INTEL PRODUCTS SENTIMENT ANALYSIS FROM ONLINE REVIEWS

Intel Unnati Industrial Training Program 2024 - Project Report

Name: Nidhi G

USN: 1BI21 CS082

Department of Computer Science and Engineering

Bangalore Institute of Technology

K.R. Road, V.V.Puram, Bangalore-560 004

Acknowledgement

I would like to express my heartfelt gratitude to everyone who contributed to the successful completion of the project titled “INTEL PRODUCTS SENTIMENT ANALYSIS FROM ONLINE REVIEWS.” This endeavor would not have been possible without the unwavering support and guidance from individuals and institutions.

First and foremost, I extend my sincere thanks to Intel Unnati Industrial Training Program for providing me with this incredible opportunity. The exposure to real-world data and industry practices has been invaluable, and I am grateful for the chance to work on such a relevant and challenging project.

I am deeply indebted to my internal guide, Dr. Bhanushree K J, for her mentorship, encouragement, and insightful feedback throughout the project. Her expertise and dedication significantly enriched my learning experience.

I also extend my appreciation to the external guide, Mr. Debdyut Hazra, whose expertise in the field of sentiment analysis was instrumental in shaping the project. His mentoring and practical insights were invaluable.

Furthermore, I would like to thank Bangalore Institute of Technology for fostering an environment conducive to learning and innovation. The support from the Computer Science and Engineering Department, led by Dr. Girija J, has been commendable. Their encouragement and resources allowed me to explore new horizons and enhance my skills.

Lastly, I express my gratitude to Intel for their commitment to nurturing young talent and fostering innovation. This project has been a remarkable learning journey, and I look forward to applying the knowledge gained in my future endeavors.

Warm regards, Nidhi

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Abstract

In this study, we perform sentiment analysis on customer reviews of Intel desktop processors from generations 11 to 14. The data is collected by scraping Amazon reviews. After cleaning and preprocessing the data, we conduct exploratory data analysis (EDA) to gain insights into the review distribution and key features.

We then employ two pretrained models, VADER and RoBERTa, to classify the sentiment of each review. By extracting results and conclusions from these models, we analyze the overall sentiment trends across different processor generations. Our findings provide valuable insights for Intel's product development and marketing strategies.

1.2 Introduction

NLP is an exciting field of artificial intelligence that bridges the gap between human communication and computational algorithms. By analyzing text data, NLP algorithms extract meaning, identify patterns, and perform various language-related tasks. One crucial subfield within NLP is sentiment analysis, which systematically quantifies emotions expressed in textual data—whether it's customer reviews, social media comments, or news articles. Understanding sentiment provides valuable insights for businesses, helping them enhance products, services, and brand strategies.

Background

This project is concerned with sentiment analysis field of NLP. User reviews play a crucial role in assessing the quality, performance, and overall satisfaction with Intel processors.

They act as a valuable feedback mechanism for Intel. By collecting reviews directly from customers, Intel gains insights into their experiences, preferences, and pain points. This feedback helps identify areas for improvement and informs future product development.

Analyzing reviews from users with varying backgrounds—such as gamers, professionals, or casual users—provides a holistic view of how different segments perceive Intel processors. This diversity ensures that Intel considers a wide range of opinions and needs.

Users discuss aspects like performance, power efficiency, compatibility, and reliability. Sentiment analysis helps extract sentiments associated with these features, revealing which aspects resonate positively or negatively with users.

Armed with sentiment analysis, Intel can prioritize enhancements. For example, if users consistently praise a particular feature, Intel can invest in optimizing it further. Conversely, if negative sentiments arise around a specific aspect, Intel can address it in future iterations.

Objective of this project

The primary goal of this project is to analyze user reviews and classify their sentiments as positive, negative, or neutral. User reviews related to Intel processors from one or more sources. Apply machine learning techniques to automatically categorize each review into one

of three sentiment classes: positive (praising), negative (critical), or neutral (neutral observations). By classifying sentiments, we gain insights into how users perceive Intel processors.

Once reviews are classified, we can understand what users are saying about Intel processors. Positive sentiments highlight features that users appreciate, while negative sentiments point out areas for improvement. Neutral sentiments provide context and balance.

Finally, we can draw conclusions from it by analysing the patterns and trends in sentiment over time. These insights can be used to understand about product strengths, weaknesses, and potential enhancements.

Scope

The scope of this project involves collecting approximately 1,500 user reviews from Amazon for Intel desktop processors spanning generations 11 to 14. The primary objectives are to classify these reviews based on sentiment (positive, negative, or neutral) and analyze the results. By doing so, we aim to gain insights into users' perceptions of Intel processors.

Chapter 2: Literature Review

2.1 Overview of Sentiment Analysis

- Sentiment analysis, also known as opinion mining, has gained prominence within natural language processing (NLP) due to the exponential growth of digital data.The need to extract insights from textual information has become crucial across various domains, including business intelligence, social sciences, and disaster response.
- Sentiment analysis involves identifying, extracting, and analyzing subjective information from text to determine the overall sentiment expressed toward a specific entity, product, topic, or event.
- Text is classified into predefined categories such as positive, negative, or neutral sentiments, or more nuanced emotions and opinions.

Traditional Approaches and Challenges

Initially, sentiment analysis relied on machine learning algorithms like support vector machines (SVM), Naive Bayes, logistic regression, and random forests.

However, as the scope of sentiment analysis expanded, new gaps emerged within sentiment classifications. Researchers shifted toward more complex algorithms to address emerging challenges.

Deep learning algorithms, including architectures like convolutional neural networks (CNNs), long short-term memory networks (LSTMs), and recurrent neural networks (RNNs), demonstrated efficacy in handling complex sentiment analysis tasks.

Challenges and Ethical Considerations

- Sentiment analysis faces challenges such as deciphering sarcasm and irony, ensuring ethical use, and adapting to new domains.
- The dynamic nature of sentiment analysis necessitates further research to understand the nuances of human sentiment expression.
- Responsible and impactful applications across industries and languages are encouraged.

2.2 Pretrained Models for Sentiment Analysis

Pre-trained Language Models (PLMs) have revolutionized downstream tasks, including sentiment analysis. These models come with extensive pre-trained knowledge from large text corpora, enabling them to capture complex linguistic nuances, contextual comprehension, and semantic associations. By fine-tuning PLMs, we can apply them directly to sentiment analysis tasks without training from scratch.

Popular Pre-trained Sentiment Analysis Models:

BERT (Bidirectional Encoder Representations from Transformers):

BERT, based on transformer architecture, has achieved state-of-the-art performance in various NLP tasks, including sentiment analysis.Its bidirectional context understanding allows it to capture rich contextual information.

