Data analytics CA

Mark Smith S00171578

1.Description of the Dataset

Steam is a game distribution service made by Valve with its initial release on September 12, 2003. Steam allows users to purchase and download pc games in their online store users can then play these games on any pc when they have signed into the steam service.

Overview of the dataset

The dataset I have chosen for this analysis is Steam games features.

What type of information is in the dataset?

Collects information on the games they sell such as the price, User Recommendations, Metacritic scores and the games genre (e.g. Action, FPS or Indie) games can have multiple genres.

How/Where is was collected

I downloaded the steam games features csv off Moodle.

What type of columns are there?

ReleaseDate - The date the game was released.

RequiredAge - The minimum age a user should be before they play the game.

DemoCount - The number of demo version available.

DeveloperCount - The number of developers listed on the store page.

DLCCount - The number of dlc packs available for purchase on the store.

Metacritic - The Metacritic score of the game.

MovieCount - The number of movies available.

PackageCount - The number of packages with the game

RecommendationCount - The number of users who recommended the game.

PublisherCount - The number of publishers listed on the steam store.

ScreenshotCount - The number of screenshots uploaded to the store page.

SteamSpyOwners - The number of users that have Steam spy ownership

SteamSpyOwnersVariance - The variance between the Owners

SteamSpyPlayersEstimate- The estimated players

SteamSpyPlayersVariance- The variance of the estimated players

AchievementCount - The number of achievements in the game.

AchievementHighlightedCount - The number of rare achievements in the game.

ControllerSupport - If the game supports controller or not.

IsFree - If the game is free or not.

FreeVerAvail - If a version of the game is available or not.

PurchaseAvail - If there is a in game purchases in the game.

SubscriptionAvail - If there is a subscription available.

PlatformWindows - If the game supports windows or not.

PlatformLinux - If the game supports Linux.

PlatformMac - If the game supports Mac OS or not.

PCReqsHaveMin - If the developer has put minimum requirements for Windows to run the game or not.

PCReqsHaveRec - If the developer has put recommended requirements for Windows to run the game or not.

LinuxReqsHaveMin - If the developer has put minimum requirements for Linux to run the game or not.

LinuxReqsHaveRec - If the developer has put recommended requirements for Linux to run the game or not.

MacReqsHaveMin - If the developer has put minimum requirements for MacOS to run the game or not.

MacReqsHaveRec - If the developer has put recommended requirements for Linux to run the game or not.

CategorySinglePlayer - If the game belongs to the single player category or not.

CategoryMultiplayer - If the game belongs to the multiplayer category or not.

CategoryCoop - If the game belongs to the co-op category or not.

CategoryMMO - If the game belongs to the MMO category or not.

CategoryInAppPurchase - If the game has in game purchases or not.

CategoryIncludeSrcSDK - If the game uses the source software development kit or not.

CategoryIncludeLevelEditor - If the game includes a level editor or not.

CategoryVRSupport - If the game supports VR or not.

GenreIsNonGame - The item is not a game (movie soundtrack etc...) or not.

GenreIsIndie - If the game belongs to the Indie genre or not.

GenreIsAction - If the game belongs to the Action genre or not.

GenreIsAdventure - If the game belongs to the Adventure genre or not.

GenreIsCasual - If the game belongs to the Casual genre or not.

GenreIsStrategy - If the game belongs to the Strategy genre or not.

GenreIsRPG - If the game belongs to the RPG genre or not.

GenreIsSimulation - If the game belongs to the Simulation genre or not.

GenreIsEarlyAccess - If the game is in Early access or not.

GenreIsFreeToPlay - If the game is Free to play or not.

GenreIsSports - If the game belongs to the Sports genre or not.

GenreIsRacing - If the game belongs to the Racing genre or not.

GenreIsMassivelyMultiplayer - If the game belongs to the MassivelyMultiplayer genre or not.

PriceCurrency - The currency the price is in.

PriceInitial – The initial price for a game.

PriceFinal- The final price for a game.

SupportEmail - The user support email address.

