APPLICATION OF SENTIMENT ANALYSIS IN E-COMMERCE RECOMMENDATION SYSTEM

Prepared by: Ong Wei Aun (TP063332)
Supervisor: Mr. Raheem Mafas
2nd Marker: Prof. Dr. R. Logeswaran

TABLE OF CONTENTS

INTRODUCTION

E-COMMERCE PRODUCT REVIEWS

Feedbacks from customers

SENTIMENT ANALYSIS

Collect opinions and analyze sentiment using NLP

RECOMMENDATION SYSTEM

Provide user-personalized experience

PROBLEM STATEMENT

Spam and irrelevant reviews can mislead buyers in making purchase decisions

- Amazon (2016)
- Shopee (2021)

Adding review sentiment as part of recommendation mechanisms

- Quantify sentiment polarity through scoring

Combine explicit (reviews) and implicit feedback (clicks/views/best-selling) to recommend the best products for customers

METHODOLOGY

DATA COLLECTION

Amazon Dataset
- 15 variables

DATA PRE-PROCESSING

- Data Exploration (EDA)
- Data Cleaning

SENTIMENT ANALYSIS

- Sentiment Scoring
- VADER

RECOMMENDATION

- Combine sentiment scores and star ratings as recommendation scores
- Collaborative filtering

