Capstone Project- Ecommerce Review-Sentiment Analysis Project

Submitted By- Diwakar Acharya

Table of Contents

Problem Statement

An e-commerce company has put forward a task to analyse the customer reviews for various products. We are supposed to create a report that classifies the products based on the customer reviews.

Project Objective

1) We have to find various trends and patterns in the review data, create useful insights that best describe the product quality.
2) Classify each review based on the sentiment associated with the same.

Data Description

Given data is E-commerce review data in CSV format. It is having 60145 rows and 10 columns. These columns represent features name.

Feature Name	Description
Id	Record ID
ProductId	Product ID
UserId	User ID who posted the review
ProfileName	Profile name of the User
HelpfulnessNumerator	Numerator of the helpfulness of the review
HelpfulnessDenominator	Denominator of the helpfulness of the review
Score	Product Rating
Time	Review time in timestamp
Summary	Summary of the review
Text	Actual text of the review

Insights from the data:

1) Basic details of data

```
ecom_review.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

2) All statistical figure

```
ecom_review.describe()
```

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
count	568454.000000	568454.000000	568454.00000	568454.000000	5.684540e+05
mean	284227.500000	1.743817	2.22881	4.183199	1.296257e+09
std	164098.679298	7.636513	8.28974	1.310436	4.804331e+07
min	1.000000	0.000000	0.00000	1.000000	9.393408e+08
25%	142114.250000	0.000000	0.00000	4.000000	1.271290e+09
50%	284227.500000	0.000000	1.00000	5.000000	1.311120e+09
75%	426340.750000	2.000000	2.00000	5.000000	1.332720e+09
max	568454.000000	866.000000	923.00000	5.000000	1.351210e+09

3) Correlation between features:

```
ecom_review.corr()
```

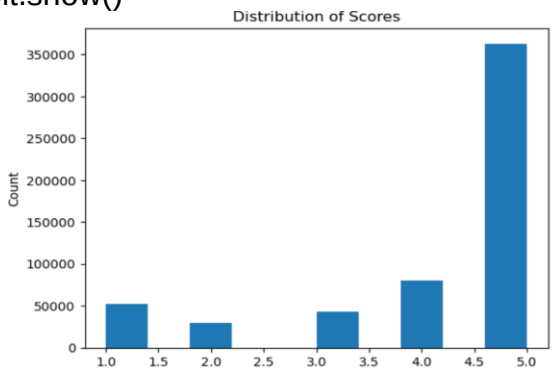
	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
Id	1.000000	0.001227	0.000770	0.010706	0.007912
HelpfulnessNumerator	0.001227	1.000000	0.974689	-0.032590	-0.154818
HelpfulnessDenominator	0.000770	0.974689	1.000000	-0.097986	-0.173289
Score	0.010706	-0.032590	-0.097986	1.000000	-0.062760
Time	0.007912	-0.154818	-0.173289	-0.062760	1.000000

1) **Various trends & patterns and other insights that best describe the product quality**

I have analysed below task and basis insights is provided.

1) **Distribution of scores:**
   Most of the rating has 5 star (350000) and least is 2 star (<50K)

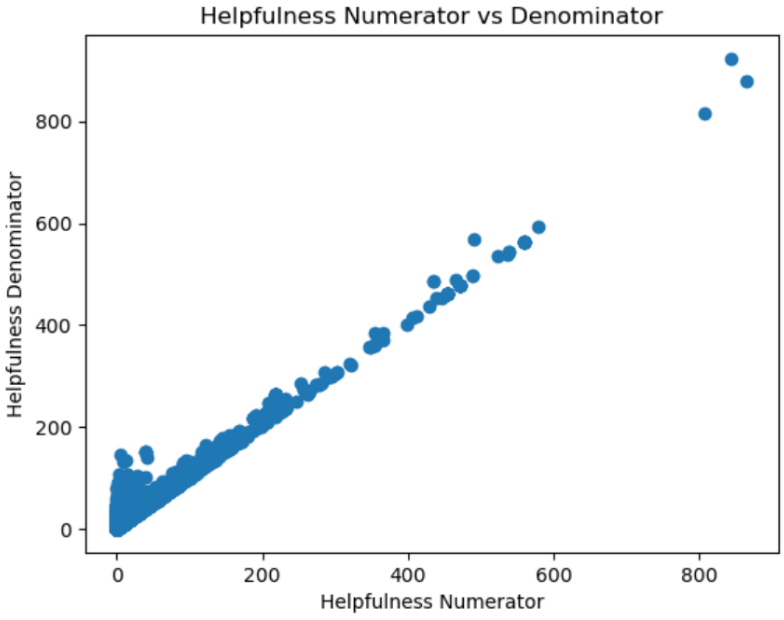
```
plt.hist(ecom_review["Score"])
plt.xlabel("Score")
plt.ylabel("Count")
plt.title("Distribution of Scores")
plt.show()
```



2) **Relationship between helpfulness numerator and denominator:**

   There seems to be linear relationship between these 2 features.

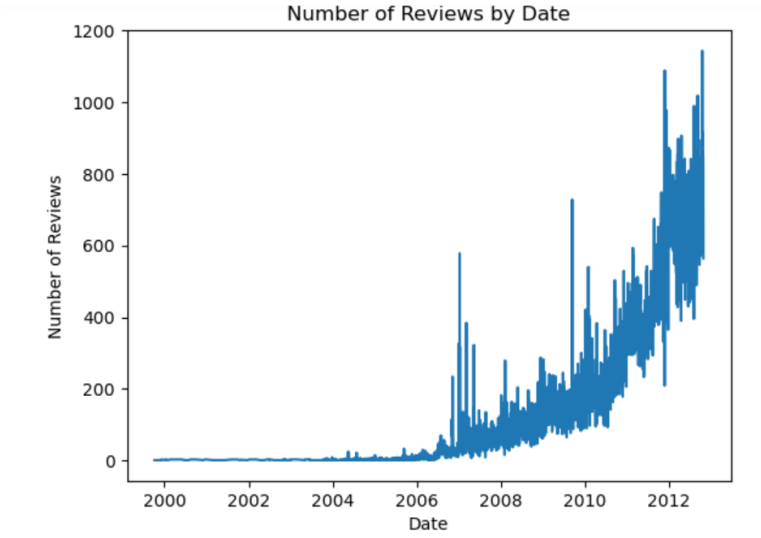
```
plt.scatter(ecom_review["HelpfulnessNumerator"],  ecom_review["Helpfulness Denominator"])
plt.xlabel("Helpfulness Numerator")
plt.ylabel("Helpfulness Denominator")
plt.title("Helpfulness Numerator vs Denominator")
plt.show()
```



3) **No. of reviews by date:**

Reviews started exponentially after 2007, it shows effect of internet penetratio n after 2007.

