Ecommerce Sentiment Analyser

Dangeti Anu Preetham
adangeti@uci.edu

Introduction: Product reviews plays an important role in the present e-commerce market, where the companies and firms take important decisions about the sustainability of the product in future. Consumers share their experience with the products they have purchased in the e-commerce sites like Amazon, Alibaba, Walmart gives an opportunity for the manufacturers and the other consumers to know about the performance of the product. With the help of machine learning tools and algorithms, there are many ways to understand and improve the performance of a particular product in the market by the reviews of the consumers. One such method is to analyse the sentiment of the product review given by the consumers. We have developed a model to predict the sentiment of a product review by using machine learning tools.

We have collected the dataset of the reviews of the amazon products from the Kaggle. And we have used Google colab Notebook and Python with Pandas, NumPy, Matplotlib and Scikit-Learn.

Link: https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products

The problem we have here is a supervised leaning with classification. The output can be considered as discrete, where we used SVM as our baseline model and the model is tested with logistic regression, Multinomial Naïve Bayes and support vector machine algorithms.

Dataset Definition: The data contains 34660 rows and 21 columns which includes reviews.id, reviews.rating, reviews.userCity ect. of different products like firestick, echo, kindle etc,.

```
dataset.shape
```

```
(34660, 21)
```

```
[39] csv_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34660 entries, 0 to 34659
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    34660 non-null  object 
 1   name                  27900 non-null  object 
 2   asins                 34658 non-null  object 
 3   brand                 34660 non-null  object 
 4   categories            34660 non-null  object 
 5   keys                  34660 non-null  object 
 6   manufacturer          34660 non-null  object 
 7   reviews.date          34621 non-null  object 
 8   reviews.dateAdded     24039 non-null  object 
 9   reviews.dateSeen      34660 non-null  object 
 10  reviews.didPurchase   1 non-null      object 
 11  reviews.doRecommend   34066 non-null  object 
 12  reviews.id            1 non-null      float64
 13  reviews.numHelpful    34131 non-null  float64
 14  reviews.rating        34627 non-null  float64
 15  reviews.sourceURLs    34660 non-null  object 
 16  reviews.text          34659 non-null  object 
 17  reviews.title         34655 non-null  object 
 18  reviews.userCity      0 non-null      float64
 19  reviews.userProvince  0 non-null      float64
 20  reviews.username      34658 non-null  object 
dtypes: float64(5), object(16)
memory usage: 5.6+ MB
```

Before the text pre-processing, we clean the data using by eliminating the null vales in the dataset. Based on our data analysis we perform transform like renaming columns, choosing required columns, dropping the nulls for ratings and reviews column.

```
[224] #csv_df is our data frame with file data

      rename=csv_df.rename(columns={'reviews.rating': 'Ratings', 'reviews.text': 'Reviews'})
```

```
[251] requiredColumns=rename_df[['Ratings','Reviews']]
```

```
dropNan_df = requiredColumns.dropna(subset=['Ratings','Reviews'])

dropNan_df['Ratings']= dropNan_df['Ratings'].astype(int)
```

Text Pre-processing: Text pre-processing helps to transform the text in to digestible form so that the machine learning algorithms can be performed. It is a method of cleaning the text data and make it ready to feed to the model. Text pre-processing is achieved by the following steps

- Tokenisation
- Stop words and punctuation removal
- Stemming
- CountVectorizer
- TFIDF

Tokenisation: Tokenisation is a process of turning a meaningful piece of data, such as the reviews, ratings, sentences or an entire text document of the data into a smaller unit such as individual words or terms called as tokens. These tokens are considered as a first step of stemming and lemmatization.

Stop words and punctuation removal: Stopwords in English is used to remove the unwanted words that add no meaning to a sentence. By removing these words will help us to remove the low level information from the available text data and helps to give importance to focus on the most wanted words.

CountVectorizer: CountVectorizer is a tool available in the scikit-learn library of python which is used to transform the text data into a vector on which the number of times the word occurs in the data.

