Study - AI Content in Amazon Reviews

September 12, 2023

1 Overview

Approximately 2,000 records of randomly selected Amazon.com Product Reviews from 2020 to date were processed through Originality.AI to determine the probability of AIContent. A series of experiments were performed on this data, to get more information about the relationship between AIContent and the different features of the reviews. This notebook describes some of the experiments done. The purpose of these experiments is to answer the following questions:

1. Is there a correlation between the severity of a review and the tested quantity of aiContent? Or in other words, are more extreme Reviews (5 - very good or 1 - very bad) more or less likely to have aiContent?
2. Is there any relationship between the helpfulness of a Review and its AIContent?
3. Has there been an increase in AIContent in Amazon reviews since the introduction of Chat GPT in the last Quarter of 2022?

[1]:
```
# import custom helper functions
# also imports the usual package libraries like pandas, numpy, etc
from helperfiles import *
```

2 The Data

Almost 27K records of raw data were scraped from Amazon. The data was cleaned using the standard processes, and customer sensitive information was removed. Approximately 2K were processed by the through Originality.AI's state-of-the-art AI detector to provide viable records for analysis. Statistical tests were performed on the dataset to confirm that the data used for the analysis, was representative of the original raw data. This is detailed in the Appendix.

[2]:
```
df = get_data_for_analysis()
display(df.info())
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2060 entries, 0 to 2059
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   date                  2060 non-null   datetime64[ns]
 1   isVerified            2060 non-null   bool
```

```
 2   ratingScore           2060 non-null   int64
 3   totalCategoryRatings  2060 non-null   int64
 4   totalCategoryReviews  2060 non-null   int64
 5   helpfulCount          2060 non-null   int64
 6   aiContent             2060 non-null   float64
 7   productGroup          2060 non-null   object
 8   averageRating         2060 non-null   float64
 9   goodreadsRating       241 non-null    float64
dtypes: bool(1), datetime64[ns](1), float64(3), int64(4), object(1)
memory usage: 147.0+ KB

None
```

[2]:

	date	isVerified	ratingScore	totalCategoryRatings	totalCategoryReviews	helpfulCount	aiContent	productGroup	averageRating	goodreadsRating
0	2023-08-17	False	1	18	14	3	0.0006	Children Books	4.8	4.6
1	2023-07-07	True	5	432	64	33	0.0001	Children Books	4.8	4.6
2	2023-07-12	True	5	432	64	15	0.0008	Children Books	4.8	4.6
3	2023-07-19	True	5	432	64	19	0.0002	Children Books	4.8	4.6
4	2022-10-07	True	5	432	64	26	0.0004	Children Books	4.8	4.6

3 The Experiments

3.1 Extreme reviews vs likelihood of aiContent

This experiment investigates if a review that is extremely rated (1 or 5) is more likely to have `aiContent` than a moderately rated (2, 3 or 4) review.

3.1.1 Process:

1. the reviews are categorized into a `ratingSeverity` binary feature: Extreme (1s and 5s) or Moderate (2s, 3s, 4s)
2. visual anslysis is performed by plotting the barplot of this feature against the `aiContent`.
3. statistical analysis is performed by the ANOVA (Analysis of Variance), the common method of comparing a categorical feature against a numerical feature.