AboutText - Text that describes the game.

ShortDescrip- A short description of the game.

DetailedDescrip- A detailed description of the game.

DRMNotice - A notice about if the game has DRM

HeaderImage- A item displayed on the games store page.

LegalNotice- The games legal notice

Reviews - Written reviews by users who have bought the game.

SupportedLanguages - the languages the game supports

Website - A link to the game's website.

PCMinReqsText - The minimum requirements for the game to run on Windows.

PCRecReqsText - The recommended requirements for the game to run on Windows.

LinuxMinReqsText - The minimum requirements for the game to run on Linux.

LinuxRecReqsText - The recommended requirements for the game to run on Linux.

MacMinReqsText - The minimum requirements for the game to run on MacOS.

MacRecReqsText - The recommended requirements for the game to run on MacOS.

2. Exploratory Data Analysis

1. **Build** a DataFrame from the data (ideally, put all data in this object)
2. **Clean** the DataFrame. It should have the following properties:
    a. Each row describes a single object
    b. Each column describes a property of that object
    c. Columns are numeric whenever appropriate
    d. Columns contain atomic properties that cannot be further decomposed
3. Explore **global properties**. Use histograms, scatter plots, and aggregation functions to summarize the data.
4. Explore **group properties**. Use groupby and small multiples to compare subsets of the data.

Building a data frame from the data

First, I used the matplotlib inline magic function to allow the graphs to display in the workbook. Then I imported the pandas, numpy and matplotlib.pyplot libraries.

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Next, I used pandas read to load the csv into the data frame called df and used the head function to examine the first five rows.

Cleaning the data frame

Examining the data frame

Firstly, I used dtypes to check the data types of each column then using shape and size I checked the shape and length of the data frame.

```
df.dtypes
print(df.shape)
print(df.size)
```

Identifying missing values

```
np.sum(df.isnull())
```

Using the numpy function sum on the data frame to check for empty values.

```
QueryName had 1

SupportEmail had 2

SupportURL had 1

LegalNotice had 1
```

Dealing with missing or nan values

To deal with the missing or Nan values I changed their values to “” as I am not planning to use them for my analysis.

```
df.loc[df.QueryName.isnull(), 'QueryName']=""
df.loc[df.SupportEmail.isnull(), 'SupportEmail']=""
df.loc[df.SupportURL.isnull(), 'SupportURL']=""
df.loc[df.LegalNotice.isnull(), 'LegalNotice']=""
```

Afterwards I checked to see if there were any null values and found there were none.

Dealing with duplicate values

Using the duplicate method, I checked to see if any of the row were duplicated.

```
len(df[df.duplicated()==True])
```

It returned that 53 rows were duplicates.

I then checked visually to see if in fact these rows duplicates.

```
df[df.duplicated()==True].head(53)
```

After that I selected one of the values to confirm that there are duplicates.

```
df.loc[df['QueryID']==80]
```

This returned two rows of identical information therefore I used drop_duplicates with keep set to first to get rid of these duplicate values.

```
df=df.drop_duplicates(keep='first')
```

Afterwards I rechecked for duplicates and confirmed that the original entries were still there.

Changing Columns to appropriate data types

Firist I used the dtypes method to visually inspect each column to see if the column has a numeric data type it's a number.

