IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SENTIMENT ANALYSIS ON PRODUCT REVIEWS USING PYTHON

✍**Mrs. Tapasya Pandey,** Research Scholar, Comp.Sc. &Appl., RNTU, Bhopal (M.P.)

Research Guide – Dr. Pratima Gautam, Professor, Comp. Appl., RNTU, Bhopal (M.P.)

Abstract: Now, more than ever, it's key for companies to pay close attention to the voice of customer (VoC) to improve their products. Product managers need insights that will help them develop a robust product roadmap; it's about providing customers with what they actually want, rather than with what businesses think they need. A good place to start collecting product feedback is from online review sites (such as Capterra, G2Crowd, and Google Play). But manually analysing this unstructured data would take far too long. Amazon gives a platform to small businesses and companies with modest resources to grow larger. And because of its popularity, people actually spend time and write detailed reviews, about the brand and the product. So, by analysing that data we can tell companies a lot about their products and also the ways to enhance the quality of the product. But that large amount of data can not be analysed by a person. This paper shows the method for analysing Sentiment of Product Review by using Python Language. It is useful for all researcher or students to grow their knowledge about Sentiment Analysis and Use of Python for Sentiment Analysis.

Index Terms **- Sentiment, Analysis, Libraries and Dataset, Pre-process.**

I. INTRODUCTION

Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service, or idea. It involves the use of data mining, machine learning (ML) and artificial intelligence (AI) to mine text for sentiment and subjective information.

Sentiment analysis systems help organizations gather insights from unorganized and unstructured text that comes from online sources such as emails, blog posts, support tickets, web chats, social media channels, forums and comments. Algorithms replace manual data processing by implementing rule-based, automatic or hybrid methods. Rule-based systems perform sentiment analysis based on predefined, lexicon-based rules while automatic systems learn from data with machine learning techniques. A hybrid sentiment analysis combines both approaches.

In addition to identifying sentiment, opinion mining can extract the polarity (or the amount of positivity and negativity), subject and opinion holder within the text. Furthermore, sentiment analysis can be applied to varying scopes such as document, paragraph, sentence and sub-sentence levels.

Vendors that offer sentiment analysis platforms or SaaS products include Brandwatch, Hootsuite, Lexalytics, NetBase, Sprout Social, Sysomos and Zoho. Businesses that use these tools can review customer feedback more regularly and proactively respond to changes of opinion within the market.

Sentiment Analysis using Python

Whether you speak of Twitter, Goodreads or Amazon—hardly is there a digital space not saturated with peoples' opinions. In today's world, it is crucial for organizations to dig into these opinions and get insights about their products or services. However, this data exists in such astounding amounts that gauging it manually is a next to impossible pursuit. This is where Data Science's yet another boon comes to play—**Sentiment Analysis**. In this paper, we'll explore what sentiment analysis encompasses and the various ways to implement it in Python.

So here comes the Machine learning part, i.e. Natural Language Processing (NLP) to overcome the problem of large datasets and analyze it. Our task is to predict whether the review given is positive or negative. The real dataset after scraping the website might include millions of reviews.

Steps to be followed

1. Importing Libraries and Datasets
2. Preprocessing and cleaning the reviews
3. Analysis of the Dataset
4. Converting text into Vectors
5. Model training, Evaluation, and Prediction

(1) Importing Libraries and Datasets

The libraries used are :

- Pandas : For importing the dataset.
- Scikit-learn : For importing the model, accuracy module, and TfidfVectorizer.
- Warning : To ignore all the warnings
- Matplotlib : To plot the visualization. Also used Wordcloud for that.

Python3

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
from sklearn.feature_extraction.textimportTfidfVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

For NLP part, we will be using NLTK Library. From that we will berequiring stopword and punkt. so let's download and import them using the below command.