```
ecom_review["Time"] = pd.to_datetime(ecom_review["Time"], unit='s')
reviews_by_date = ecom_review.groupby(ecom_review["Time"].dt.date).count()["Id"]
plt.plot(reviews_by_date)
plt.xlabel("Date")
plt.ylabel("Number of Reviews")
plt.title("Number of Reviews by Date")
plt.show()
```



4) **Reviews per products**

ProductID B007JFMH8M is most reviewed.

```
reviews_per_product = ecom_review.groupby("ProductId").size().reset_index(name=
"Reviews").sort_values(by="Reviews",ascending=False)
print(reviews_per_product.head())
```

	ProductId	Reviews
71170	B007JFMH8M	913
37898	B0026RQTGE	632
42257	B002QWHJOU	632
42263	B002QWP89S	632
42264	B002QWP8H0	632

5) **Reviews per user:**

UserID 182011 has given the most reviews.

```
reviews_per_user = ecom_review.groupby("UserId").size().reset_index(name="Reviews").sort_values(by='Reviews', ascending=False)
print(reviews_per_user.head())
```

```
                UserId  Reviews
182011  A3OXHLG6DIBRW8      448
65586   A1YUL9PCJR3JTY      421
252275   AY12DBB0U420B      389
82919   A281NPSIMI1C2R      365
66162   A1Z54EM24Y40LL      256
```

6) **Average score per product:**

```
average_score_per_product = ecom_review.groupby("ProductId")["Score"].mean().reset_index().sort_values(by='Score',ascending=False)
average_score_per_product.head()
```

	ProductId	Score
37129	B00214BO58	5.0
42070	B002PDYN44	5.0
42053	B002P9QAJE	5.0
42054	B002P9QRCO	5.0
42057	B002PA1KW0	5.0

Data pre-processing Steps and Inspiration

1) Data cleaning- The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc
   Here, we have 16 missing data for ProfileName and 27 missing values for Summary.