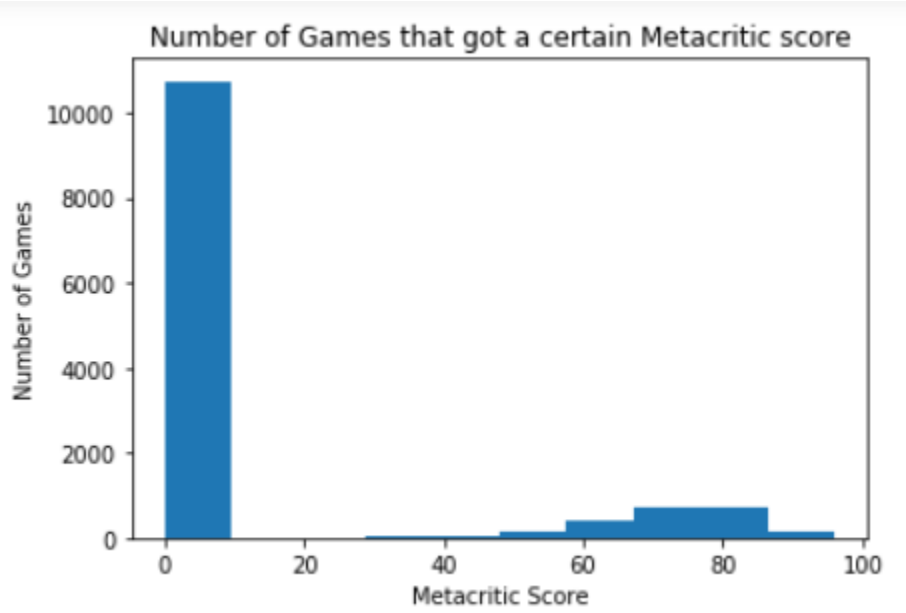
```
df.dtypes
```

None of the relevant columns were in the wrong datatype.

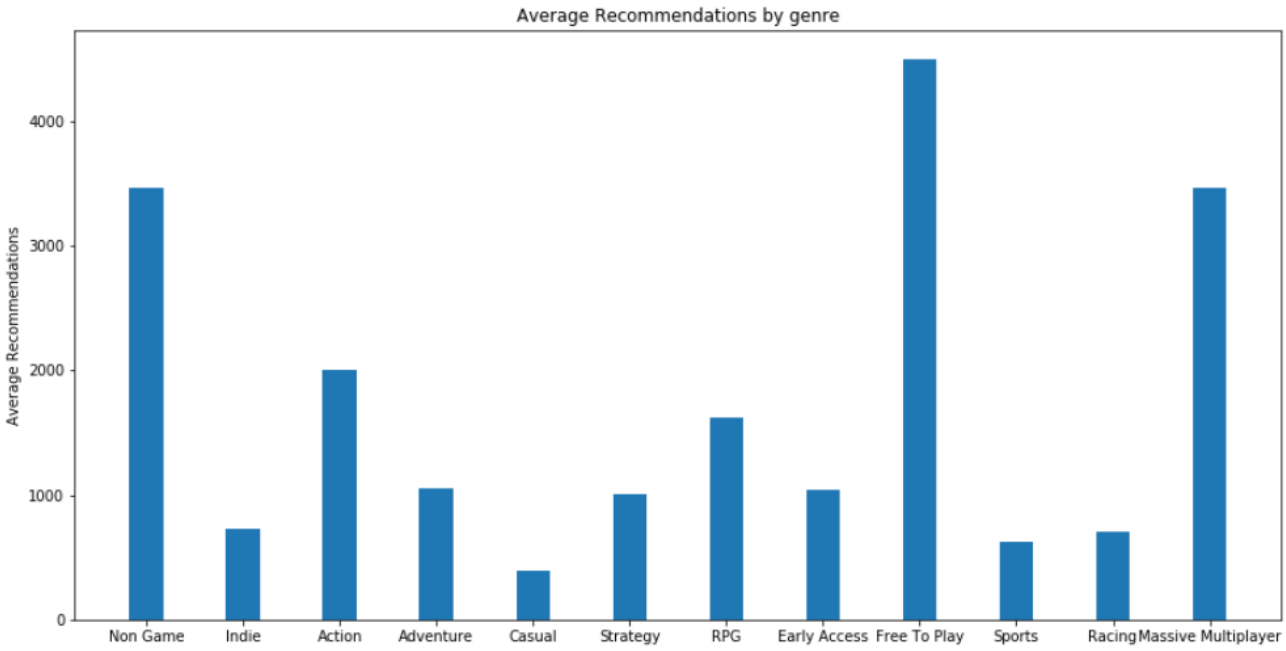
EDA Results

The next step is to get to know the data by using a few "global property" visualizations eg(Histograms and Bar charts etc.) this will help to spot any interesting or strange patterns.