Python3

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
```

After that import the downloaded dataset using the below code.

Python3

```
data =pd.read_csv('AmazonReview.csv')
data.head()
```

Output :

	Review	Sentiment
0	Fast shipping but this product is very cheaply...	1
1	This case takes so long to ship and it's not e...	1
2	Good for not droids. Not good for iPhones. You...	1
3	The cable was not compatible between my macboo...	1
4	The case is nice but did not have a glow light...	1

(2) Preprocessing and cleaning the reviews

Python3

```
data.info()
```

Output:

Data columns (total 2 columns):

#	Column	Non-Null Count	Dtype
---	------	--------------	-----
0	Review	24999 non-null	object
1	Sentiment	25000 non-null	int64

Now, To drop the null values (if any), run the below command.Python3

```
data.dropna(inplace=True)
```

To predict the Sentiment as positive(numerical value = 1) or negative(numerical value = 0), we need to change them the values to those categories. For that the condition will be like if the sentiment value is less than or equal to 3, then it is negative(0) else positive(1). For better understanding, refer the code below.

Python3

```
#1,2,3->negative(i.e 0)
data.loc[data['Sentiment']<=3,'Sentiment'] =0
 #4,5->positive(i.e 1)
data.loc[data['Sentiment']>3,'Sentiment'] =1
```

Now, once the dataset is ready, we will clean the review column by removing the stopwords. The code for that is given below

Python3

```
stp_words=stopwords.words('english')
def clean_review(review):
  cleanreview=" ".join(word for word in review.
                split() if word not in stp_words)
  return cleanreview
data['Review']=data['Review'].apply(clean_review)
```

Once we have done with the preprocess. Let's see the top 5 rows to see the improved dataset.

Python3

```
data.head()
```

Output :

	Review	Sentiment
0	Fast shipping product cheaply made I brought g...	0
1	This case takes long ship even worth DONT BUY!!!!	0
2	Good droids. Not good iPhones. You cannot use ...	0
3	The cable compatible macbook iphone. Also conn...	0
4	The case nice glow light. I'm disappointed pro...	0

(3) Analysis of the Dataset

Let's check out that how many counts are there for positive and negative sentiments.

Python3

```
data['Sentiment'].value_counts()
```

Output :

0 15000

1 9999

To have the better picture of the importance of the words let's create the Wordcloud of all the words with sentiment = 0 i.e. negative

Python3

```
consolidated=' '.join(word for word in data['Review'][data['Sentiment']==0].astype(str))
wordCloud=WordCloud(width=1600,height=800,random_state=21,max_font_size=110)
plt.figure(figsize=(15,10))
plt.imshow(wordCloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.show()
```

Output :

Let's do the same for all the words with sentiment = 1 i.e. positive

Python3

```
consolidated=' '.join(word for word in data['Review'][data['Sentiment']==1].astype(str))
wordCloud=WordCloud(width=1600,height=800,random_state=21,max_font_size=110)
plt.figure(figsize=(15,10))
plt.imshow(wordCloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.show()
```

Output :

Now we have a clear picture of the words we have in both the categories.

Let’s create the vectors.

(4) Converting text into Vectors

TF-IDF calculates that how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set). We will be implementing this with the code below.

Python3

```
cv =TfidfVectorizer(max_features=2500)
X =cv.fit_transform(data['Review'] ).toarray()
```

(5) Model training, Evaluation, and Prediction

Once analysis and vectorization is done. We can now explore any machine learning model to train the data. But before that perform the train-test split.

Python3

```
from sklearn.model_selection import train_test_split
x_train ,x_test,y_train,y_test=train_test_split(X,data['Sentiment'],
                                                test_size=0.25,
                                                random_state=42)
```

Now we can train any model, Let’s explore the Logistic Regression.

Python3

```
from sklearn.linear_model import LogisticRegression
 model=LogisticRegression()
 #Model fitting
model.fit(x_train,y_train)
```

