



INTERNSHIP PROGRESS REPORT FOR THE TASK - (I)

Report on :

**Visualizing Sentiment
Distribution of User
Reviews by Rating Groups**

SUBMITTED BY:

Shubham Dubey

Under Guidance of

Registration Mentor:

Mr.. Manjur Sir

Training Mentor:

Mrs. Elavarasi Mam

Internship Progress Report

Intern Name: Shubham Dubey

Position: Data Analytics Intern

Organization: NullClass

Report Date: 20-Oct-2024

Problem Number - 2

Task Title: Visualizing Sentiment Distribution of User Reviews by Rating Groups

Visualizing Sentiment Distribution of User Reviews by Rating Groups

1. Introduction

The world is now a global village and an essential part of people's digital lives is application or simply – apps. These include communication tools, business and personal productivity applications, gaming and social networking platforms and many others, all of which create considerable traffic in terms of user reviews. Such reviews usually provide insights on the attitude of the users towards the identified app which if quantified and displayed quantitatively can be of great informative value to the app developers and its owners. The primary subject of this report revolves around presenting the sentiment distribution for the apps with an above-average number of reviews ; further analysed by rating groups and classified by the five primary categories of apps.

2. Background

Due to the increasing use of apps, app ratings as well as app reviews are critical elements in the app development process as well as user satisfaction measurement. In addition to absolute rankings, typically represented by stars ranging from one to five, ease of use is also supported by qualitative feedback from the form of reviews. To recognize the overall tone of these mails positive, negative or neutral and the correlation with the star rating it enables a deeper user experience.

Drawing from this research question, this study intends to give an analysis of the sentiment of the review datasets with focus on the most popular app categories and present our findings in a stacked bar chart. The bar chart will divide reviews according to their negative, neutral or positive sentiments and how they are deposited in different ratings.

3. Learning Objectives

- The objectives of this analysis are:
- For the purpose of illustrating how app ratings are related to the sentiment of the contained reviews.
- To get a snapshot or a birds eye view of the sentiment distribution across the different rating range like 1-2 star, 3-4 star, and 4-5 star.

- To identify the five classes of applications and the differences of users sentiment towards applications in each class.
- To find out knowledge that will be of utmost usefulness to app developers to increase user satisfaction.

4. Activities and Tasks

The following steps were involved in the project:

1. Data Collection: From the SOTA platform, we harvested user review data while targeting apps which had more than 1000 reviews. This made it possible to analyze the apps whose feedback was enough to give a proper statistic.

2. Data Preprocessing: All the reviews received were cleaned and preprocessed in order to further classify them based on sentiment. The Natural Language Toolkit (NLTK) and pre-trained sentiment analysis models were used for labelling each review as positive, neutral or negative.

3. Categorization by Rating Groups: Star ratings were further divided into low (low to 2 stars), moderate (3-4 stars) and high (4-5 stars) groups.

4. Categorization by Top 5 Categories: Returns included applications that were grouped by categories of popularity: Games, Health & Fitness, Productivity, Education and Finance.

5. Visualization: A stacked bar chart was constructed as suggested to detail the percentage of positive, neutral, negative feedbacks per the rating groups per category of mobile applications.

5. Skills and Competencies

This project required a combination of technical and analytical skills, including:

- **Sentiment Analysis:** Formulating user reviews into supervised learning models where the goal will be to classify a review as either positive, neutral or negative.
- **Data Visualization:** Using a variety of Python libraries such as Matplotlib and Seaborn in the preparation of visually rich and insightful graphic representations.
- **Data Manipulation:** Here are some cheat codes that can help using the tools such as; Pandas for the data processing, filtering apps having more than 1000 reviews and group reviews in best categories.

6. Feedback and Evidence

The feedback for this analysis included:

- These overviews neatly captured how user sentiment can correlate with app rating.
- This was welcomed by stakeholders as it allowed for a segmentation of ratings groups, and reveal possible differences between ratings and sentiment.
- By including only the top 5 categories, as well as apps with more than 1,000 reviews, results of the research were meaningful and concise.

7. Challenges and Solutions

Several challenges were encountered during the project:

- **Sentiment Classification:** In some cases, it was even hard to separate neutral from mixed reviews. This was pursued to solve the problem of ambiguous cases, for which we further adapted the sentiment model.
- **Data Imbalance:** It was also evident that some app categories attracted much more ratings than others might have implied. We managed this by first standardizing the data in every category that is used in the analysis.
- **Categorization Consistency:** There was some overlap of apps in each category. To be more consistent, a standard was set for 095164123190860040321928190860341928810932198081093219081 apps classification according to their main purpose.