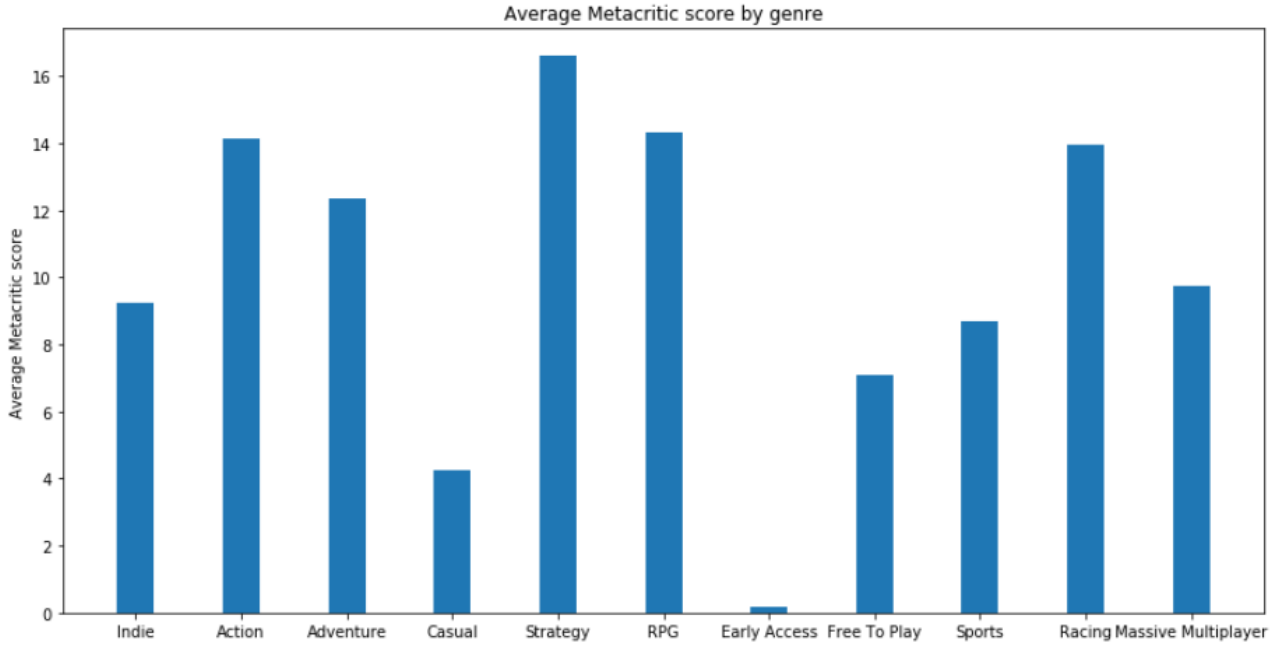
These are the graphs I generated:



Interpretation: Most Games in this dataset receive a Metacritic score of 0 to 12 while little to none have a score in the early 20s.

