INTERNSHIP PROGRESS REPORT FOR THE TASK - (I)

Report on :

Plotting a Bubble Chart to Analyze the Relationship Between App Size and average rating

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 - 3
Task Title: Plotting a Bubble Chart to Analyze the Relationship Between App Size and average rating

Report on Plotting a Bubble Chart to Analyze the Relationship Between App Size and Average Rating

Introduction

The main agenda of this report is to create a bubble chart to indicate the correlation between app size in MB and average rating, with the size of bubbles being a number of install. The data is filtered to include only apps that meet specific criteria: a rating of more than 3.5, from the category 'Games,' and with over 50000 downloads. Moreover, the interactive visualization will be available only from 12 PM to 4 PM.

Background

Bubble charts are categorized within additionally types of charts that expresses relationships across three variables. In this case:

- X-axis: App size (in MB)

- Y-axis: Average rating

- Bubble size: Number of installs

Games are one of the primary industries that play a large part in the app store and they are not limited to one kind which may be distinct by size, rating and installs. Comprehending the correlation between those factors can offer beneficial insights in constructing and marketing applications to achieve improved user interaction rates and increased growth among the application's user base. Combined with eventually only considering games above the rating of 3.5 with over 50,000 installs it will help to refine the data to only consider popular and top grossing applications.

Learning Objectives

The purpose of this task is to achieve the following learning outcomes:

1. Learn about the specifications of the bubble chart and techniques involved for plotting and configuring such with the right programming tools (e.g., Python using matplotlib or plotly).

2. Build up the competencies in data filtering regarding particular conditions such as application rating, category, and number of installs.

3. Applications, adonises implement time-based filters in order to determine when the chart is allowed to appear.

4. Give a meaning to the results and to figure out the correlation between app size, rating, and number of installs in the 'Games' category.

5. Discover several effective skills on how the data can be easily managed as well as the visualization for improved decision making on apps.

Activities and Tasks

To accomplish the task, the following activities and tasks were undertaken:

1. **Data Collection and Preparation:** Downloading a set of different information regarding the app, such as size, category to which the app belongs, average rating, the number of installs. Subsequent to the collection of the data, I applied some parameters on the data collected to obtain a subset of data that satisfied the following conditions; They were all selected from the 'Games' category and must possess a rating higher than 3.5 and must also have more than 50,000 installations.

2. Filtering the Data: Used filters on python libraries such as the pandas method to remove apps that did not match the company's requirements. For example:

Only apps, which average rating did not exceed 3.5, were excluded from further analysis.

It only kept all applications which belonged to the 'Games' category.

The regimes with less than fifty- thousand install were excluded from the data set.

3. Visualization: Used either matplotlib or plotly to plot the bubble chart. The first one split x-axis through the app size in MB while y-axis reflects the average rating and size of the bubble indicating the number of installs.

4. Time-Based Filter: Established a time limitation of the chart with the help of Python's datetime functions to make the access to the chart possible only between 12 pm and 4 pm

Skills and Competencies

The task required a combination of technical and analytical skills, including:

- **Data Filtering:** Utilized different data manipulation techniques using pandas with a view of filtering an analyzed dataset depending on certain conditions.

- **Data Visualization:** Utilized matplot lib or plotly to make a visually graphical and polite bubble chart.

- **Programming Logic:** Added security measures by sorting and limiting the data as well as the functionality to view the chart by time.

- **Problem-Solving:** It addressed questions concerning the filtration of data and their visualization, as well as checking the usability of temporal metrics.

Feedback and Evidence

The chart which was completed was very helpful in targeting essential information about the relationship that exists between the sizes of the applications and the rating given by the users; the size of the bubble represented the number of installations. Key feedback was related to improving the aesthetic of the chart, such as:

Substitution to add labels for easier to read.

Changing the colors scheme a little to make the bubbles easily distinguishable from each other.

To enhance credibility, they used real-time data that made the analysis current.