```
ecom_review.isnull().sum()
```

```
Id                      0
ProductId               0
UserId                  0
ProfileName            16
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                27
Text                    0
dtype: int64
```

2) Removing duplicates-

   ```
   ecom_review = ecom_review.drop_duplicates()
   ```

3) Removing null values-
   ```
   ecom_review = ecom_review.dropna()
   ```

4) Removing special character and numbers-

   ```
   import re

   def clean_text(text):
       text = re.sub(r"[^a-zA-Z]", " ", text)
       text = re.sub(r"\s+", " ", text)
       return text

   ecom_review["Text"] = ecom_review["Text"].apply(clean_text)
   ```

5) Lowering the text- to make the text consistent

   ```
   ecom_review["Text"] = ecom_review["Text"].str.lower()
   ```

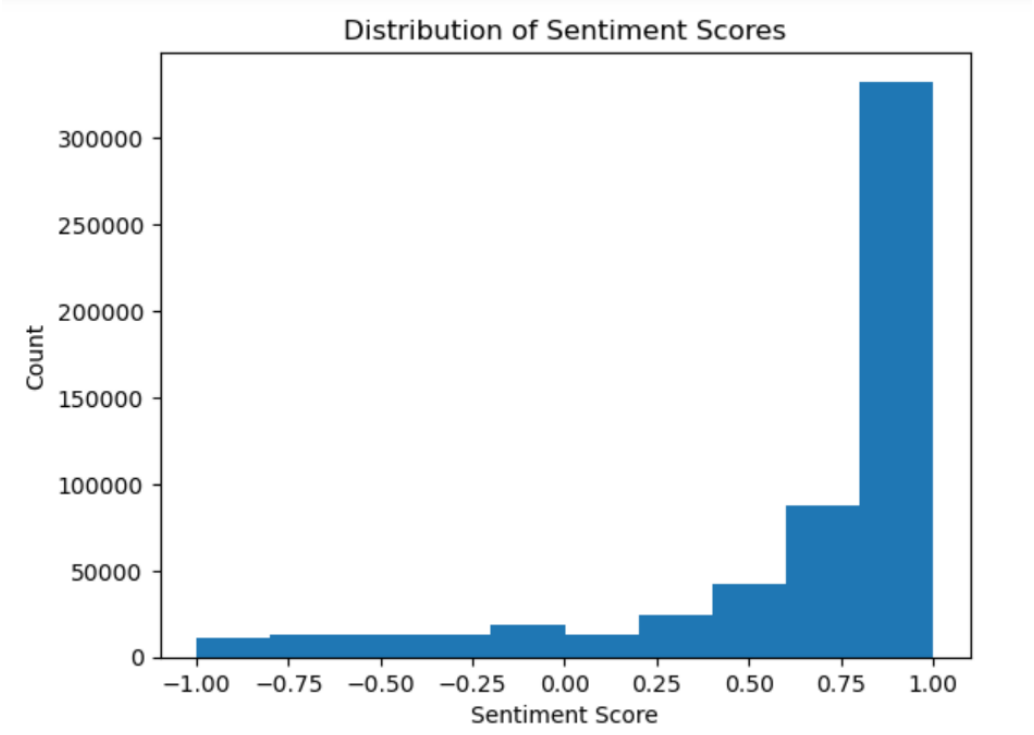
Choosing the Algorithm for the project

I have chosen SentimenIntensityAnalyzer algorithm from nltk

1) SentimenIntensityAnalyzer

Sentiment Analysis

```
import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

# Create an instance of the SentimentIntensityAnalyzer

sia = SentimentIntensityAnalyzer()

# Add a new column to the dataframe with the sentiment scores

ecom_review["Sentiment"] = ecom_review["Text"].apply(lambda x:
sia.polarity_scores(x)["compound"])

# Plot the distribution of sentiment scores

plt.hist(ecom_review["Sentiment"])

plt.xlabel("Sentiment Score")

plt.ylabel("Count")

plt.title("Distribution of Sentiment Scores")

plt.show()
```



Classify each Review based on the sentiment associated with the same.

```
# Function to classify a review as positive, negative, or neutral

def classify_review(text):

    sentiment = sia.polarity_scores(text)["compound"]

    if sentiment > 0.5:

        return "Positive"

    elif sentiment < -0.5:

        return "Negative"

    else:

        return "Neutral"


# Add a new column to the dataframe with the sentiment labels

ecom_review["Sentiment_Label"] = ecom_review["Text"].apply(classify_review)


# Print the first few rows of the dataframe to see the new column

print(ecom_review.head())
```

```
   Id   ProductId          UserId                      ProfileName  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
3   4  B000UA0QIQ  A395BORC6FGVXV                             Karl
4   5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

   HelpfulnessNumerator  HelpfulnessDenominator  Score       Time  \
0                     1                       1      5 2011-04-27
1                     0                       0      1 2012-09-07
2                     1                       1      4 2008-08-18
3                     3                       3      2 2011-06-13
4                     0                       0      5 2012-10-21

                Summary                                                Text  \
0  Good Quality Dog Food  I have bought several of the Vitality canned d...
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
2  "Delight" says it all  This is a confection that has been around a fe...
3         Cough Medicine  If you are looking for the secret ingredient i...
4            Great taffy  Great taffy at a great price.  There was a wid...