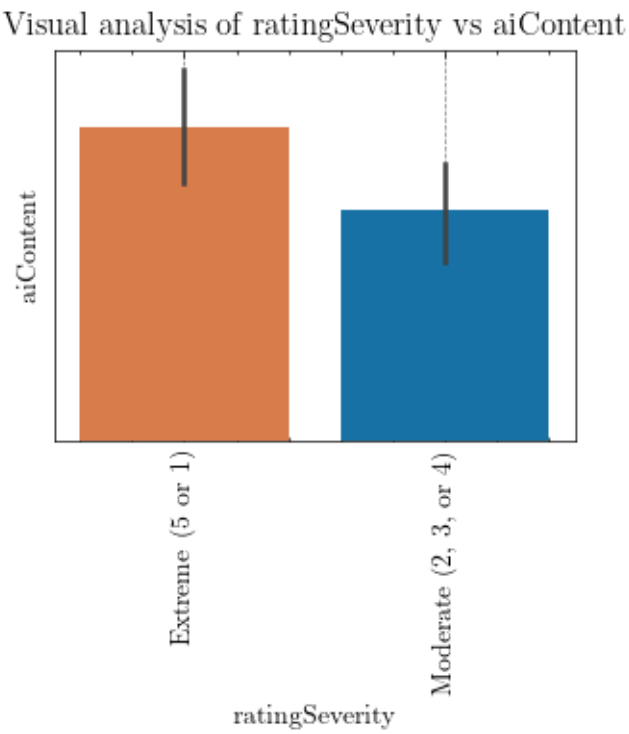
[3]:
```
from scipy.stats import spearmanr, kendalltau, pearsonr
import math
```

[4]:
```
# convert ratingScore into ratingSeverity category 'extreme', and 'moderate'
def severe(ratingScore):
    ratingSeverity = 'Extreme (5 or 1)' if (ratingScore==5 or ratingScore==1)␣
 ↪else \
                     'Moderate (2, 3, or 4)'
    return ratingSeverity
df['ratingSeverity'] = df['ratingScore'].apply(lambda x: severe(x))
df.head(3)
```

[4]:

	date	isVerified	ratingScore	totalCategoryRatings	totalCategoryReviews	helpfulCount	aiContent	productGroup	averageRating	goodreadsRating	ratingSeverity
0	2023-08-17	False	1	18	14	3	0.0006	Children Books	4.8	4.6	Extreme (5 or 1)
1	2023-07-07	True	5	432	64	33	0.0001	Children Books	4.8	4.6	Extreme (5 or 1)
2	2023-07-12	True	5	432	64	15	0.0008	Children Books	4.8	4.6	Extreme (5 or 1)

[5]:
```
# call the categorical testing function that plots the graph and performs the␣
 ↪ANOVA test
categorical_testing(df, 'ratingSeverity')
```

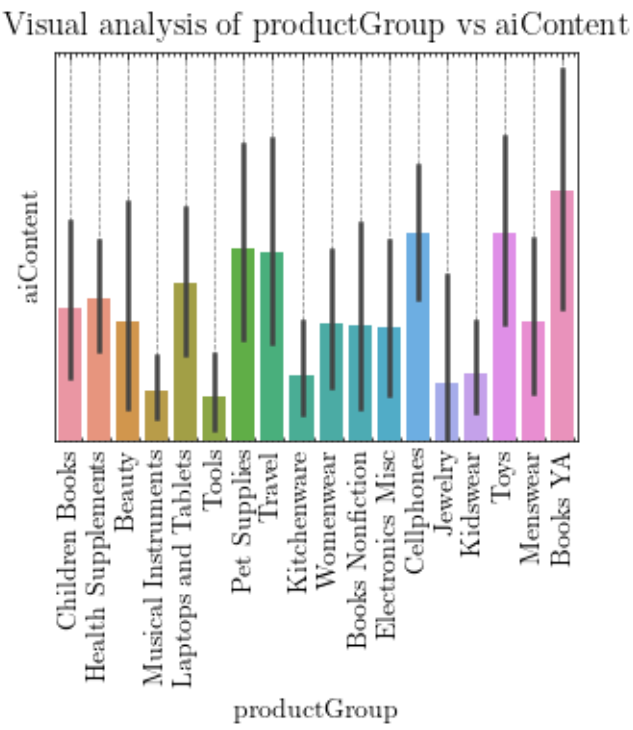



```
p value is 0.031420.
aiContent and ratingSeverity are correlated.
```