```
#testing the model
pred=model.predict(x_test)
#model accuracy
print(accuracy_score(y_test,pred))
```

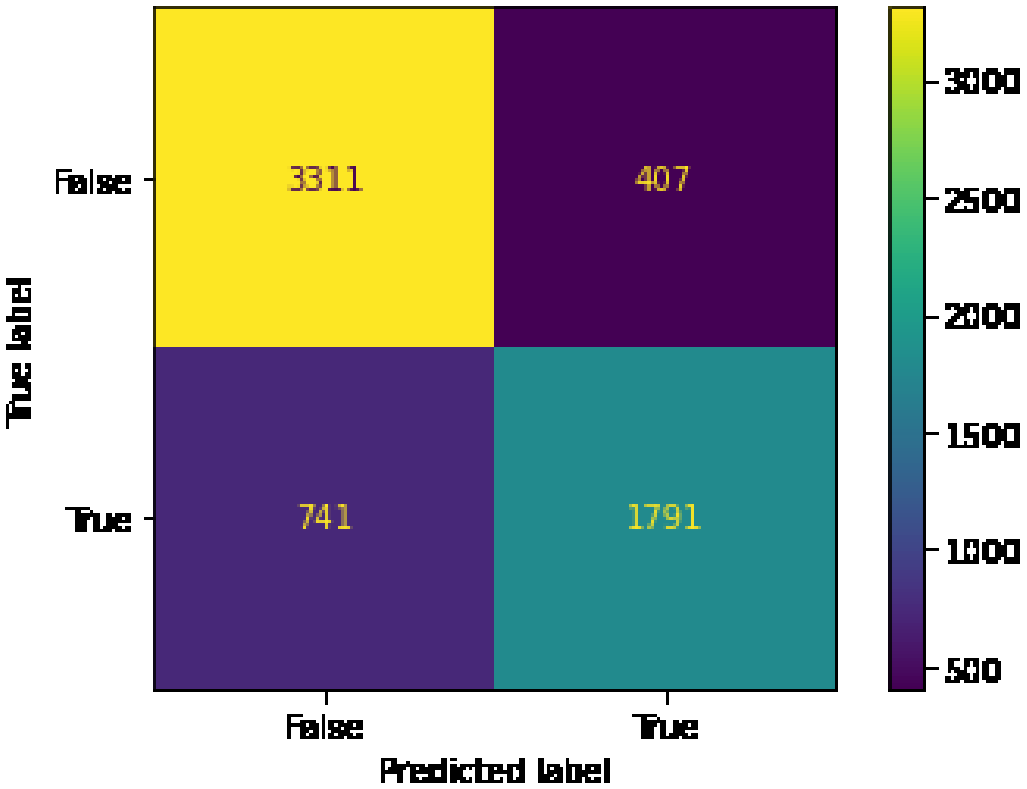
Output : 0.81632

Let's see the confusion matrix for the results.

Python3

```
fromsklearnimportmetrics
cm =confusion_matrix(y_test,pred)
cm_display=metrics.ConfusionMatrixDisplay(confusion_matrix=cm,
                                          display_labels=[False, True])
cm_display.plot()
plt.show()
```

Output :



Conclusion :

In this age when users can express their viewpoints effortlessly and data is generated in superfluity in just fractions of seconds—drawing insights from such data is vital for organizations to make efficient decisions—and Sentiment Analysis proves to be the missing piece of the puzzle!

By now we have covered in great detail what exactly sentiment analysis entails and the various methods one can use to perform it in Python. But these were just some rudimentary demonstrations—readers or researchers or students must surely go ahead and fiddle with the models and try them out on their own data.

References :

- Pang, L. Lee, and S. Vaithyanathan. Thumbs up: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pages 79_86. Association for Computational Linguistics, 2002.
- N. Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. In Proceedings of the International Conference on Weblogs and Social Media (ICWSM). Citeseer, 2007.
- P. Melville, W. Gryc, and R.D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1275_1284. ACM, 2009.
- Prof. B Nithya Ramesh, Aashay R Amballi, Vivekananda Mahanta, “Django Python Framework”, International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online) Vol. 6, Issue 2.