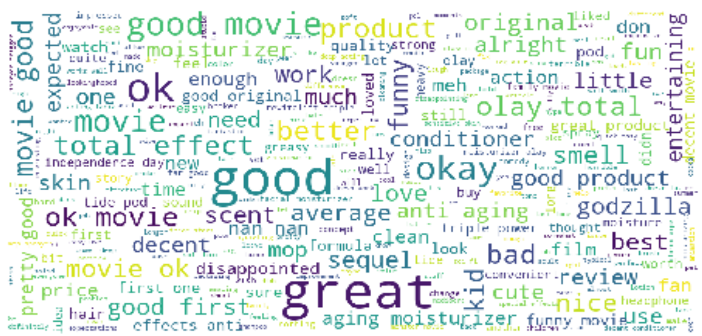


Word cloud for Multi - class

Review Score One



Review Score Two



Review Score Three

Review Score Four

Predictive Model Comparison for Review title

Model	Accuracy	Area under curve
Multinomial	94.72	.70
Bernoulli	94.39	.77
Support Vector Classification	95.8	.82
Logistic Regression	95.6	.81
Decision Tree Classifier	95.86	.84

Decision Tree Classifier yields the best performing model in terms of both accuracy and AUC

Predictive Model Comparison for Reviews text

Model	Accuracy	Area under curve
Multinomial	92.8	0.54
Bernoulli	87.7	0.76
Support Vector Classifier	96.0	0.83
Logistic Regression	93.1	0.81
Decision Tree Classifier	93.5	0.77

- SVC provides an accuracy of 96% and AUC of 0.83

Predictive Model Comparison for multi class

Model	Accuracy
Multinomial	69.4
Bernoulli	68.7
Support Vector Classifier	70.9
Logistic Regression	70.1
Decision Tree Classifier	68.9

- Support Vector Classifier and Logistic Model provide better accuracy

Conclusion

- Our model is more **biased** towards positive reviews compared to negative ones.
- Tried under sampling the majority class, ROC improves for sentiment classification. Accuracy reduces for multi label classification.
- In conclusion, although our data was biased towards positive reviews, (SVC) model was fairly accurate with its predictions, achieving an accuracy ,precision and recall of 96% on the test set. [Binary classification]
- For Multi- label classification , the best model (SVC) achieved an accuracy of 71 % , with precision of 66% and recall of 67%

Background/Reference Slides

https://data.world/datafiniti/grammar-and-online-product-reviews

https://www.lexalytics.com/lexablog/machine-learning-vs-natural-language-processing-part-1

https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation

https://www.reviewtrackers.com/online-review-bias/