Stemming: The process of reducing the infected words to their word stem. For example, going, goes, gone will be converted to go. The stem word is the base word which helps to determine the word if it is positive or negative. This is available in the Nltk library. But for some of the words, the stemming cannot give a meaningful representation. For example, consider the words history and historical, the stemming will convert the words to histori which is not meaningful. So, here we use lemmatization. Stemming is used in mail spam classifier, positive and negative classification and sentiment analysis etc.,

```
def process_review(review):

    review_tokens = tokenize_review(review)
    print(review_tokens)
    review_clean = remove_stopwords_punctuations(review_tokens)
    print(review_clean)
    reviews_stem = get_stem(review_clean)
    print(reviews_stem)

    return reviews_stem
```

```
[20] droppingNan['reviews.text']

0        This product so far has not disappointed. My c...
1        great for beginner or experienced person. Boug...
2        Inexpensive tablet for him to use and learn on...
3        I've had my Fire HD 8 two weeks now and I love...
4        I bought this for my grand daughter when she c...
                              ...
29783    Bought this to streaming media in HD and 4K an...
29784    I have a bunch of sticks but to be honest, I l...
29785    We're an unplugged family - multiple devices (...
29786    I bought this to utilize playstaion vue to cut...
29787    If you are looking to add smart TV capabilitie...
Name: reviews.text, Length: 29754, dtype: object

[21]
droppingNan['reviews.text'].apply(process_review)

0        [thi, product, far, disappoint, My, children, ...
1        [great, beginn, experienc, person, bought, gif...
2        [inexpens, tablet, use, learn, step, nabi, He,...
3        [I, 've, fire, HD, 8, two, week, I, love, thi,...
4        [I, bought, grand, daughter, come, visit, I, s...
                              ...
29783    [bought, stream, media, HD, 4K, work, great, o...
29784    [I, bunch, stick, honest, I, like, box, better...
29785    [We, 're, unplug, famili, multipl, devic, kind...
29786    [I, bought, util, playstaion, vue, cut, cabl, ...
29787    [If, look, add, smart, TV, capabl, TV, replac,...
Name: reviews.text, Length: 29754, dtype: object
```

TFIDF: Term frequency can be defined as the number of repetitive words in a sentence to the number of words in a sentence.

Term Frequency = number of repetitive words in a sentence / number of words in a sentence

Inverse document Frequency: log**[(number of sentences) / (number of sentences containing words)]**

TFIDF Score of a word= (Term Frequency) * (Inverse document Frequency), TFIDF provides score of word to rank its importance.

Tagging:

The label is considered as the ratings with 4 and 5 are considered as positive and the ratings with less than 4 are considered as negative labels.

```
def tagging(rating):
  if rating<4 and rating>0:
    return 0
  if rating==5 or rating==4:
    return 1
```

Because the dataset contains more positive ratings when compared to negative ratings.

The data set is split into 20% for training and 80% for testing.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_reviews, X_rating, test_size=0.20)
```

Multinomial Naïve bayes: Multinomial Naïve Bayes algorithm is a probabilistic learning method that is mostly used in Natural language Processing (NLP). Multinomial naïve bayes works well for the data with discrete values like word count in text for its likelihood.

It is based on the bayes formula **P(A|B) = P(A) * P(B|A)/P(B)**
Where we are calculating the probability of class A when predictor B is already provided.
P(B) = prior probability of B
P(A) = prior probability of class A
P(B|A) = occurrence of predictor B given class A probability
This formula helps in calculating the probability of the tags in the text.
The Multinomial Naïve Bayes classifier becomes:

$$C_{NB} = argmax\left(logP(c_k) + \sum_{i=1}^{n} X_i \log p_{ki}\right)$$

When features x_i $are\ the\ number\ of\ occurences\ of\ n\ possible\ events$ $(words)$.
$Where\ p_{ki} = probability\ of\ ith\ event\ occuring\ in\ class\ k$
$x_{ith} = frequency\ of\ the\ ith\ event$
argmax= α

```
[246] from sklearn.naive_bayes import MultinomialNB
      from sklearn.pipeline import Pipeline
      nbm = Pipeline([("cv", CountVectorizer(ngram_range=(1,1))), ("tf-idf", TfidfTransformer()), ("multiNB", MultinomialNB())])
      nbm.fit(X_train, Y_train)

      Pipeline(steps=[('cv', CountVectorizer()), ('tf-idf', TfidfTransformer()),
                      ('multiNB', MultinomialNB())])
```

```
[247] prednbm = nbm.predict(X_test)
      np.mean(prednbm == Y_test)