SEARCH ENGINE WITH RECOMMENDATION SYSTEM

- Match search term with product titles and create relevant recommendations

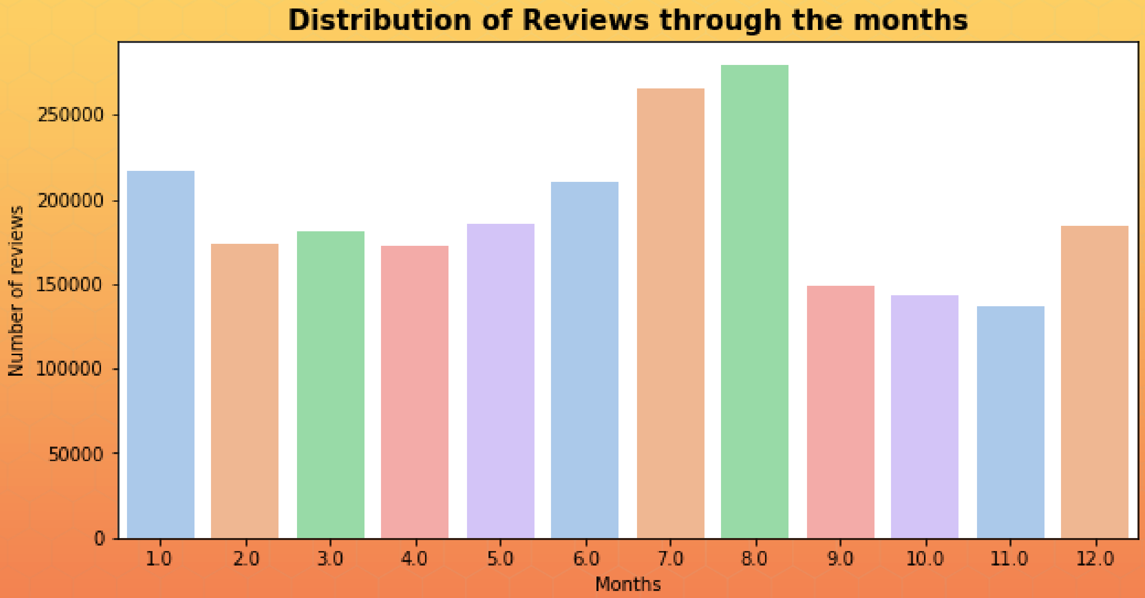
DATASET

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2299811 entries, 0 to 2299810
Data columns (total 15 columns):
 #   Column             Dtype
---  ------             -----
 0   marketplace        object
 1   customer_id        int64
 2   review_id          object
 3   product_id         object
 4   product_parent     int64
 5   product_title      object
 6   product_category   object
 7   star_rating        object
 8   helpful_votes      float64
 9   total_votes        float64
 10  vine               object
 11  verified_purchase  object
 12  review_headline    object
 13  review_body        object
 14  review_date        object
dtypes: float64(2), int64(2), object(11)
memory usage: 263.2+ MB
```

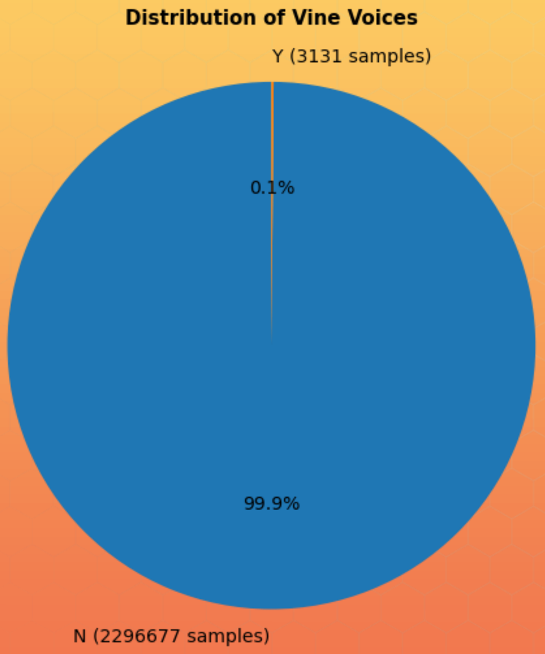
- Retrieved from Amazon AWS dataset repository (Niemi, 2015)
- Outdoors Category
- 15 variables
- Close to 2.3 million rows/observations

EDA – SALES SEASONALITY



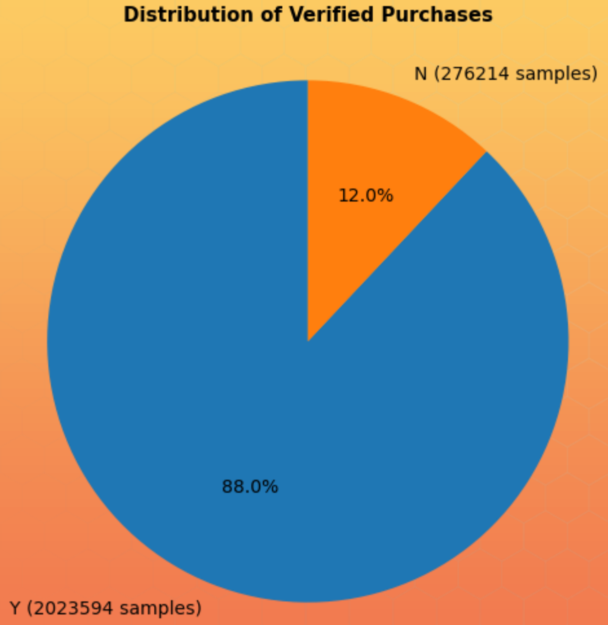
- Sales seasonality follows the weather season; summer is arriving in July and August
- Year end are shopping peak seasons with Christmas/Black Friday etc.

EDA – AMAZON VINE



Amazon Vine is a program that invites reviewers to post their reviews on the Amazon platform. In return, the Vine members, also known as Vine Voices, get free products from participating vendors. Since this is an invitation-only program, only the most trusted reviews on Amazon are invited, which explains the skewness of the distribution in the dataset.

EDA – VERIFIED PURCHASE

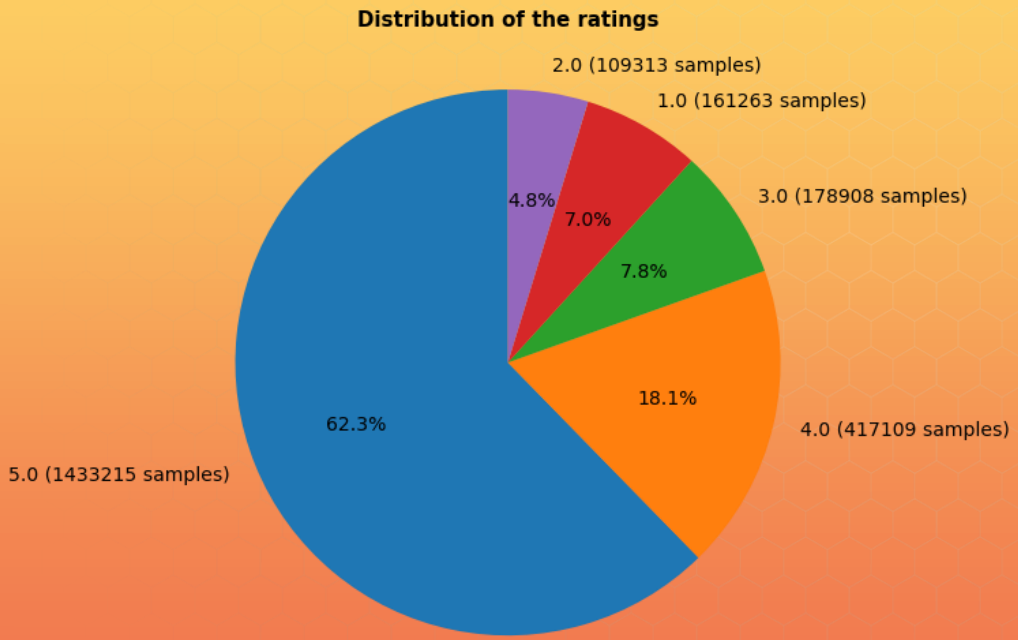


88.0% of the reviews have been verified as actual purchases.