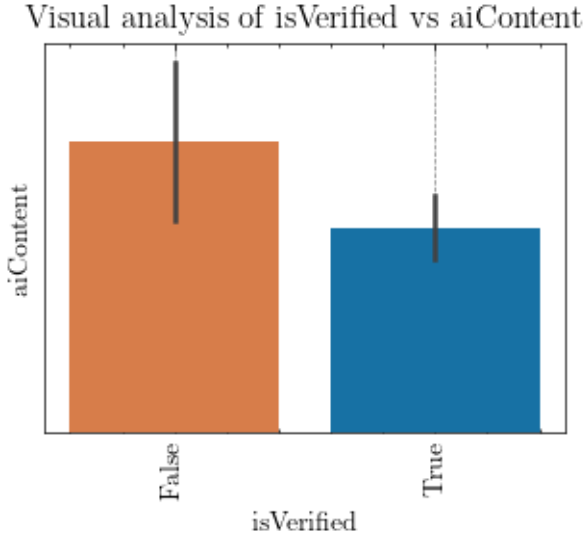
3.1.2 Analysis of other categorical features

Other categorical features were tested/analyzed:

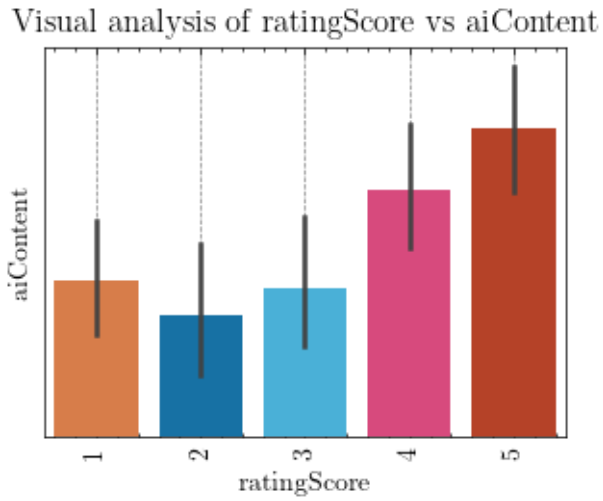
[6]:
```
categorical_cols = [
    'productGroup',
    'isVerified',
    'ratingScore'
]
for col in categorical_cols:
    categorical_testing(df, col)
```




```
p value is 0.020994.
aiContent and productGroup are correlated.
```




```
p value is 0.024949.
aiContent and isVerified are correlated.
```




```
p value is 0.000358.
aiContent and ratingScore are correlated.
```

3.1.3 Summary of Findings

- Extreme reviews (rated 1 and 5) are more likely to be detected with AI Content than moderate reviews (2, 3, 4).
- Verified reviews are less likely to be detected with AI Content.
- productGroup variation is not conclusive because of the overlapping between categories (e.g. a jewelry box item can be tagged under both Jewelry and Travel)

3.2 Helpfulness votes and AI Content

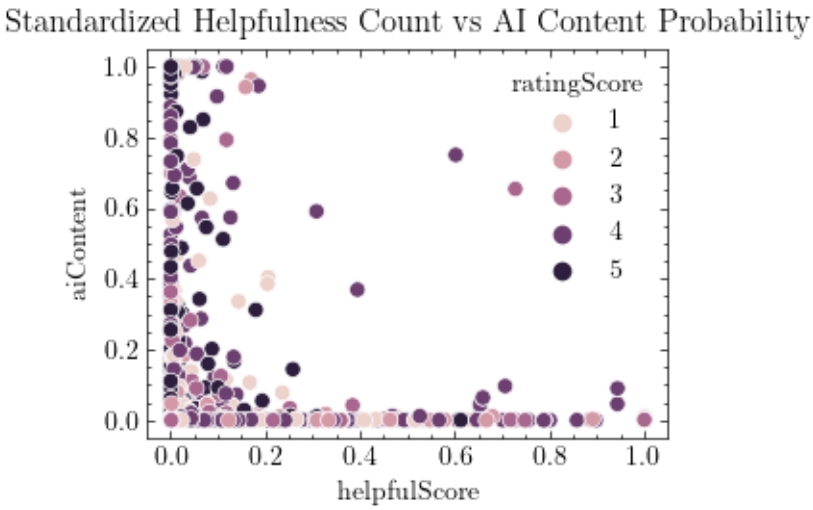
`helpfulCount` represents the votes that are given to a review by logged-in Amazon subscribers. This experiment wants to observe if there is any correlation between a review being widely regarded as helpful and its aiContent. It also considers the impact the age of the review has on its `helpfulCount` and therefore the correlation with `aiContent`.