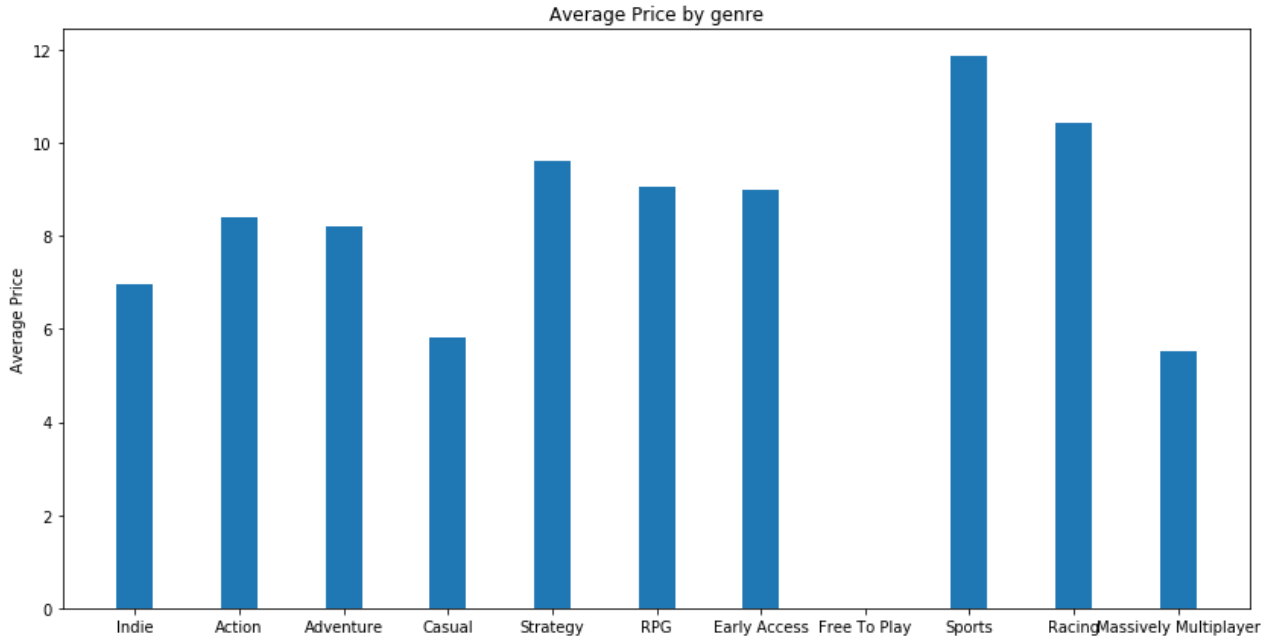


Interpretation: The free to play genre receives on average the most recommendation's this however is expected as these games will have a larger exposure to the community due to having no cost to play.