Google Natural Language:

Google's Natural Language API provides pre-trained models for sentiment analysis.It leverages deep learning techniques to analyze sentiment in text data.

IBM Watson Natural Language Understanding:

IBM's NLU service offers pre-trained models for sentiment analysis. It combines rule-based and machine learning approaches to extract sentiment information.

TextBlob Simplified Text Processing:

TextBlob, a Python library, provides simple sentiment analysis tools.While not as sophisticated as deep learning models, it's easy to use and suitable for basic tasks.

VADER Sentiment Analysis:

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon-based approach. It assigns sentiment scores to words and aggregates them to determine overall sentiment.

Chapter 3: Data Collection

Data collection is fundamental for informed decision-making, research, and quality assurance. Researchers rely on data to analyze trends and patterns, while predictive modeling benefits from accurate data. Additionally, organizations use collected data to evaluate performance and identify areas for improvement.

3.1 Data sources

Data can be collected from various sources , some include survey, customer reviews, reviews from technical experts , social media , news articles, online shopping sites, and many more.

In this project the customer reviews has been scraped from Amazon.com which provides a rich dataset. Amazon's global user base ensures a wide range of perspectives, making it a valuable resource. By analyzing these reviews, you can gain insights into Intel products' real-world performance and customer satisfaction. The unbiased nature of customer feedback allows you to assess products without external influence. Trustworthy platforms like Amazon contribute significantly to accurate sentiment analysis.

3.2 Data Acquisition

In this project, Playwright library of python is utilized to extract data from Amazon. Specifically, leveraging the "async_playwright" module, which integrates with "asyncio" for asynchronous web scraping. Playwright is a powerful tool for automating browser interactions, allowing to navigate web pages, interact with elements, and extract relevant information. By combining it with "asyncio", one can efficiently handle concurrent tasks, making it ideal for scraping large amounts of data from dynamic websites like Amazon. This collected data is stored in JSON files

3.3 Data Description

The features of the collected data are following:

- Product name -has the complete name of the product reviews extracted from amazon. Example- " Intel Core I9-14900K LGA 1700 New Gaming Desktop Processor 24 Cores (8 P-Cores + 16 E-Cores) with Integrated Graphics – Unlocked "
- Rate – The rating of the product given by users . This rating is out of 5. Example – " 5.0 out of 5 stars "
- Review Date – The date and location of the review. Example- " Reviewed in India on 20 November 2023"
- Review Text -The actual review by the customer. Example – " Fast shipping. Works as advertised. "
- Patter Name- Name of the pattern of the processor. Only present in those products where categories are present under the same product name. Example – "Pattern Name: ProcessorStyle Name: Core™ i9-14900k "
- Product MRP – The maximum retail price of the product . " ₹82,350"

	product_name	rate	review_date	review_text	product_style	product_mrp
0	Intel Core I9-14900K LGA 1700 New Gaming Deskt...	5.0 out of 5 stars	Reviewed in India on 20 November 2023	Just upgraded from i5 9400f to i9 14900k and s...	Pattern Name: ProcessorStyle Name: Core™ i9-14...	₹82,350
1	Intel Core I9-14900K LGA 1700 New Gaming Deskt...	1.0 out of 5 stars	Reviewed in India on 28 March 2024	I don't liked the price of it .\nI Also don't ...	Pattern Name: ProcessorStyle Name: Core™ i9-14...	₹82,350
2	Intel Core I9-14900K LGA 1700 New Gaming Deskt...	1.0 out of 5 stars	Reviewed in India on 7 December 2023	13 gen is better with same performence at low ...	Pattern Name: ProcessorStyle Name: Core™ i9-14...	₹82,350
3	Intel Core I9-14900K LGA 1700 New Gaming Deskt...	5.0 out of 5 stars	Reviewed in Spain on 16 May 2024	Buen procesador que te da para poder jugar y r...	Pattern Name: ProcessorStyle Name: Core™ i5-14...	₹82,350
4	Intel Core I9-14900K LGA 1700 New Gaming Deskt...	5.0 out of 5 stars	Reviewed in Japan on 23 February 2024	何をするにも早いですね！\n流石です！\n大満足です。	Pattern Name: ProcessorStyle Name: Core™ i9-14...	₹82,350

Figure 1 – Features and a peek into the dataset

Description of the dataset

	product_name	rate	review_date	review_text	product_style	product_mrp
count	1650	1650	1650	1650	1259	1650
unique	18	5	1311	1625	14	17
top	Intel Core I9-14900K LGA 1700 New Gaming Deskt...	5.0 out of 5 stars	Reviewed in the United States on 1 December 2023		Pattern Name: Processor	₹82,350
freq	100	1328	6	10	479	190

Figure 2 – Characteristics and description of the data

The dimension of this dataset is 1650 * 6. All the features are of object type.

Chapter 4 : Data Pre-processing

Data pre-processing is a critical step in preparing raw data for analysis. It involves cleaning, transforming, and organizing the data to improve its quality and suitability for further processing. Common tasks include handling missing values, removing duplicates, standardizing formats, and scaling features. Proper data pre-processing ensures accurate and reliable results in subsequent analyses.

4.1 Cleaning of data

Missing values

The data doesn't seem to have any missing values. But the missing values may be disguised as empty string. This need to be checked.

It also does not have any duplicate rows.

```
duplicate_rows = dataset[dataset.duplicated()]
duplicate_rows
```

product_name	rate	review_date	review_text	product_style	product_mrp

Figure 3 – Checking for duplicate rows

```
#Checking for any duplicated review text
print(dataset[dataset.duplicated('review_text')].shape)
dataset[dataset.duplicated('review_text')]
```

(25, 6)

	product_name	rate	review_date	review_text	product_style	product_mrp
13	Intel Core i5-11400F Desktop Processor 6, 6 Co...	1.0 out of 5 stars	Reviewed in India on 12 February 2023		Pattern Name: Processor	₹24,000
29	Intel Core i9-11900K Desktop Processor 1, 8 Co...	5.0 out of 5 stars	Reviewed in Canada on 9 November 2023	Very good	Style Name: CPU Only	₹80,000
49	Intel Core i3 12100F 12th Gen Generation Deskt...	5.0 out of 5 stars	Reviewed in Brazil on 18 November 2023	Top	NaN	₹14,500
6	Intel Core i5-12400 Desktop Processor 18M Cach...	5.0 out of 5 stars	Reviewed in India on 25 October 2023	Super	Style Name: Cache	₹28,500
10	Intel Core I5 12400F 12 Gen Generation Desktop...	4.0 out of 5 stars	Reviewed in India on 18 November 2023		Pattern Name: Processor	₹29,999
19	Intel Core I5 12400F 12 Gen Generation Desktop...	5.0 out of 5 stars	Reviewed in India on 5 July 2023	Value for money	Pattern Name: Processor	₹29,999
32	Intel Core I5 12400F 12 Gen Generation Desktop...	5.0 out of 5 stars	Reviewed in India on 22 July 2022	Got this product in good package 📦.\nI have pa...	None	₹29,999
45	Intel Core i5-12600K Desktop Processor 10 (6P+...	5.0 out of 5 stars	Reviewed in the United States on 25 May 2024	Excelente	NaN	₹41,100

Figure 4 – Duplicate Reviews

The above command is used to check if same review is recorded twice while collecting it. It doesn't seem like this has happened. Though there are some empty spaces and some common review text like "Very Good" which are seen as duplicate. So this doesn't account for any actual duplicated review data.