```
#Drop verified purchase, filter out N values, left with Y values
df2 = df2[df2['verified_purchase'] != 'N']
```

To reduce the possibility of having fake reviews, only verified purchases are included for the further development in the study.

EDA – DISTRIBUTION OF STAR RATINGS



Rating 5 - 62.3% of the dataset
Rating 4 - 18.1%

Combined 80.4% indicating the overall satisfaction towards the products by the customers. These rating scores will be integrated with sentiment scoring to form recommendation scores

MISSING VALUE TREATMENT

Dropping observations with missing values

- With the large dataset of around 2.3 million observations, dropping the observations with missing values is simple and does not create biasness effect when the number of dropped observations is relatively small compared to the total.

```
df2.isnull().sum()
```

```
customer_id            0
review_id              0
product_id             0
product_title          0
star_rating            3
vine                   3
verified_purchase      3
review_body          135
review_date           13
dtype: int64
```

```
#Drop Null values in the columns in Pandas
df2=df2.dropna()
```

```
df2.isnull().sum()
```

```
customer_id          0
review_id            0
product_id           0
product_title        0
star_rating          0
vine                 0
verified_purchase    0
review_body          0
review_date          0
dtype: int64
```

DROPPING VARIABLES

Dropping Variables

```
# Dropping variables
df2 = df.drop(['marketplace','product_parent','product_category','helpful_votes','total_votes','review_headline'], axis = 1)
```

- The variables are less helpful for the study and takes up memory consumption in the system, so they are dropped to improved runtime

TOKENIZATION – WORD CLOUD

- ‘Great Product’
- ‘works great’
- ‘water bottle’

SENTIMENT SCORING

VADER

Using VADER SentimentIntensityAnalyser to calculate Sentiment Score

```
[ ]  #Using the NLTK library for importing the SentimentIntensityAnalyzer.

     import pandas as pd
     import nltk
     from nltk.sentiment.vader import SentimentIntensityAnalyzer
     nltk.download('vader_lexicon')
```

- Valence Aware Dictionary for Sentiment Reasoning (VADER) from NLTK
- Analyse sentiment through scoring, 4 classes (positive, negative, neutral & compound)
- Compound score, scale of -1 to 1, where -1 indicates the most negative sentiment and 1 is the most positive with 0 as neutral sentiment

CORRELATION BETWEEN SENTIMENT SCORE & STAR RATING

