Google Play Store: User Story 95 Report

**User would like a Jupyter Dashboard**

KenzieM

1. Introduction

For this report, I used data on apps from the Google Play Store to answer the user story: “User would like a Jupyter Dashboard.” This dashboard should contain a drop down menu with all the categories, and when a category is selected, the number of apps, mean rating, and total number of installs should display. The dashboard should also be able to display this information for all of the apps in the data set. The data contains 10,841 rows and 13 columns. The columns of this data set include information such as name, category, genre, rating, size, number of installations, price, content rating and more. Every row of the set represents an individual application available on the Google Play Store. In the following report, I will discuss how I transformed the data in order to answer the question, and provide some visual aids to better understand that answer.

1. Data

To begin, I needed to clean the master csv. This involved deleting several duplicate rows, as well as one row that had most information entered incorrectly. Once these rows were removed, I reset the index and inspected the columns. I found four columns with numerical data that were not converted upon download. I removed unnecessary symbols from this data, such as plus signs, units, commas, and dollar signs, and converted these to their correct data types. The size column needed some extra work as some of the apps were measured in kilobytes and others in megabytes. I’ve included the code I used for that transformation below.

|  |
| --- |
| def new\_size(x):  if 'k' in x:  return (float(x.replace('k', ''))) \* 1000  if 'M' in x:  return (float(x.replace('M', ''))) \* 1000000   df['Size'] = df['Size'].apply(new\_size) |

After completing these data type conversions, I saved the new clean csv in a separate file to work on this story.

1. Methods

In order to create the dashboard, I needed to work with all the rows of the data set and three of the columns. For this project, I kept all of the columns in the data set, because it did not affect how my code functioned and created less lines of code. I am not sure how this affects efficiency, however, and I will keep this in mind for future projects. I have displayed the block of code below, and in the next paragraph I will explain how it functions.

|  |
| --- |
| def display\_values(Category):  mean\_rating = df.groupby('Category').mean()['Rating']  app\_count = df.groupby('Category').count()['App']  installs = df.groupby('Category').sum()['Installs']  if (Category == 'ALL'):  print('Number of Apps: ', df.count()['App'])  print('Average Rating: ', df.mean()['Rating'])  print('Total Installs: ', df.sum()['Installs'])  else:  print('Number of Apps: ', app\_count[category])  print('Average Rating: ', mean\_rating[category])  print('Total Installs: ', installs[category])  a = pd.Series(['ALL']) categories = a.append(df['Category'].drop\_duplicates().sort\_values())  interact(display\_values,  Category = categories); |

I created a function that takes a parameter named Category as its input. Then I defined three variables inside the function that use the groupby method to create a pandas series containing each category as the index and the particular value (count, average rating, or total installs) for that category. I use a conditional statement to check if the input is ‘ALL’, which uses its own functions on the entire dataframe to produce a result. If the input is any of the categories, the else statement performs a search on the list, using the category name as an index and produces the desired output. After that I create a variable that contains a list of all the categories and appends the category ‘ALL’. Finally, I use the interact method from ipywidgets to create the drop down widget and the output fields.

The final step was to use the dashboard add-on to create a dashboard layout of this interactive function. Unfortunately, Jupyter Dashboard does not seem to function correctly in its dashboard layout. I can at times get the correct cell to display in the output, but I am not sure how to save and share the notebook in this fashion. The online documentation and help spaces for this application are sparse, but it could provide some awesome functionality if it worked. Please reference the Jupyter notebook titled GPS\_95\_dashboard to interact with the function in its current state..

1. Results

I was able to produce a functioning interactive cell that allows the user to quickly access the requested information about the data by category. I was not able to get this to display as a dashboard, but I believe there are some other libraries available in python that allow me to create dashboards and/or share interactive functions over the web.

1. Conclusion

In conclusion, I was able to produce the desired information, but I was not able to display it in the way the user requested. In the future, depending on the flexibility of the client, I may ask to do this project as a shiny app in R instead, as it seems to have better functionality, robust documentation, with many people online sharing information and help. I will also look into other ways to create dashboards using python, as I believe there are other libraries available to do this. I am happy to answer any further questions regarding this story and would love to hear any feedback you have.