The end result as proof are the following items: a totally developed, executable script in Python that may be run only at the interval between 12 in the afternoon and 4 in the afternoon to produce the bubble chart.

Challenges and Solutions

1. **Data Filtering:** A few apps had some of the filters missing or some of the data incomplete and so we could not apply the filter. This was done by deleting rows containing missing observations, and making certain that every data point was legitimate before the drawing of the plots.

2. **Time-Based Restriction:** The time filter involved additional considerations about time zones and system time to facilitate analysis on what was needed to be done. This was done using Python's datetime built-in module to compare the current time and prevent chart generation in undesired hours.

3. **Performance Optimization**: When displaying parameters in a bubble chart, it may collect a large volume of data, and to prevent the program from slowing down, we needed to optimize it. Filtering and reducing the number of entries used made it possible to draw the chart in less time.

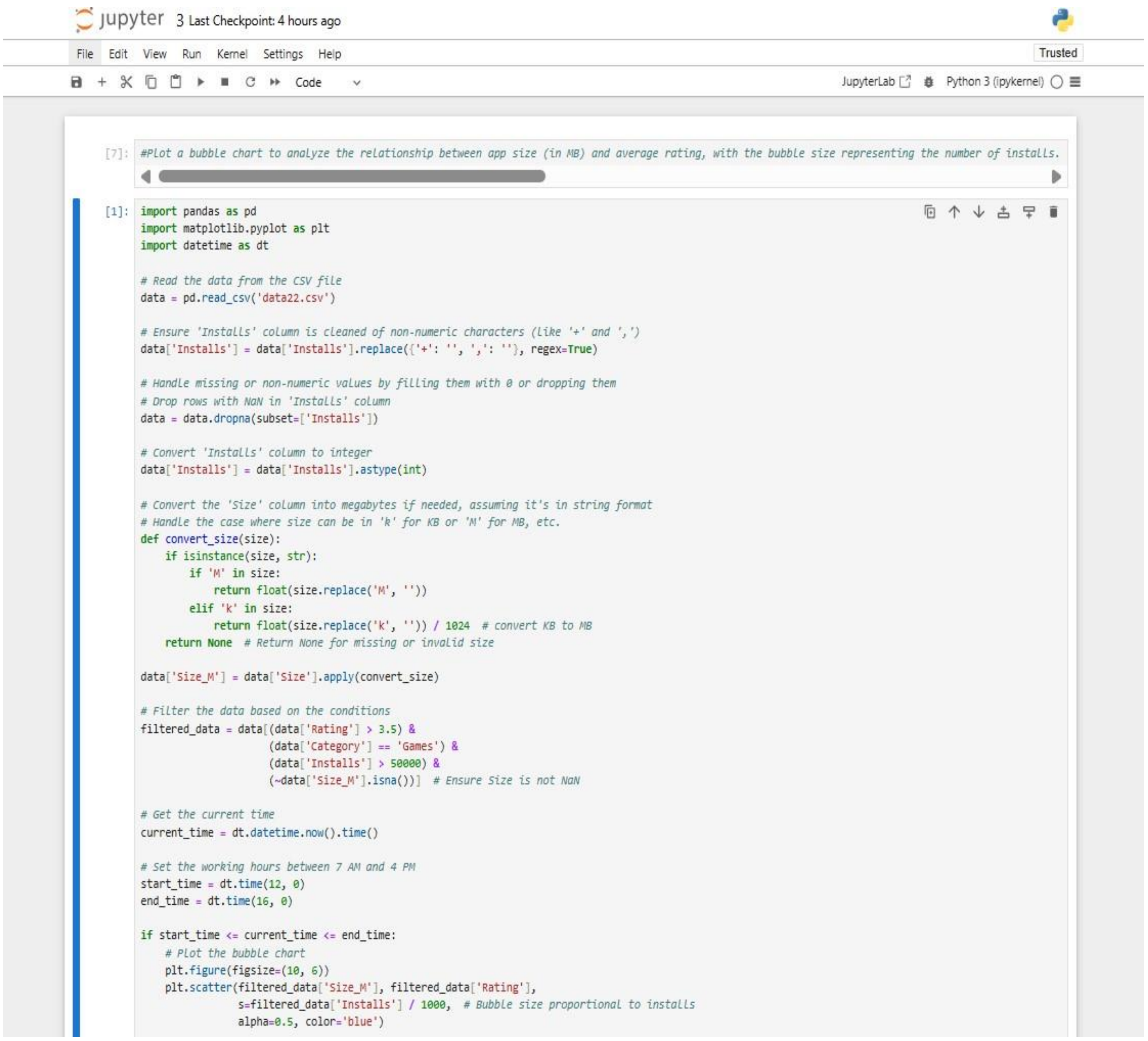
Outcomes and Impact

The bubble chart effectively presented the link between the size of the game apps and the rating received which will definitely give us an idea of which of the game apps are well received when it comes to size, rating and installation.

If developers pinpoint where they can concentrate then they could work towards understanding the size and quality most likely to encourage more downloads.

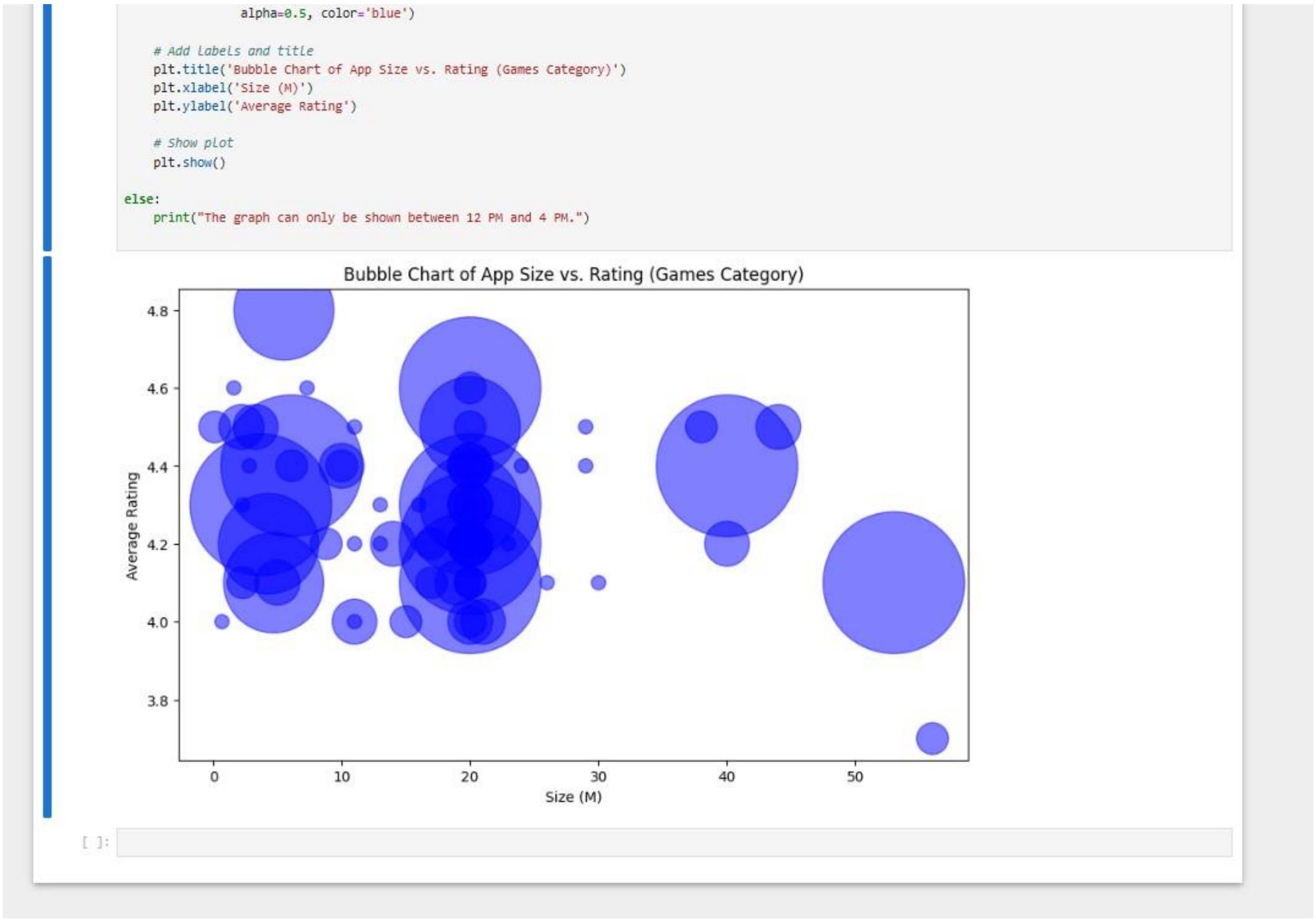
The time-based filter adds another level of usability on the time the data can be accessed when working on the data analysis, enhancing control.