3.2.1 Process:

1. Standardizing to `helpfulScore`: Due to the wide variety of the sample data, the range of values of the `helpfulCount` of each record is extreme. To properly analyze this relationship, the `helpfulCount` has to be standardized against the total Ratings and total Reviews of the product item that is being reviewed.
2. Visualizing the relationship between helpfulScore and aiContent with a scatter plot.
3. Statistics testing with the 3 common correlation tests (Spearman's Rho, Kendall Tau, and Pearson R) to check for a correlation between these features.
4. Extracting the age of the review into a new feature `days`.
5. Performating visual and statistical analysis of `days` and `helpfulScore` by the methods describes in (2) and (3)
6. Creating an OLS model to analyze the multivariate relationship between the three features.

[7]:
```
# 1. standardizing the Score
df['helpfulScore'] = list(map(generate_helpfulScore,
        df.helpfulCount,
        df.totalCategoryRatings,
        df.totalCategoryReviews))

# 2 & 3. call the numerical testing function that plots the scatter plot and␣
 ↪returns the stats results
numerical_testing(df, 'helpfulScore')
```

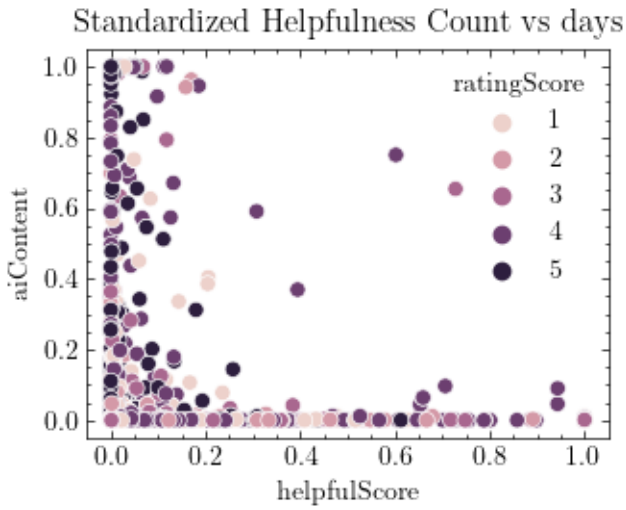



```
Results of the Spearman test: Correlation is -0.09721369907134281, with a
p-value of 0.00001.
aiContent and helpfulScore are correlated.

Results of the Kendall Tau test: Correlation is -0.06742367408043363, with a
p-value of 0.00001.
aiContent and helpfulScore are correlated.

Results of the Pearson test: Correlation is -0.059739531243374236, with a
p-value of 0.00668.
aiContent and helpfulScore are correlated.
```

[8]:
```
# 4. getting the days feature
df['days'] = pd.to_datetime('2023-08-21') - df['date']
df['days'] = df['days'].dt.days
# 5. statistical and visual analysis
numerical_testing(df, 'helpfulScore', col_2='days')
```




```
Results of the Spearman test: Correlation is 0.24150283708349082, with a p-value
of 0.00000.
days and helpfulScore are correlated.

Results of the Kendall Tau test: Correlation is 0.16391636607726975, with a
p-value of 0.00000.
days and helpfulScore are correlated.

Results of the Pearson test: Correlation is 0.06938210584330276, with a p-value
of 0.00163.
days and helpfulScore are correlated.
```