   Sentiment Sentiment_Label
0     0.9441        Positive
1    -0.5664        Negative
2     0.8265        Positive
3     0.0000         Neutral
4     0.9468        Positive
```

Motivation and Reasons for choosing the Algorithm

The motivation for choosing SentimentIntensityAnalyzer algorithm from NLTK library for sentiment analysis is:

1) It is a pre-trained model that can quickly and accurately classify text into positive, negative, and neutral categories. It uses a combination of lexical heuristics and a pre-trained model to calculate a compound sentiment score between -1 and 1. The compound score is the sum of the valence scores of each word in the text, adjusted according to the grammar and context of the text. This algorithm can handle negations, sarcasm, and other forms of linguistic nuances which makes it more robust.

2) Additionally, NLTK library provides a wide range of functionalities for natural language processing tasks like tokenization, stemming, lemmatization and stop-word removal, which can be used in conjunction with SentimentIntensityAnalyzer to improve the accuracy of sentiment analysis.

3) Another motivation for choosing this algorithm is that it is easy to use and understand, and it can be easily integrated into a wide range of applications and projects.

4) In summary, SentimentIntensityAnalyzer is a fast, accurate, and easy-to-use algorithm that can handle a wide range of text inputs, which makes it suitable for sentiment analysis of the reviews in above dataset.

Assumptions

The SentimentIntensityAnalyzer algorithm makes several assumptions when classifying text into positive, negative, and neutral categories. Some of the main assumptions are:

1) The algorithm assumes that the text input is in English, and it may not work as well for text in other languages.

2) The algorithm is based on a pre-trained model which is trained on a large amount of text data from different sources. The model may not work well for text that is significantly different from the training data.

3) The algorithm assumes that the text input is in a single sentence and it may not work well for multi-sentence text input.

4) It assumes that the text input is relatively clean, without any spelling errors, grammatical errors, or special characters.

5) The algorithm uses a compound score to classify text. The compound score is the sum of the valence scores of each word in the text, adjusted according to the grammar and context of the text. It may not be able to pick up on more subtle or nuanced emotions, such as sarcasm or irony.

6) The algorithm assumes that the text is providing a personal opinion about a subject, which means that it may not work well for text that does not have an opinion, such as news articles, or text that does not have a clear subject, such as short phrases.

Model Evaluation and techniques

There are several techniques that can be used to evaluate the performance of a sentiment analysis model, such as the SentimentIntensityAnalyzer algorithm. Some of the most common evaluation techniques are:

1) Accuracy: This is the most basic and widely used evaluation metric for classification models. It measures the proportion of correct predictions made by the model. However, accuracy may not always be the best metric to evaluate a sentiment analysis model, as it does not take into account the balance of positive, negative, and neutral examples in the dataset.

2) Confusion Matrix: A confusion matrix is a table that is used to define the performance of a classification algorithm. It is used to define the number of true positive, false positive, true negative, and false negative predictions made by the algorithm. The confusion matrix can be used to calculate other metrics like precision, recall, and F1 score.

3) Precision and Recall: Precision is the proportion of true positive predictions among all positive predictions made by the model. Recall is the proportion of true positive predictions among all actual positive examples. A high precision means that the model has a low rate of false positives, and a high recall means that the model has a low rate of false negatives.

4) F1 Score: The F1 score is the harmonic mean of precision and recall. It is a balance between precision and recall, and it is commonly used to evaluate the performance of a sentiment analysis model.

5) ROC and AUC: Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) are widely used for binary classification problems. It shows the trade-off between sensitivity and specificity.

6) Cross-validation: Cross-validation is a technique used to evaluate the performance of a model by training it on different subsets of the data and testing it on the remaining data. This technique can help to reduce the bias and variability in the model evaluation, and it can be useful to check the robustness of the model.

7) Human evaluation: Human evaluation is a technique where a sample of the model's predictions is evaluated by human annotators. This technique is considered to be the gold standard for evaluating the performance of a sentiment analysis model, as it takes into account the nuances and subtleties of natural language. However, it is also time-consuming and expensive.

Inferences from the same

Inference refers to drawing conclusions or making predictions based on the output of a model. Once we have evaluated the performance of a sentiment analysis model, like SentimentIntensityAnalyzer algorithm, we can use the results to make inferences about the dataset and the model. Some of the inferences that we can make are:

1) Model performance: We can infer the overall performance of the model based on the evaluation metrics that we have used. E.g., if the model has a high accuracy, precision, recall, and F1 score, we can infer that the model is performing well.