      0.9311290788333815
```

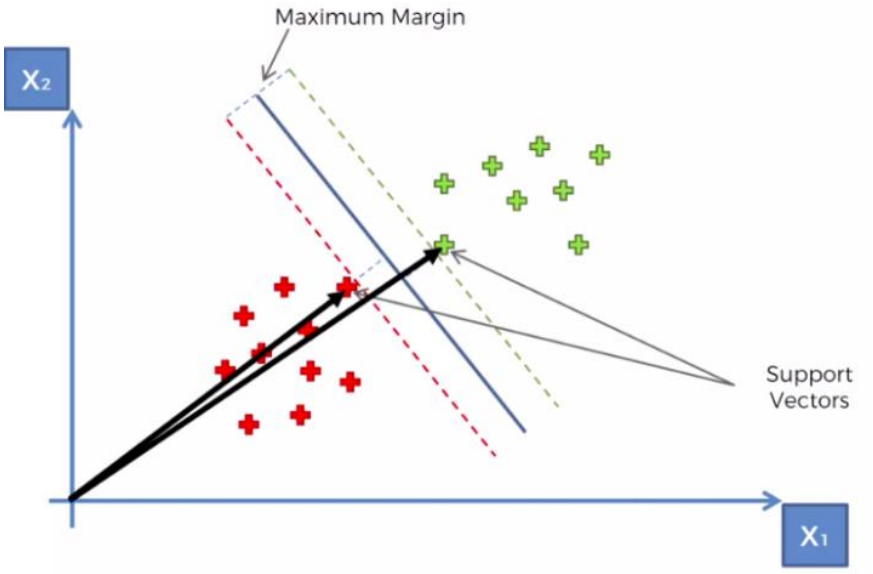
Logistic Regression: Logistic regression is a classification technique used for analysing a dataset in which there are one or more independent variables that determine an outcome. The sigmoid function, which is also called as logistic function gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1.

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
lrpipe = Pipeline([("cv", CountVectorizer()), ("tf-idf", TfidfTransformer()), ("lr", LogisticRegression())])
lrpipe.fit(X_train, Y_train)


predlr = lrpipe.predict(X_test)
np.mean(predlr == Y_test)
```

```
0.9371931850996246
```

Support Vector Machines: Support Vector Machines (SVM) are a set that comes under supervised leaning where we consider a hyper plane line which divides the data points and also, we have marginal distance which provides a cushion which is used for regression, classification and outliner detection. SVM performs well when there is a clear margin of separation of classes.



Mathematically the equation of a hyperplane is given by $w^t x + b = y$ where b is the y intercept, w^t is the slope of the hyperplane.

$\max(w^*, b^*) = \max\left(\frac{2}{||w||}\right)$ which is the optimising function.

Therefore $(w^*, b^*) = \min\left(\frac{2}{||w||}\right) + C_i \sum_{i=1}^{n} \beta_i$

C_i values give us the information about how many errors a model can consider and $\beta\ gives$ information about the value of the error. C is called the Regulization.

```
[243] from sklearn.svm import LinearSVC
      from sklearn.pipeline import Pipeline
      from sklearn.feature_extraction.text import CountVectorizer
      from sklearn.feature_extraction.text import TfidfTransformer

      svc_l = Pipeline([("cv", CountVectorizer(ngram_range=(1,1))),("tf-idf", TfidfTransformer()), ("clf_linearSVC", LinearSVC())])
      svc_l.fit(X_train, Y_train)

      pred = svc_l.predict(X_test)
      np.mean(pred == Y_test)

      0.938492636442391
```

```
[245] from sklearn.model_selection import GridSearchCV
      param = {'cv__ngram_range': [(1, 1), (1, 2)],
               'tf-idf__use_idf': (True, False),
               }
      svc_g = GridSearchCV(svc_l, param, n_jobs=-1)
      svc_g = svc_g.fit(X_train, Y_train)

      pred_g=svc_g.predict(X_test)
      np.mean(pred_g==Y_test)

      0.9423909904706902
```

Conclusion: The accuracy with Grid svm is 94.2% is the maximum among the models tested. When compared with the logistic regression and multinomial naive base, the Line svm has provided us with better accuracy. We further optimised linear svm with grid search svm to get the accuracy of 94.239%. So, we observed this model gives best result with the available dataset. If the dataset is more, we can improve the model to get the better results.