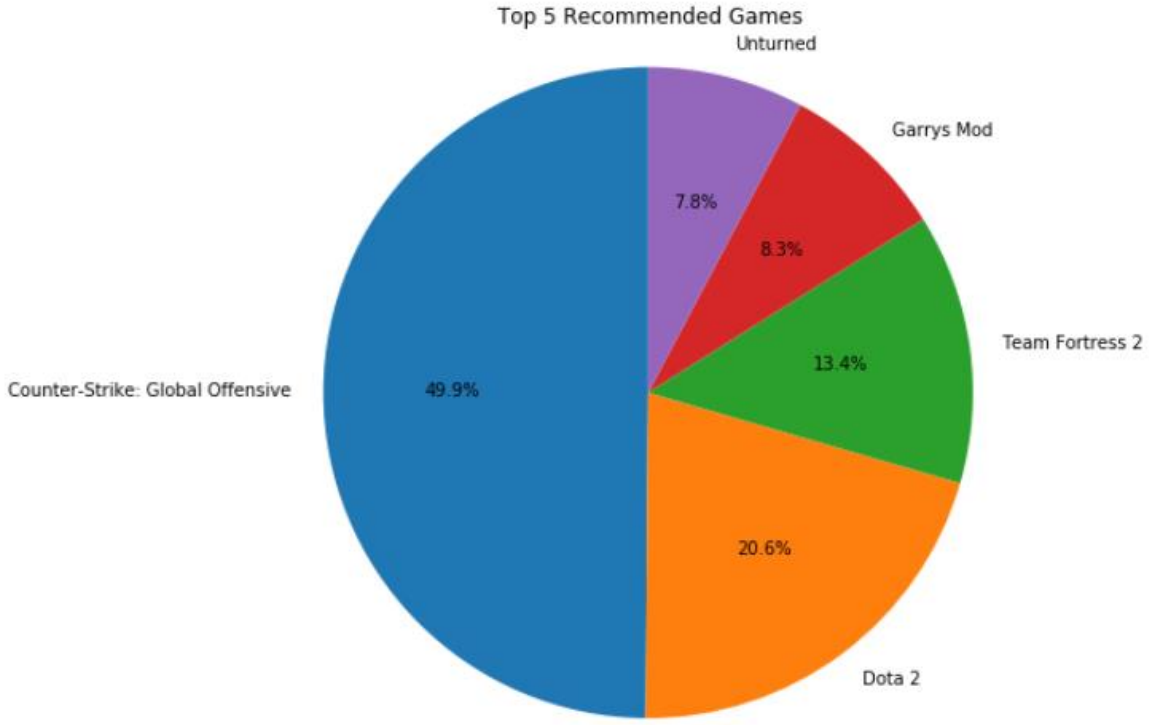


Note: I removed from this analysis as the non-game genre does not get a Metacritic score.

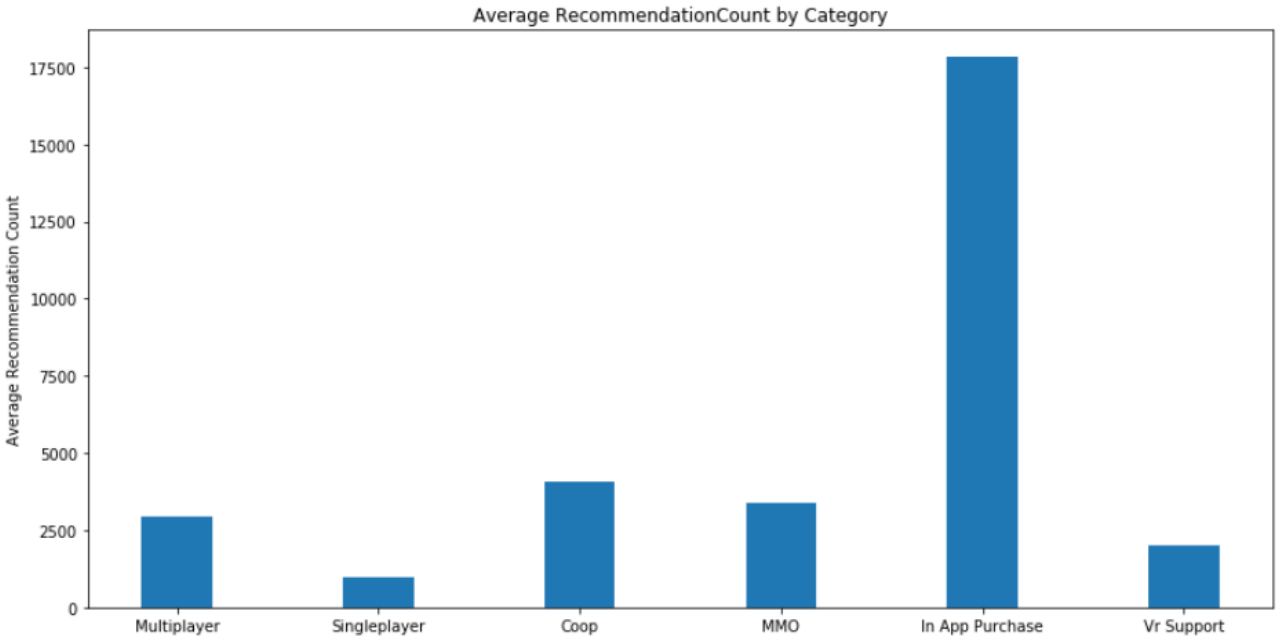
Interpretation: The strategy genre has the highest average Metacritic score at 16.5 while early access is bar far the lowest with a mean of .15.