2) Class imbalance: If the dataset has a class imbalance, where one class has more examples than the others, we can infer that the model may be biased towards the majority class. We can use techniques like oversampling or weighting to balance the classes and improve the model's performance.

3) Model robustness: We can infer the robustness of the model by evaluating its performance on different subsets of the data. If the model's performance is consistent across different subsets, we can infer that the model is robust.

4) Model limitations: Based on the evaluation metrics and the confusion matrix, we can infer the limitations of the model. E.g., if the model has a high rate of false negatives, we can infer that the model is not very good at detecting negative sentiment.

5) Data quality: We can infer the quality of the data based on the model's performance. If the model's performance is poor, we may need to improve the quality of the data, e.g. by removing duplicate or irrelevant examples.

6) Use case: We can infer the use case of the model based on its performance on specific subsets of the data or specific classes. E.g., if the model performs well on a specific subset of the data, we can infer that the model is suitable for that use case.

It's important to note that, inferences should be drawn carefully and cautiously, since they are based on the assumptions and limitations of the model. It's always a good practice to validate the inferences with domain experts and real-world data.

Future possibilities of the project

There are several future possibilities for a sentiment analysis project, like the one using SentimentIntensityAnalyzer algorithm, depending on the specific use case and the goals of the project. Some of the possibilities are:

1) Fine-tuning the model: The SentimentIntensityAnalyzer algorithm is a pre-trained model, and it may not work well for text that is significantly different from the training data. One possible future direction is to fine-tune the model by training it on a labeled dataset that is specific to the use case of the project.

2) Multi-lingual support: The SentimentIntensityAnalyzer algorithm assumes that the text input is in English, and it may not work well for text in other languages. One possible future direction is to support multiple languages by training different models for each language or using a multi-lingual model.

3) Aspect-based sentiment analysis: Sentiment analysis can be enhanced by identifying the aspect or feature of a product or service that is being discussed in a review. This is known as aspect-based sentiment analysis. It is possible to use techniques like named entity recognition or topic modeling to identify the aspects and then classify the sentiment of each aspect.

4) Real-time sentiment analysis: Sentiment analysis can be integrated into real-time systems, such as social media monitoring or customer service chatbots, to provide real-time feedback and insights.

5) Combining with other NLP techniques: Sentiment analysis can be combined with other natural language processing (NLP) techniques, such as text summarization, named entity recognition, or question answering, to provide a more comprehensive understanding of the text.

6) Application to other domains: Sentiment analysis can be applied to a wide range of domains, such as healthcare, finance, and political analysis. The future possibilities can be explored by applying the model to different domains and fine-tuning it for the specific use cases.

7) Human in the loop: As sentiment analysis models are not perfect, one approach that can be taken is to use a "human in the loop" approach, where the model's predictions are reviewed and corrected by human annotators. This approach can improve the accuracy of the model and make it more robust.

These are just few other possibilities.

Conclusion:

In conclusion, sentiment analysis is a powerful tool that can be used to understand the opinions and emotions expressed in text data. The SentimentIntensityAnalyzer algorithm is a pre-trained model that can quickly and accurately classify text into positive, negative, and neutral categories. It uses a combination of lexical heuristics and a pre-trained model to calculate a compound sentiment score between -1 and 1. The algorithm is easy to use and understand, and it can be easily integrated into a wide range of applications and projects.

However, it's important to keep in mind that all ml models have assumptions and limitations, and it is important to be aware of them and test the model on a variety of inputs to understand its performance and limitations. Additionally, multiple evaluation metrics should be used to get a comprehensive understanding of the model's performance.

There are several future possibilities for a sentiment analysis project, depending on the specific use case and the goals of the project. These possibilities include fine-tuning the model, multi-lingual support, aspect-based sentiment analysis, real-time sentiment analysis, combining with other NLP techniques, application to other domains and human in the loop.

In summary, SentimentIntensityAnalyzer is a powerful tool for sentiment analysis, and it can be used to gain valuable insights from text data. However, it's important to be aware of its assumptions and limitations, and it's always a good idea to stay up-to-date with the latest advancements in the field and consider new techniques and technologies that may be relevant to the project.

References

- Intellipaat live session recording & notes
- Wikipedia
- Google.com
- https://www.packtpub.com/data/sentiment-analysis-with-python
- https://www.nltk.org/api/nltk.sentiment.html
- https://www.oreilly.com/library/view/natural-language-processing/9780596516499/