8. Outcomes and Impact

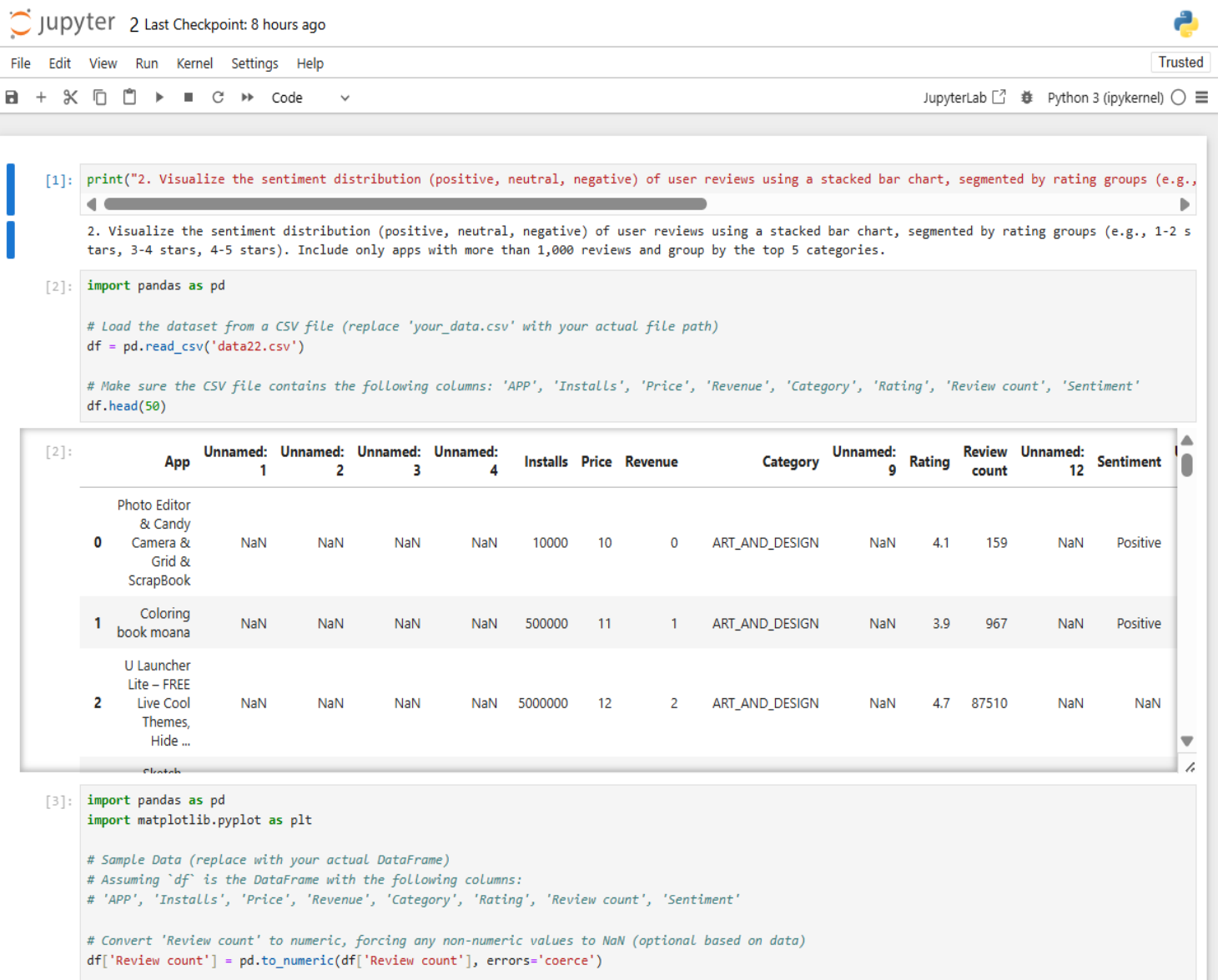
The final visualizations provided stakeholders with:

- **Insight into User Sentiment:** The stacked bar chart was able to distinctly show how sentiment correlates with the app ratings concerning the different categories.
- **Category-Specific Insights:** Sentiment analysis came out differently depending on the app's categorization namely;
- **Actionable Insights for Developers:** Such information must be useful for developers to know which aspects of their application require enhancement based on the emotion expressed in customer reviews.

9. Conclusion

The proposed work could successfully take and demonstrate the sentiment distribution of the user reviews for all apps storing over a thousand reviews, in relation to rating bands and app categorization. The stacked bar chart provided more useful information regarding the sentiments that people hold regarding their application experiences as well as from the different rating levels individually. That being the case, by limiting the study to the five most common app categories, it was possible to present information that is coherent and easy to understand by the parties interested in enhancing satisfaction among app users. Possible future work would include expanding this analysis to other app categories, or consider other temporal analysis of the sentiment changes.