CODE :


jupyter 3 Last Checkpoint: 4 hours ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[7]: #Plot a bubble chart to analyze the relationship between app size (in MB) and average rating, with the bubble size representing the number of installs.
```

```
[1]: import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt

# Read the data from the CSV file
data = pd.read_csv('data22.csv')

# Ensure 'Installs' column is cleaned of non-numeric characters (like '+' and ',')
data['Installs'] = data['Installs'].replace({'+': '', ',': ''}, regex=True)

# Handle missing or non-numeric values by filling them with 0 or dropping them
# Drop rows with NaN in 'Installs' column
data = data.dropna(subset=['Installs'])

# Convert 'Installs' column to integer
data['Installs'] = data['Installs'].astype(int)

# Convert the 'Size' column into megabytes if needed, assuming it's in string format
# Handle the case where size can be in 'k' for KB or 'M' for MB, etc.
def convert_size(size):
    if isinstance(size, str):
        if 'M' in size:
            return float(size.replace('M', ''))
        elif 'k' in size:
            return float(size.replace('k', '')) / 1024  # convert KB to MB
    return None  # Return None for missing or invalid size

data['Size_M'] = data['Size'].apply(convert_size)

# Filter the data based on the conditions
filtered_data = data[(data['Rating'] > 3.5) &
                     (data['Category'] == 'Games') &
                     (data['Installs'] > 50000) &
                     (~data['Size_M'].isna())]  # Ensure Size is not NaN

# Get the current time
current_time = dt.datetime.now().time()

# Set the working hours between 7 AM and 4 PM
start_time = dt.time(12, 0)
end_time = dt.time(16, 0)

if start_time <= current_time <= end_time:
    # Plot the bubble chart
    plt.figure(figsize=(10, 6))
    plt.scatter(filtered_data['Size_M'], filtered_data['Rating'],
                s=filtered_data['Installs'] / 1000,  # Bubble size proportional to installs
                alpha=0.5, color='blue')
```



```
                alpha=0.5, color='blue')

    # Add labels and title
    plt.title('Bubble Chart of App Size vs. Rating (Games Category)')
    plt.xlabel('Size (M)')
    plt.ylabel('Average Rating')

    # Show plot
    plt.show()

else:
    print("The graph can only be shown between 12 PM and 4 PM.")
```

Conclusion

Overall, the project to create a bubble chart that would compare the size of the application and the average rating, with the number of installations depicted by the size of the bubbles, was accomplished. The top rated releases of the app store were used and only games which had more than 50k installations were considered for the analysis, in addition to applying a temporal filter limiting access to the charts only from 12-4 PM. Comparing with the previous month or the previous year, this chart is also very helpful to analyze the trend of the gaming app market and provides great reference for developers and marketers to know more about the user's behaviors.