The reviews with only rating and no text review has to be removed since doesn't meet our criteria for sentiment analysis.

```
#droping the rows which has no review text in it
dataset.drop(dataset.loc[dataset['review_text'] == ''].index, inplace=True)
```

```
print(dataset.loc[dataset['review_text'] == ''])
print(dataset.loc[dataset['review_text'] == ''].shape)
```

```
Empty DataFrame
Columns: [product_name, rate, review_date, review_text, product_style, product_mrp]
Index: []
(0, 6)
```

Figure 5 - Dropping empty reviews

The rating of products is object type, it can be converted to integer type by having a value from 1 to 5 depending on the rating.

```
#converting rating to integer
dataset['rating'] = dataset['rate'].apply(lambda x: int(re.findall(r'\d+', x)[0]))
dataset['rating']
```

```
0     5
1     1
3     5
4     5
5     5
     ..
55    1
56    1
57    5
58    5
59    1
Name: rating, Length: 1481, dtype: int64
```

Figure 6 – Cleaning the rating column

Next, the prices of each product is checked to ensure that its right, some prices are wrong , so it is corrected. Later the MRP is changed to integer type.

```
df['product_mrp'].unique()
```

```
array(['₹79,999', '₹45,000', '₹73,700', '₹53,000', '₹24,000', '₹58,500',
       '₹80,000', '₹14,500', '₹28,500', '₹29,999', '₹41,100', '₹50,200',
       '₹60,000', '₹99,999', '₹20,250', '₹27,500', '₹31,700', '₹46,500',
       '₹59,999', '₹82,350', '₹59,100'], dtype=object)
```

```
#converting mrp to int
def rupee_to_int(rupee_str):
    # Remove ₹ symbol and commas
    numeric_value = re.sub(r"₹|,", "", rupee_str)
    # Convert to integer
    return int(numeric_value)
dataset['mrp'] = df['product_mrp'].apply(rupee_to_int)

dataset['mrp']
```

```
0       79999
1       79999
2       45000
3       73700
4       79999
        ...
1476    59100
1477    59100
1478    59100
1479    59100
1480    59100
Name: mrp, Length: 1481, dtype: int64
```

Figure 7 - MRP of products

The review data column is made into date type and a separate column consisting of the country names from where the review was written is made. Following is how the data looks after all the cleaning.

```
df.head()
```

	review_date	review_text	rating	mrp	country	product_id
0	2023-11-20	Just upgraded from i5 9400f to i9 14900k and s...	5	79999	India	14900K
1	2024-03-28	I don't liked the price of it .\nI Also don't ...	1	79999	India	14900K
2	2024-05-16	Buen procesador que te da para poder jugar y r...	5	45000	Spain	14600KF
3	2024-02-23	何をするにも早いですね！\n流石です！\n大満足です。	5	73700	Japan	14900KF
4	2024-03-05	bonne decision pour mon ordi	5	79999	Canada	14900K

Figure 8 – Cleaned data

4.2 Text pre-processing

To apply various models for training, text preprocessing one crucial step. It involves cleaning the text, tokenization, removing stop words, lemmatization and various other techniques are applied to do the same.

In this project, text preprocessing is done so that meaningful data can be extracted.

Firstly , the review text is cleaned by removing parts that are unnecessary.

```
#cleaning the text by removing '\n'
df['review_text'] = df['review_text'].str.replace('\n', ' ')
df['review_text'][0]
```

```
'Just upgraded from i5 9400f to i9 14900k and super happy with the performance that I am able to achieve. It simply works buttery smooth. Large applicati
ons open like notepad apps.  It heats a lot though so I went with MSI coreliquid liquid cooling solution.  If looking to upgrade from 13th gen then save
money because the performance is identical, however if setting up new PC then definitely go with it.  My setup - MSI MPG z790 carbon wifi Intel i9 14900k
MSI RTX 4070ti MSI MAG coreliquid liquid cooler Samsung 980 Pro - 1 TB Ripjaws 32gb ddr5  Bonus tip: If u are struck setting up ur PC then go to MSI supp
ort and they are super responsive to assist you. I got my wiring fixed in matter of minutes ona video call by their assistant.'
```

Figure 9- Removing '/n' from review

Many reviews are present in other languages, this has to be translated. This is done using Google's Translator API.

```
translator = Translator(service_urls=['translate.google.com'])
translate_1=pd.Series()
failed=[]
for i in range(1481):
    try:
        a=translator.translate(df.iloc[i,2],dest='en').text
        translate_1 = pd.concat([translate_1,pd.Series([a])])
    except:
        print(i)
        failed=int(df.iloc[i,0])
        translate_1 = pd.concat([translate_1,pd.Series('')])
    if i%10 ==0:
        time.sleep(3)
translate_1
```

Figure 10 – Translating the text to English if it is not in English

Stop word removal is a preprocessing step in natural language processing (NLP) where common words (such as "the," "and," "is") are filtered out from text data. These words don't carry significant meaning and can be safely removed to improve the efficiency and accuracy of NLP tasks like sentiment analysis.

```
#removing stop words
def remove_stopwords (text):
    new_text = []
    for word in text.split():
        if word in stopwords.words('english'): new_text.append('')
        else:
            new_text.append(word)
    x = new_text[:]
    new_text.clear()
    return " ".join(x)

a=df['trans_review'].apply(remove_stopwords)
```

```
a[6]
```

```
'good surprise  power  processor  monster  power crucial point  heats  lot  put  aio  420mm  correctly cool  processor second point  consumes  lot
always  possibility  paying  maximum  125 watts  bios  deal  loss  4 8 perf honestly  gamer  need raw power turn  i5  make  trick'
```