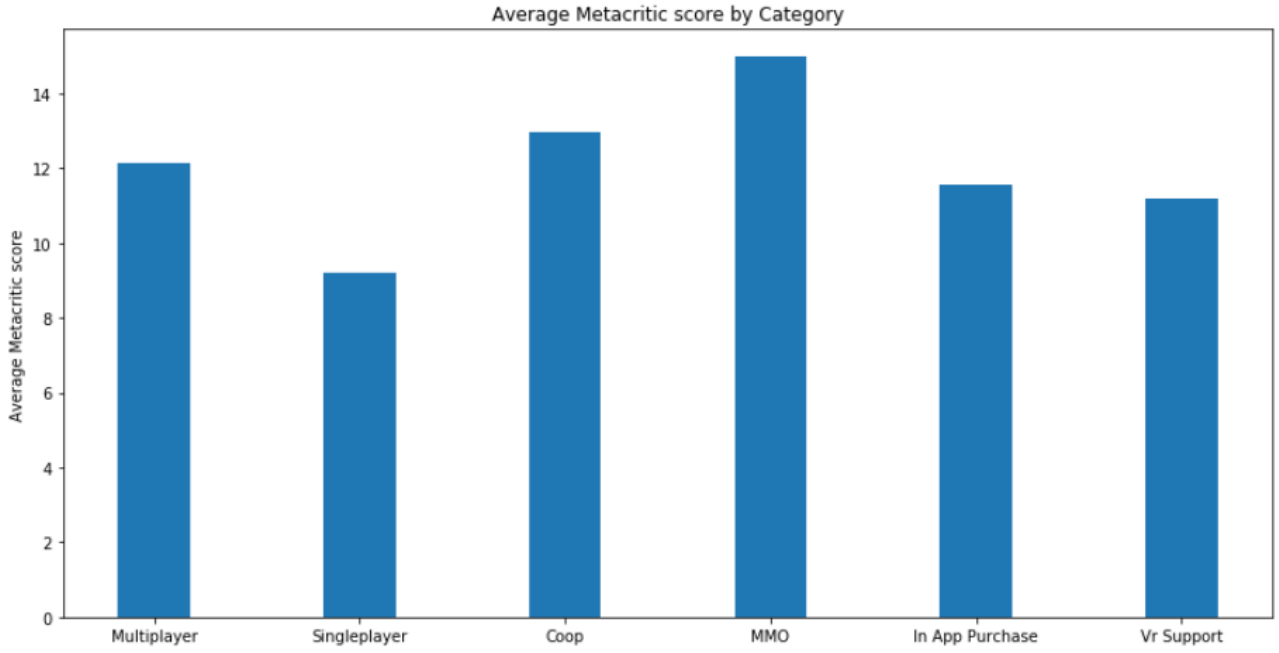
Interpretation: Games belonging to the sports genre on average the most expensive, yet they do not have a large average Metacritic score or user recommendations.



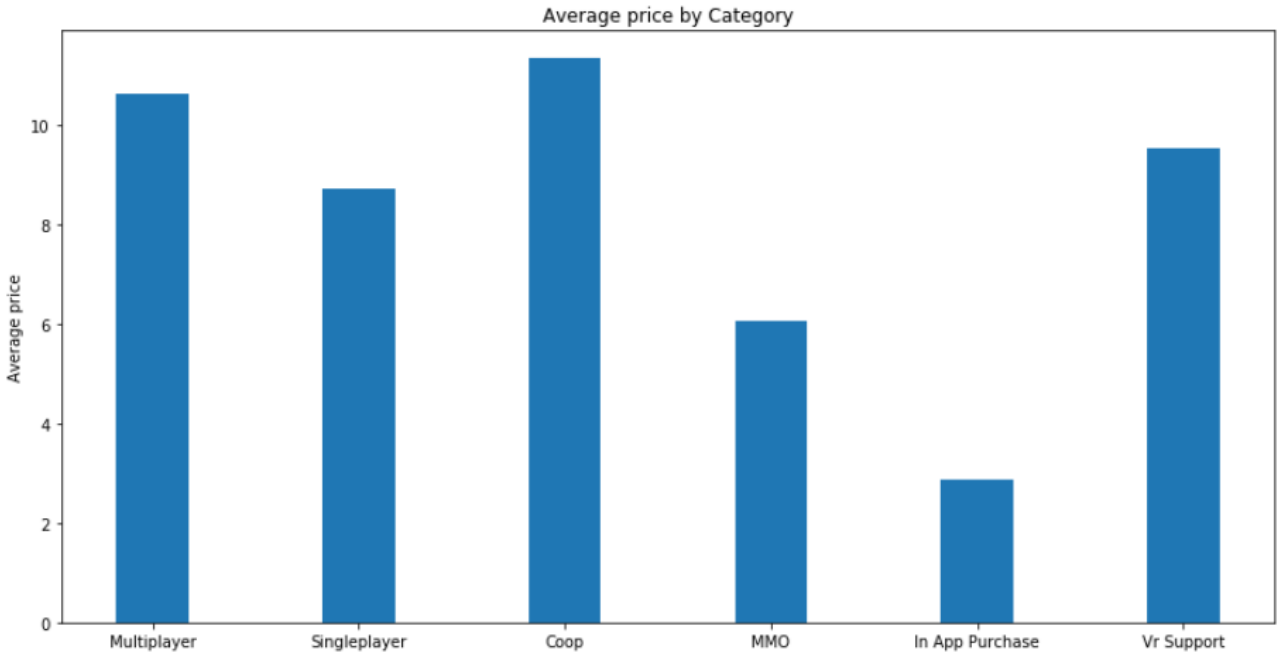
Interpretation: Counter-Strike: Global Offensive has not only the most recommendations but it almost has as many as the other four put together.