[9]:
```python
# 6. modelling with aiContent and days

import pandas as pd
import statsmodels.api as sm

# Create the OLS model
X = df[['days', 'aiContent']]
y = df['helpfulScore']
X = sm.add_constant(X)
model = sm.OLS(y, X)

# Fit the model and print the summary statistics
print(model.fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           helpfulScore   R-squared:                       0.008
Model:                            OLS   Adj. R-squared:                  0.007
```

```
Method:                 Least Squares   F-statistic:                     7.835
Date:                Tue, 12 Sep 2023   Prob (F-statistic):           0.000407
Time:                        10:05:47   Log-Likelihood:                643.29
No. Observations:                2060   AIC:                            -1281.
Df Residuals:                    2057   BIC:                            -1264.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0719      0.005     14.567      0.000       0.062       0.082
days        1.624e-05   5.65e-06      2.876      0.004    5.17e-06    2.73e-05
aiContent     -0.0479      0.020     -2.386      0.017      -0.087      -0.009
==============================================================================
Omnibus:                     1583.900   Durbin-Watson:                   1.118
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            22414.227
Skew:                           3.691   Prob(JB):                         0.00
Kurtosis:                      17.375   Cond. No.                     4.31e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.31e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

3.2.2 Analysis and Summary of Findings:

- The three statistical tests and the visualization show that there is a statistically significant but quantitatively small negative correlation between the helpfulness of a review and its AI Content.
- There is also a statistically signficant correlation with `days`, the age of the Review.
- However this is relatively small compared to the negative correlation to `aiContent`.

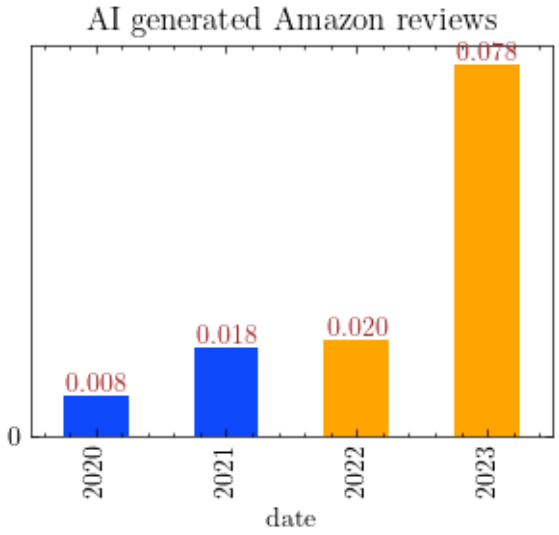
3.3 What is the trend of aiContent volume since Chat GPT launch?

This experiment observes the trend of aiContent volume prior to and after the launch of Chat GPT Launch.

3.3.1 Process:

1. The `aiContent` numerical feature was categorized into a binary feature: 0 for less than 50% and 1 for greater than 50%.
2. The timeseries plot of annual daily `aiContent` for the past 4 years was visualized and studied.

[10]:
```
df.aiContent = df.aiContent.apply(lambda x: 1 if x>0.5 else 0 if x < 0.5 else␣
↪None)
time_df = generate_df_plot(df, start='2020') # returns this as a timeseries
```



3.3.2 Summary of Findings

- After a steady rate of neglible `aiContent`, the **trend** began increasing from late 2022 and has been on the rise since then.
- In the year following GPT Launch, `aiContent` jumped by approximately 4 times its average value.

4 Appendix

4.1 Checking sample to population distribution

In order to generalize the results we get from analyzing the subset `df` of ~2K records to the set of 27K records collected, we need to confirm if it represents the same distribution.

Note that: this only generalizes to the larger dataset, not to the entire corpus of Amazon reviews as that dataset is not available to the public.

4.1.1 Process:

1. Visual analysis by plotting the bargraphs of the distributions.
2. Statistical analysis by using the KS and Chisquare power divergence tests.

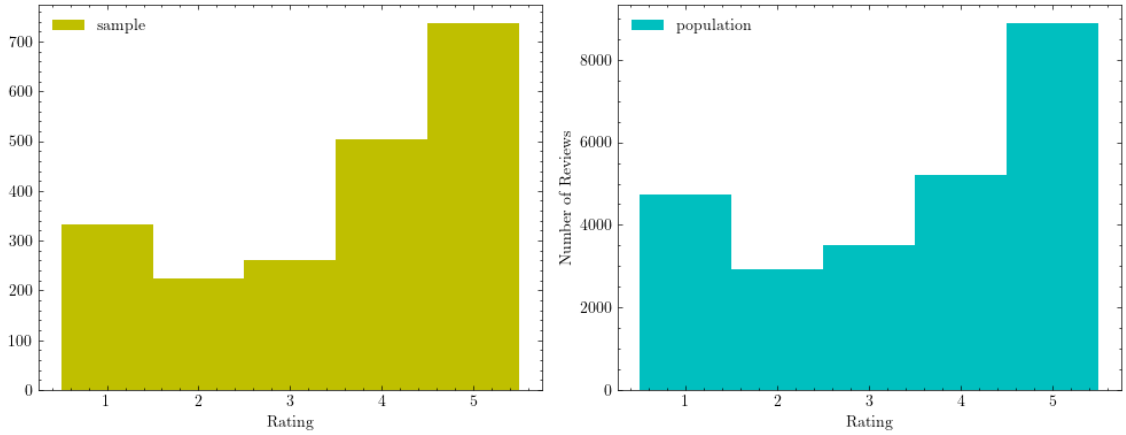
[11]:
```
df_population = get_raw_data()
df_population.date = pd.to_datetime(df_population.date)
df_population = df_population[df_population.date.dt.year>=2019]
df_sample = get_data_for_analysis()
print(f"Population size is {len(df_population)}.")
print(f"Sample size is {len(df_sample)}.")
print(f"The sample is {len(df_sample)/len(df_population)*100:.1f}% of the␣
 ↪population.")
```