Figure 11- Stop words are removed from review text

The text has some emojis in it. Since it is not a lot, it can be removed .

```
#handling emojis , romving them
import re
def remove_emoji(text):
    emoji_pattern = re.compile("["
    u"\U0001F600-\U0001F64F" # emoticons
    u"\U0001F300-\U0001F5FF" # symbols & pictographs
    u"\U0001F680-\U0001F6FF" # transport & map symbols
    u"\U0001F1E0-\U0001F1FF" # flags (iOS)
    u"\U00002702-\U000027B0"
    u"\U000024C2-\U0001F251"
    "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
a=a.apply(remove_emoji)
```

Figure 12 – Handling Emojis

The words that have numbers are not significant, hence they are removed.

```
#removing the words which have numbers in them
def remove_words_with_numbers(input_string):
    # Split the input string into words
    words = input_string.split()

    # Filter out words containing numbers
    filtered_words = [word for word in words if not re.search(r'\d', word)]

    # Join the filtered words back into a string
    result = ' '.join(filtered_words)
    return result
a=a.apply(remove_words_with_numbers)
a
```

```
0       upgraded super happy performance able achieve ...
1       liked price also liked performance rather purc...
```

Figure 13 – Removing words containing Numbers

Next crucial step is to tokenize and then lemmatize the text.

```
#tokenizing the text then applying lemmatization
from nltk.stem import WordNetLemmatizer
wordnetlemmatizer =WordNetLemmatizer()
def tokenize_lemmatize(text):
    tokenized_list = text.split(" ")
    lemmatized_list = [wordnetlemmatizer.lemmatize(word) for word in tokenized_list]
    return lemmatized_list
b=a.apply(tokenize_lemmatize)
```

Figure 14 – Tokenizing and Lemmatizing

Thus, the data is prepared and can be used to perform sentiment classification

Chapter 5: Exploratory Data Analysis

Exploratory Data Analysis (EDA) plays a pivotal role in data science. It involves visualizing and summarizing data to uncover patterns, outliers, and relationships, helping analysts gain insights before diving into more complex modelling.

Number of reviews for each product

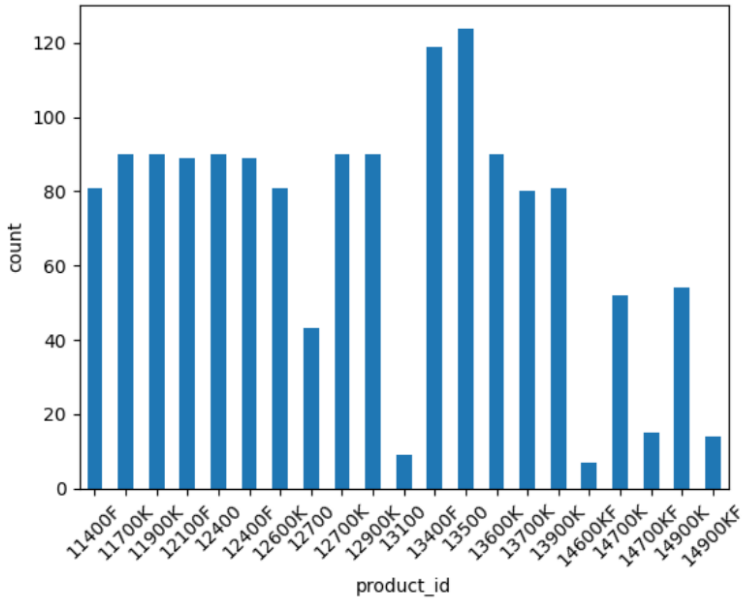

Figure 15 – Number of reviews for each product

As we can see all the products don't have equal number of reviews. Data for some products is very less.