```python
# calculate Pearson's correlation
corr, _ = pearsonr(new_df['avg_rating'], new_df['avg_sentiment_score'])
print('Pearsons correlation: %.3f' % corr)

# calculate ttest significance
stats.ttest_ind(new_df['avg_rating'], new_df['avg_sentiment_score'])
```

```
Pearsons correlation: 0.667
Ttest_indResult(statistic=869.6214804286179, pvalue=0.0)
```

Pearson Correlation: 0.667
- High correlation

Scale of correlation coefficient	Value
$0 < r \leq 0.19$	Very Low Correlation
$0.2 \leq r \leq 0.39$	Low Correlation
$0.4 \leq r \leq 0.59$	Moderate Correlation
$0.6 \leq r \leq 0.79$	High Correlation
$0.8 \leq r \leq 1.0$	Very High Correlation

RECOMMENDATION SCORING

Rescaling the sentiment score from VADER

$$Y = \left(\frac{X - X_{min}}{X_{range}}\right)n$$

- Since VADER score scales from -1 to 1, rescaling to generate a 50% weightage in recommendation score.
- Sentiment scores and star ratings are added to form the recommendation scores

RECOMMENDATION – COLLABORATIVE FILTERING

Applying KNN model to user-product map matrix

```
from scipy.sparse import csr_matrix

def create_matrix(df):

    N = len(new_df['customer_id'].unique())
    M = len(new_df['product_id'].unique())

    # Map Ids to indices
    user_mapper = dict(zip(np.unique(new_df["customer_id"]), list(range(N))))
    product_mapper = dict(zip(np.unique(new_df["product_id"]), list(range(M))))

    # Map indices to IDs
    user_inv_mapper = dict(zip(list(range(N)), np.unique(new_df["customer_id"])))
    product_inv_mapper = dict(zip(list(range(M)), np.unique(new_df["product_id"])))

    user_index = [user_mapper[i] for i in new_df['customer_id']]
    product_index = [product_mapper[i] for i in new_df['product_id']]

    X = csr_matrix((new_df["vader_sentiment_score"], (product_index, user_index)), shape=(M, N))

    return X, user_mapper, product_mapper, user_inv_mapper, product_inv_mapper

X, user_mapper, product_mapper, user_inv_mapper, product_inv_mapper = create_matrix(new_df)
```

```
from sklearn.neighbors import NearestNeighbors
"""
Find similar products using KNN
"""
def find_similar_products(product_id, X, k, metric='cosine', show_distance=False):

    neighbour_ids = []

    product_ind = product_mapper[product_id]
    product_vec = X[product_ind]
    k+=1
    kNN = NearestNeighbors(n_neighbors=k, algorithm="brute", metric=metric)
    kNN.fit(X)
    product_vec = product_vec.reshape(1,-1)
    neighbour = kNN.kneighbors(product_vec, return_distance=show_distance)
    for i in range(0,k):
        n = neighbour.item(i)
        neighbour_ids.append(product_inv_mapper[n])
    neighbour_ids.pop(0)
    return neighbour_ids


product_titles = dict(zip(new_df['product_id'], new_df['product_title']))
```

- KNN can calculate the feature similarity distance between a target item with others in the database, hence returning K-nearest products as the most similar product recommendations.

SEARCH ENGINE & WEB APP DEMO

Try the Web App here!

- The final outcome ➔ creating a search engine-like tool to allow customers to search for their relevant products.

CONCLUSION

- Framework to apply sentiment analysis as part of recommendation systems.
- Quantify the sentiment of the reviews through sentiment scoring
- Combining explicit feedback (ratings/review) and implicit feedback (KNN similarity between products and customers)

FUTURE RECOMMENDATIONS

- Only collaborative filtering done, effect of sales velocity and content-based filtering by using sentiment analysis is yet to be explored.
- Optimization study of the weightages between the 5-star rating score and the sentiment scores
- Create weightage for the dates of the reviews because recent reviews should be prioritised

THANKS