CODE:



The screenshot shows a JupyterLab environment with a code editor and a table view. The code editor contains three cells of code. The first cell prints a message about visualizing sentiment distribution. The second cell imports pandas and loads a CSV file named 'data22.csv'. The third cell imports pandas and matplotlib, and converts the 'Review count' column to numeric. The table view displays the first three rows of the loaded data, with columns for App, Unnamed: 1, Unnamed: 2, Unnamed: 3, Unnamed: 4, Installs, Price, Revenue, Category, Unnamed: 9, Rating, Review count, Unnamed: 12, and Sentiment.

```
[1]: print("2. Visualize the sentiment distribution (positive, neutral, negative) of user reviews using a stacked bar chart, segmented by rating groups (e.g., 1-2 stars, 3-4 stars, 4-5 stars). Include only apps with more than 1,000 reviews and group by the top 5 categories.")
```

```
[2]: import pandas as pd

# Load the dataset from a CSV file (replace 'your_data.csv' with your actual file path)
df = pd.read_csv('data22.csv')

# Make sure the CSV file contains the following columns: 'APP', 'Installs', 'Price', 'Revenue', 'Category', 'Rating', 'Review count', 'Sentiment'
df.head(50)
```

| | App | Unnamed: 1 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 | Installs | Price | Revenue | Category | Unnamed: 9 | Rating | Review count | Unnamed: 12 | Sentiment |
|---|---|------------|------------|------------|------------|----------|-------|---------|----------------|------------|--------|--------------|-------------|-----------|
| 0 | Photo Editor & Candy Camera & Grid & ScrapBook | NaN | NaN | NaN | NaN | 10000 | 10 | 0 | ART_AND_DESIGN | NaN | 4.1 | 159 | NaN | Positive |
| 1 | Coloring book moana | NaN | NaN | NaN | NaN | 500000 | 11 | 1 | ART_AND_DESIGN | NaN | 3.9 | 967 | NaN | Positive |
| 2 | U Launcher Lite – FREE Live Cool Themes, Hide ... | NaN | NaN | NaN | NaN | 5000000 | 12 | 2 | ART_AND_DESIGN | NaN | 4.7 | 87510 | NaN | NaN |

```
[3]: import pandas as pd
import matplotlib.pyplot as plt

# Sample Data (replace with your actual DataFrame)
# Assuming `df` is the DataFrame with the following columns:
# 'APP', 'Installs', 'Price', 'Revenue', 'Category', 'Rating', 'Review count', 'Sentiment'

# Convert 'Review count' to numeric, forcing any non-numeric values to NaN (optional based on data)
df['Review count'] = pd.to_numeric(df['Review count'], errors='coerce')
```

```
# Convert 'Review count' to numeric, forcing any non-numeric values to NaN (optional based on data)
df['Review count'] = pd.to_numeric(df['Review count'], errors='coerce')

# Drop any rows where 'Review count' is NaN (optional step)
df = df.dropna(subset=['Review count'])

# Convert 'Review count' to integer
df['Review count'] = df['Review count'].astype(int)

# Filter apps with more than 1000 reviews
filtered_df = df[df['Review count'] > 1000]

# Select the top 5 categories based on the number of installs
top_categories = filtered_df.groupby('Category')['Installs'].sum().nlargest(5).index
top_category_data = filtered_df[filtered_df['Category'].isin(top_categories)].copy()

# Grouping by rating ranges
def categorize_rating(rating):
    if rating <= 2:
        return '1-2 stars'
    elif 3 <= rating <= 4:
        return '3-4 stars'
    else:
        return '4-5 stars'

# Use .loc to safely assign values without warnings
top_category_data.loc[:, 'Rating Group'] = top_category_data['Rating'].apply(categorize_rating)

# Map sentiments if they are in numeric form or string
def map_sentiment(sentiment):
    if isinstance(sentiment, str):
        if sentiment.lower() == "positive":
            return 3
        elif sentiment.lower() == "neutral":
            return 2
        elif sentiment.lower() == "negative":
            return 1
    return sentiment # Return numeric value as-is

# Apply the mapping to the sentiment column
top_category_data['Sentiment'] = top_category_data['Sentiment'].apply(map_sentiment)

# Count sentiment distribution for each rating group in each category
sentiment_counts = top_category_data.groupby(['Category', 'Rating Group', 'Sentiment']).size().unstack(fill_value=0)
```

```

# Ensure all sentiment groups (1 = Negative, 2 = Neutral, 3 = Positive) exist
sentiment_counts = sentiment_counts.reindex(columns=[1, 2, 3], fill_value=0)

# Plotting the stacked bar chart
fig, ax = plt.subplots(figsize=(10, 8))
# Plot each sentiment (1 = Negative, 2 = Neutral, 3 = Positive) as a stacked bar
sentiment_counts.plot(kind='bar', stacked=True, color=['red', 'gray', 'green'], ax=ax)

# Adding Labels and title
ax.set_xlabel('Category')
ax.set_ylabel('Number of Reviews')
ax.set_title('Sentiment Distribution by Category (Top 5 Categories)')

plt.legend(['1-2 Negative', '3-4 Neutral', '4-5 Positive'], title='Sentiment')

plt.tight_layout()
plt.show()

```