MRP of different products

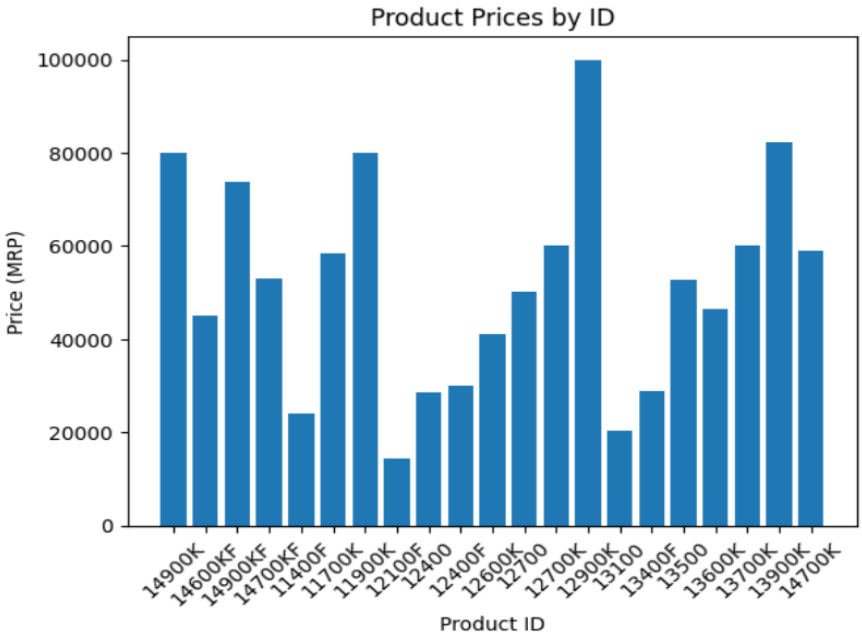

Figure 16 – MRP vs Product

Average rating vs Price

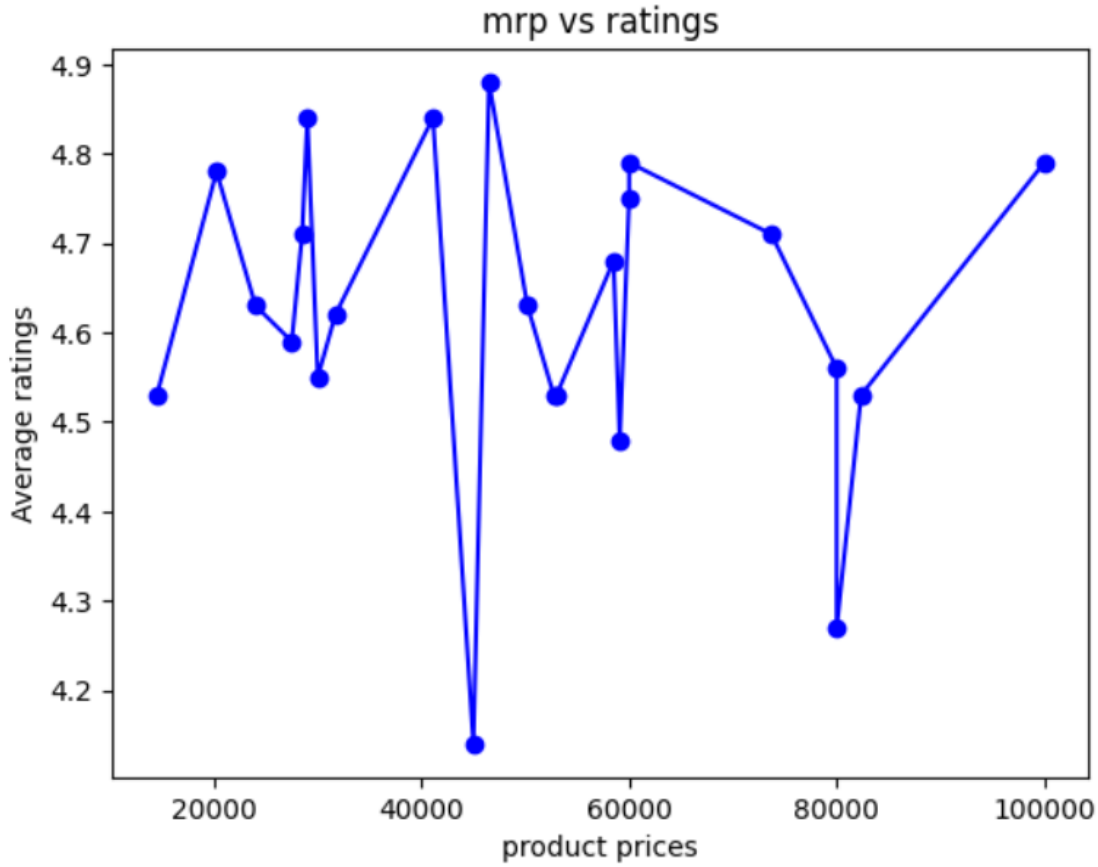

Figure 17 – MRP vs Average rating

We can see that product prices doesn't seem to influence the ratings.

Chapter 6 : Sentiment Analysis Methodology

6.1 Model and Approach

For sentiment analysis in this project, VADER and RoBERTa models have been used.

The VADER (Valence Aware Dictionary and sentiment Reasoner) model is a rule-based approach for sentiment analysis. It relies on a pre-defined lexicon of words and their associated sentiment scores to determine the overall sentiment of a text.

On the other hand, the RoBERTa model, which is based on the Transformer architecture, falls under the category of deep learning approaches. It fine-tunes a large-scale language model on a specific sentiment analysis task, learning contextual representations from vast amounts of text data.

VADER

VADER relies on a pre-built sentiment lexicon, which contains words with associated sentiment scores.

Each word in the lexicon is assigned a polarity score (positive, negative, or neutral) and an intensity score. The lexicon includes common words, phrases, and emoticons, along with their sentiment values. By summing up the scores of individual words in a text, VADER determines the overall sentiment.

VADER incorporates specific rules (heuristics) to account for context and emphasize sentiment intensity. These rules go beyond what a typical bag-of-words model captures.

Some examples of these rules:

- Capitalization: Uppercase words may indicate strong sentiment.
- Punctuation: Exclamation marks increase intensity.
- Degree modifiers: Intensifiers (e.g., “very,” “extremely”) affect sentiment.
- Conjunctions: “But” can reverse sentiment.
- Negation: Words like “not” change polarity.

RoBERTa

RoBERTa (A Robustly Optimized BERT Pretraining Approach) is a powerful language model that builds upon the foundation of BERT (Bidirectional Encoder Representations from Transformers).

RoBERTa is based on the Transformer architecture, which excels at capturing contextual information from text.

Unlike BERT, RoBERTa pretrains on full sentences without the Next Sentence Prediction (NSP) loss. This change allows it to learn better sentence representations. It uses dynamic masking during pretraining, enhancing its ability to understand context.

RoBERTa employs large mini batches during training, which improves efficiency and generalization. The tokenization process involves a larger byte-level BPE (Byte-Pair Encoding), leading to richer subword representations.

6.2 Model Selection

K Means Clustering

K-means clustering is an iterative process that partitions a dataset into similar groups based on the distance between their centroids. It assigns each observation to the nearest centroid and updates the centroids until convergence. Widely used in customer segmentation, image compression, and anomaly detection, K-means minimizes the sum of squared distances within clusters.

This model is chosen, because it is an unsupervised learning approach and this can help to classify the data. The dataset is not labelled as positive, negative or neutral, hence unsupervised learning approach is the best way.

On applying K Means clustering it is found that the classification is done. But this classification doesn't show the separation between positive, negative and neutral rather it just separates longer and shorter reviews. Hence, a requirement for pretrained model arises. This is when models like VADER and RoBERTa are useful.

Choosing RoBERTa and VADER

These models, will classify the data and help in gaining the required insights from the data.

Benefits of using VADER:

- Rule-Based Simplicity: Lexicon-based approach with context-aware rules.
- Social Media Focus: Ideal for short, social media-style content.
- Emoticon Handling: Considers emoticons and punctuation.
- Efficient Inference: Real-time analysis with low computational overhead.

Benefits of using RoBERTa:

- Deep Learning Powerhouse: Captures rich context using Transformers.
- Fine-Tuning Flexibility: Adaptable to specific tasks like sentiment analysis.
- Large-Scale Pretraining: Learns from vast text data for better generalization.
- Domain Generalization: Performs well across diverse domains and languages.

Chapter 7: Implementation

7.1 Tools and Libraries

Python

Python is a versatile, high-level programming language widely used for various applications.

Using Python is advantageous due to its extensive library support and simplicity, making it ideal for sentiment analysis and natural language processing with libraries like NLTK and spaCy, which facilitate efficient text analysis and manipulation. For web scraping, Python offers powerful tools like BeautifulSoup and Playwright, allowing easy extraction and automation of web data. Its clear syntax and active community enhance rapid development and problem-solving. Additionally, Python's adaptability across different domains and platforms makes it a go-to language for developers and data scientists alike.

playwright.async_api

Playwright is a powerful library for browser automation and testing. It allows you to control browsers (such as Chromium, Firefox, and WebKit) programmatically.

Asyncio

Asyncio is a Python library used for concurrent programming, specifically asynchronous I/O. It allows you to write code that doesn’t block execution while waiting for I/O operations (such as reading from files, making network requests, or interacting with databases).

Pandas

The pandas library in Python is a powerful tool for data manipulation and analysis, providing data structures like DataFrame and Series for handling structured data efficiently. It simplifies data cleaning, transformation, and aggregation tasks, making it indispensable for data science and machine learning workflows.

Numpy

NumPy is a fundamental library for numerical computing in Python, providing support for arrays, matrices, and a vast collection of mathematical functions. Its efficient handling of large datasets and powerful operations on n-dimensional arrays make it essential for scientific computing and data analysis.