```
# extract ratingScore from both distributions
sample = df_sample.ratingScore
population = df_population.ratingScore

dists = [sample, population]
titles = ['sample', 'population']
colors = ['y','c']

# plot the bar graphs and extract the frequency distribution for statistics␣
 ↪testing
counts_for_testing = plotbars(dists, titles, colors)
```

```
Population size is 25285.
Sample size is 2060.
The sample is 8.1% of the population.
```



[12]:
```
# statistics testing
from scipy.stats import ks_2samp, chisquare, entropy
# normalize the distributions
x = counts_for_testing[0]/counts_for_testing[0].sum()
y = counts_for_testing[1]/counts_for_testing[1].sum()

display(ks_2samp(x, y))
display(chisquare(x, y))
kld = entropy(x, y)
print("KullbackLeibler divergence",kld)
```

```
KstestResult(statistic=0.2, pvalue=1.0, statistic_location=0.20652560806802453,␣
 ↪statistic_sign=-1)

Power_divergenceResult(statistic=0.012194273193480093, pvalue=0.9999814878440287)

KullbackLeibler divergence 0.006009413073349793
```

4.1.2 Summary of Findings

Visual Analysis: The two distributions are very similar. With very close examining, one will only notice that Rating '4' has a small delta across the 2 distributions.

The null hypothesis or prevailing assumption of KS and Chisquare tests is that these two distributions are the same, and therefore represent samples from the same source. For this to be challenged, the p-values of these tests should be less than 0.05, which is the usual value set for alpha - the level of statistical significance.

As observed, they both have p-values > 0.05, showing that the Hypothesis that these the sample subset is representative of the population of raw data, can be assumed.

KLD measures how much one distribution differs from the other. The low KLD score (<0.01) indicates that the two distributions are very similar, which corresponds with the other statistics tests.