Googletrans

The googletrans library in Python provides a simple interface for translating text using Google Translate's API. It supports numerous languages and can detect the source language automatically, making it a convenient tool for multilingual text processing and analysis.

matplotlib.pyplot

The `matplotlib.pyplot` module in Python is a plotting library that provides a MATLAB-like interface for creating static, interactive, and animated visualizations. It's widely used for

generating a variety of charts, including line plots, scatter plots, and histograms, making it essential for data visualization in scientific and analytical applications.

Seaborn

The `seaborn` library in Python is built on top of `matplotlib` and provides a high-level interface for creating attractive and informative statistical graphics. It simplifies complex visualizations such as heatmaps, violin plots, and pair plots, making it ideal for exploratory data analysis and presentation.

nltk.sentiment

The `nltk.sentiment` module in Python is part of the Natural Language Toolkit (NLTK) and provides tools for sentiment analysis. It includes pre-trained models like VADER (Valence Aware Dictionary and sEntiment Reasoner) for accurately assessing the sentiment polarity of text. This module simplifies the process of determining whether a piece of text expresses positive, negative, or neutral sentiments. It is widely used in applications like social media analysis, customer feedback evaluation, and opinion mining.

AutoTokenizer

The `transformers` library by Hugging Face includes the `AutoTokenizer` class, which provides an easy way to load pre-trained tokenizers for various transformer models. It automates the process of tokenizing text for models like BERT, GPT-3, and T5, facilitating seamless integration into natural language processing tasks.

AutoModelForSequenceClassification

The `AutoModelForSequenceClassification` class in the `transformers` library by Hugging Face allows for easy loading of pre-trained models specifically tailored for sequence classification tasks. This class simplifies implementing tasks such as sentiment analysis, text categorization, and binary or multi-class classification by leveraging powerful transformer models like BERT, RoBERTa, and DistilBERT.

Softmax

The `softmax` function from `scipy.special` is used to compute the softmax of an array, transforming it into a probability distribution. It is commonly applied in machine learning for multi-class classification tasks, ensuring that the output values are non-negative and sum to one.

nltk.corpus.stopwords

The `stopwords` module from `nltk.corpus` provides a list of common words in various languages that are typically filtered out in natural language processing tasks. By removing stopwords, such as "the," "is," and "in," it helps improve the efficiency and relevance of text analysis and model training.

Regular expression

The `re` module in Python provides support for regular expressions, allowing users to search, match, and manipulate strings based on specific patterns. It is widely used for tasks such as text parsing, validation, and complex string replacements in data processing and analysis.

7.6 Sentiment classification

Using VADER model

To use VADER, nltk.sentiments library provides a sentiment analyser. The sentiment intensity analyser object needs to be created first.

The model can then be applied on any sentence to test its polarity.

```
#VADER approach
sia = SentimentIntensityAnalyzer()
```

```
#testing
sia.polarity_scores("I don't liked the price of it . I Also don't liked the performance of it . Rather you can purchase i9 13th generation")
```

```
{'neg': 0.206, 'neu': 0.794, 'pos': 0.0, 'compound': -0.5667}
```

Figure 18 – Applying VADER sentiment analyser

Here, the negative score shows how negative the input sentences is , its value ranging from 0 to1. Similarly neutral and positive scores show the neutrality and positivity of the sentence respectively ranging from 0 to 1. 1 being more of the characteristic and 0 being does not show that characteristic.

Compound score the overall sentiment score for the data ranging from -1 to 1. If the score is between -1 and -0.1 , it shows negative sentiment, if it is above 0.1 , it depicts positive sentiment and every other score as neutral.

This is applied to the entire dataset.

```
[6]: # Run the polarity score on the entire dataset
     scores = {}
     for i in range(len(df)):
         text = df['trans_review'].iloc[i]
         scores[i] = sia.polarity_scores(text)
```

```
[8]: vaders = pd.DataFrame(scores).T
     vaders
```

[8]:

	neg	neu	pos	compound
0	0.013	0.775	0.212	0.9818
1	0.206	0.794	0.000	-0.5667
2	0.000	0.739	0.261	0.6486
3	0.000	0.685	0.315	0.5686
4	0.000	0.580	0.420	0.4404
...	...	...	...	...
1473	0.080	0.824	0.096	0.4972

Figure 19 – Applying VADER model to entire dataset

This returns a dataframe having negative, neutral, positive, and compound score for each review.

Checking accuracy

The accuracy of the model can be assessed by comparing the ratings given by each person with the model scores.

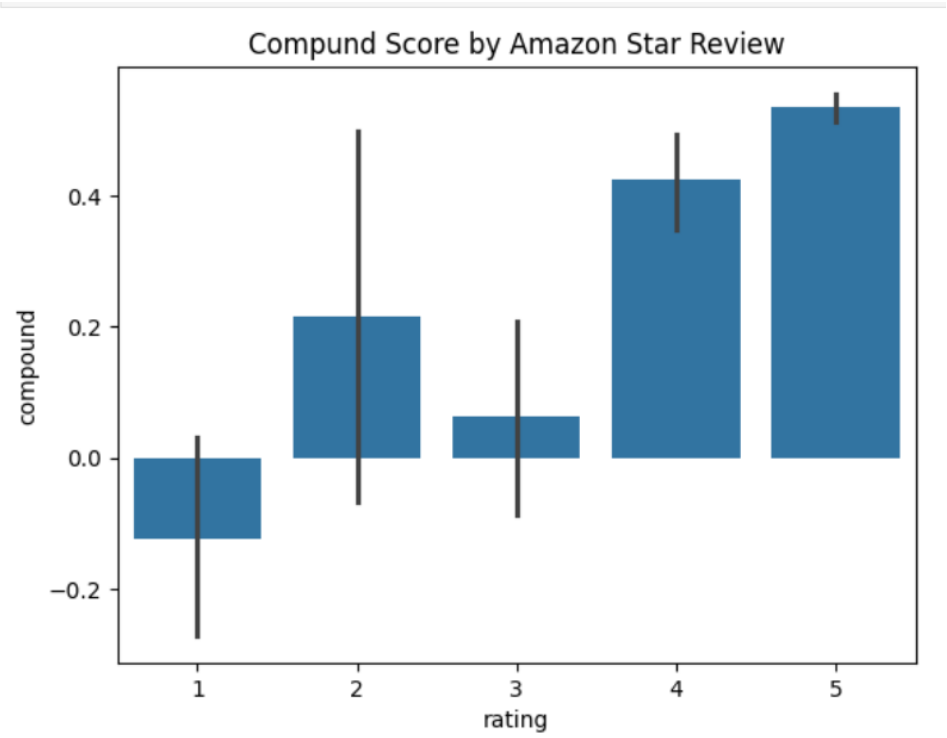

Figure 20- Compound score vs rating

As we can see that , as the rating increases , the score does too except for 3 star rating.

Comparing with other scores

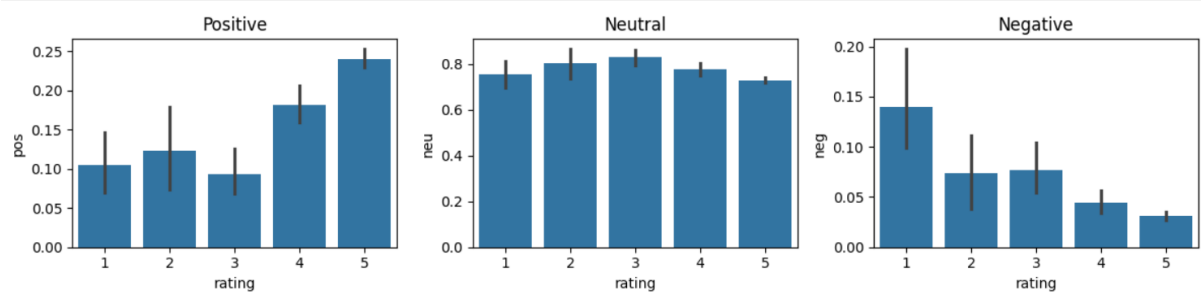

Figure 21 – rating Vs other scores

These scores also makes sense. The positive scores increase with increase in rating , the negative scores decrease. While the neutral scores is high for rating 3.

Based on the compound score ,the sentiment is decided for the review

- Positive- compound score > 0.1
- Negative – compound score< -0.1
- Neutral – -0.1 < compound score <0.1

	Unnamed: 0	review_date	rating	mrp	country	product_id	trans_review	sentiment
0	0	2023-11-20	5	79999	India	14900K	Just upgraded from i5 9400f to i9 14900k and s...	positive
1	1	2024-03-28	1	79999	India	14900K	I don't liked the price of it . I Also don't l...	negative
2	2	2024-05-16	5	45000	Spain	14600KF	Good processor that gives you to be able to pl...	positive
3	3	2024-02-23	5	73700	Japan	14900KF	It's early to do anything!As expected!I am ver...	positive
4	4	2024-03-05	5	79999	Canada	14900K	Good decision for my computer	positive

Figure 22 – Sentiments classified

Using RoBERTa model

The same process can be repeated with RoBERTa model. This will help in comparing both the models.

The text is initially tokenized using AutoTokenizer library. On this tokenized text , the model is applied. This gives the scores as output. On the scores , softmax is applied. Finally the scores contain positive , negative and neutral.

```
encoded_text = tokenizer("I don't liked the price of it . I Also don't liked the perfc
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
}
print(scores_dict)
```

```
{'roberta_neg': 0.9449626, 'roberta_neu': 0.050749753, 'roberta_pos': 0.004287674}
```

Figure 23 – Apply RoBERTa model on a Sentence

Next, the model is applied on the whole dataset to find the sentiment of the review.

```
robert= pd.DataFrame(roberta_scores).T
robert.head(10)
```

	roberta_neg	roberta_neu	roberta_pos
0	0.005560	0.080855	0.913585
1	0.944963	0.050750	0.004288
2	0.008118	0.185093	0.806788
3	0.003771	0.027059	0.969169
4	0.006656	0.081752	0.911592
5	0.010468	0.134045	0.855487
6	0.211626	0.442205	0.346169
7	0.007718	0.198480	0.793802
8	0.050000	0.050000	0.900000

Figure 24 – RoBERTa applied for the whole dataset

The scores are compared with the rating values to see their accuracy. Following results are obtained.

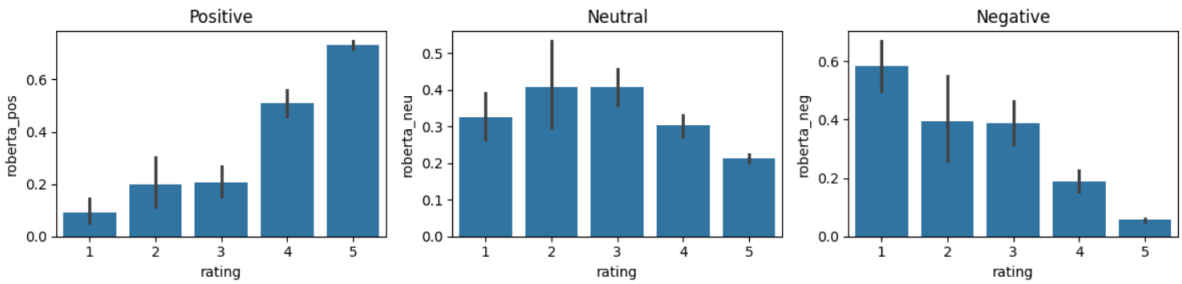

Figure 25 – RoBERTa scores vs rating

The sentiment is estimated based on these scores. Whichever score out of positive, negative and neutral is highest, the review is classified to that class.

0	review_date	rating	mrp	country	product_id	trans_review	sentiment
0	2023-11-20	5	79999	India	14900K	Just upgraded from i5 9400f to i9 14900k and s...	positive
1	2024-03-28	1	79999	India	14900K	I don't liked the price of it . I Also don't l...	negative
2	2024-05-16	5	45000	Spain	14600KF	Good processor that gives you to be able to pl...	positive
3	2024-02-23	5	73700	Japan	14900KF	It's early to do anything!As expected!I am ver...	positive
4	2024-03-05	5	79999	Canada	14900K	Good decision for my computer	positive

Figure 26 – RoBERTa classification

RoBERTa and VADER classifications are compared and it seems like both have done almost same classification.

Chapter 8: Conclusion

There are a few analysis that can be drawn from the data after classification.

Comparing the reviews of people from different countries

Looking at the reviews of people from different countries, we can see the acceptance and how they find INTEL products.

Here is the comparison:

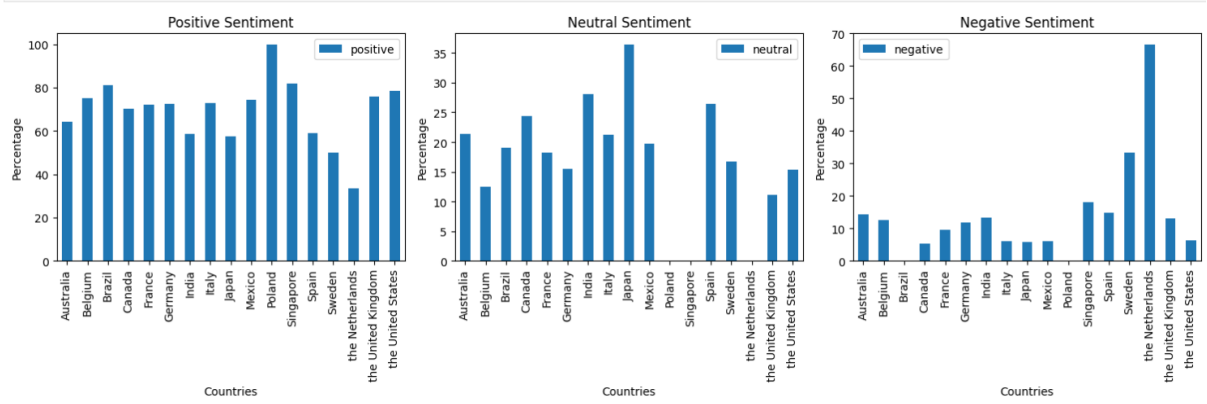

Figure 27 – Comparing the reviews of people from different countries

We can see that people from Poland have given the most positive reviews and Netherlands has the most negative reviews.

This does not imply much because of limitation of the available data. This process can be used to withdraw meaningful insights when the data is of large scale.

Checking the reviews for each product

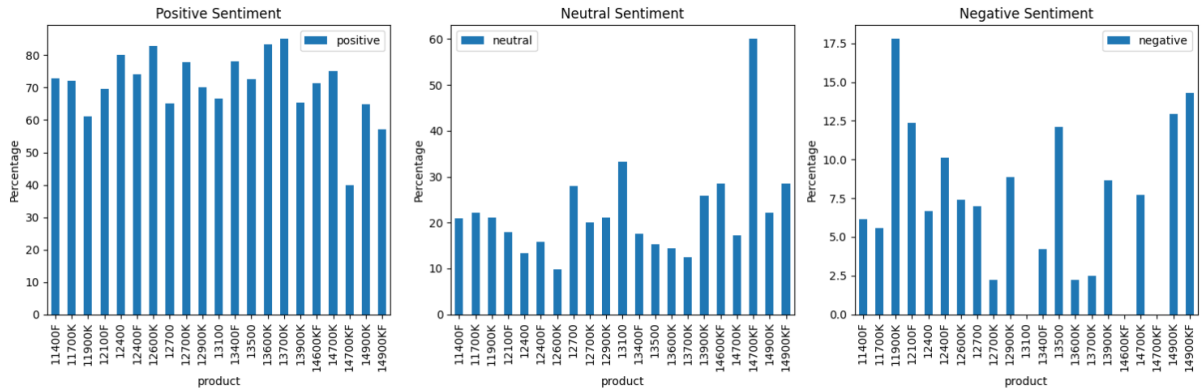

Figure 28 – Product vs their review sentiment

Most of the products have similar trend in positive reviews. The negative reviews are highest for the model 11900K.

Sentiment of the product and price of the product

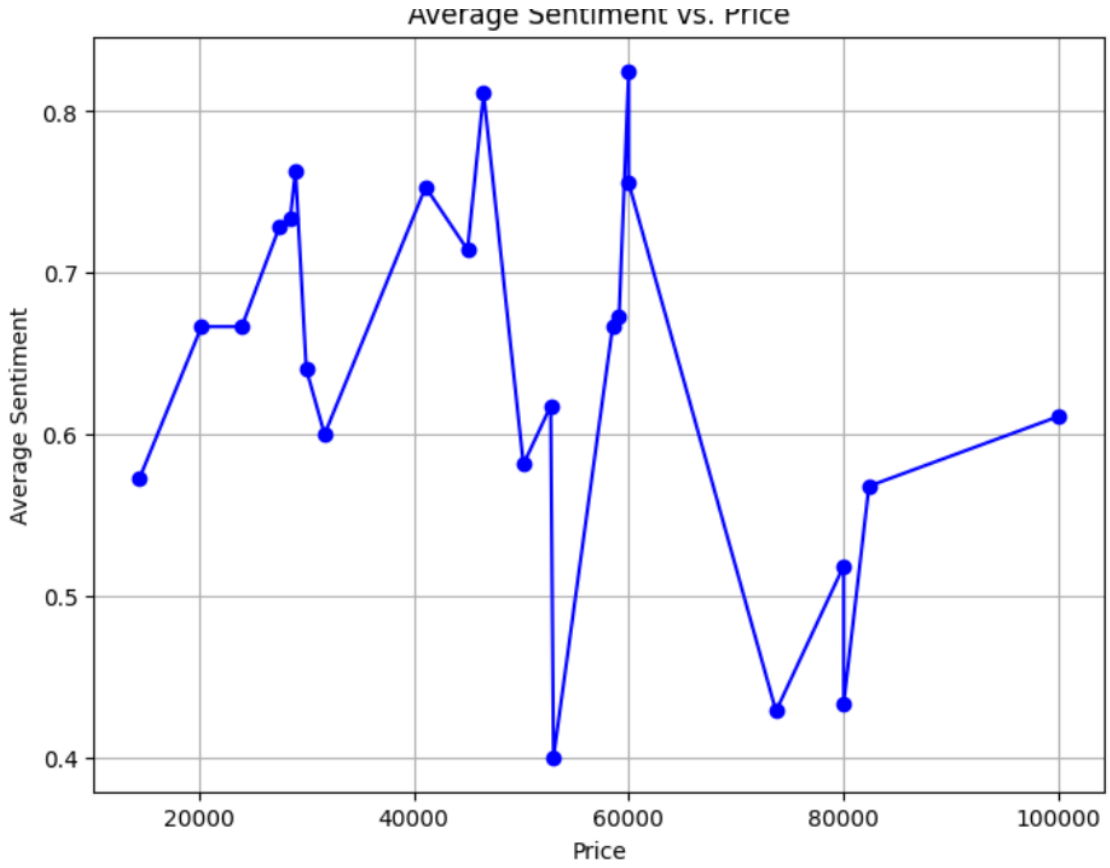

Figure 29 – Average sentiment vs price

As we can see, there is no trend seen here. So we can conclude that the price doesn't influence the sentiment of a review. The average sentiment of the product priced 60,000 is more than the others.

References

Natural language processing - Wikipedia
A review of sentiment analysis: tasks, applications, and deep learning techniques | International Journal of Data Science and Analytics (springer.com)
Getting Started with Sentiment Analysis using Python (huggingface.co)
Sentiment Analysis: Exploring Pre-trained and Domain-dependent Models | by Alex Ianovski | Analytics Vidhya | Medium