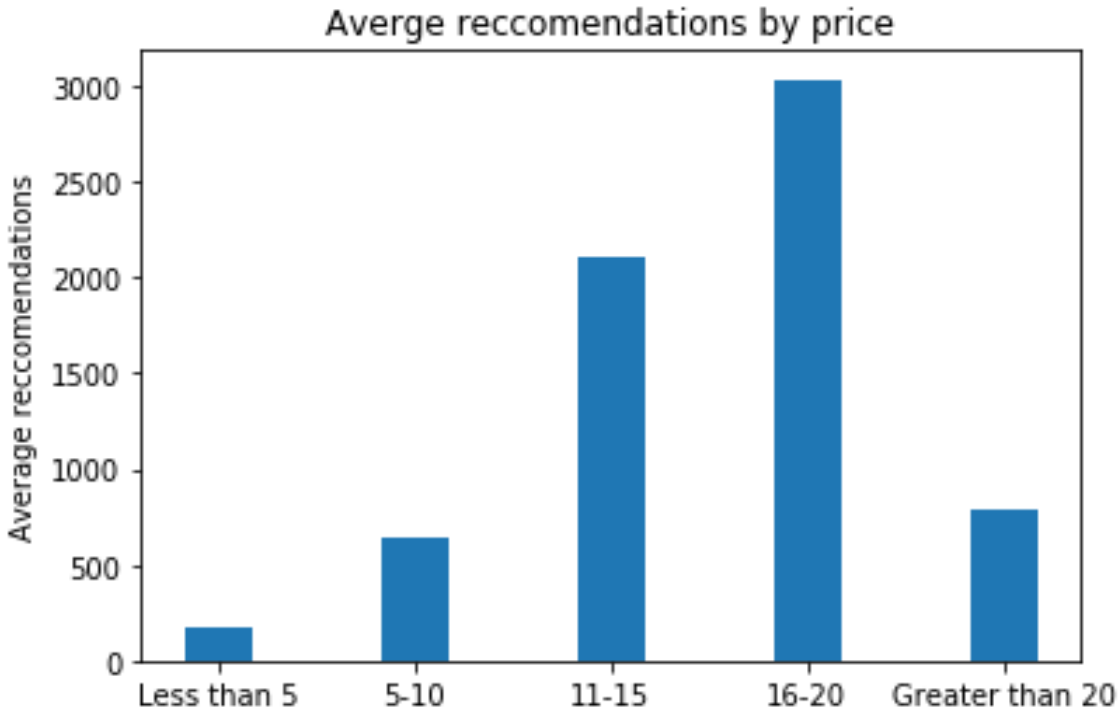
Interpretation: In app purchases is by far has on average the most recommendations this although unexpected makes sense in context of the previous pie chart as four of the top five most recommended (all except Garrys Mod) are in the in-app purchases category.



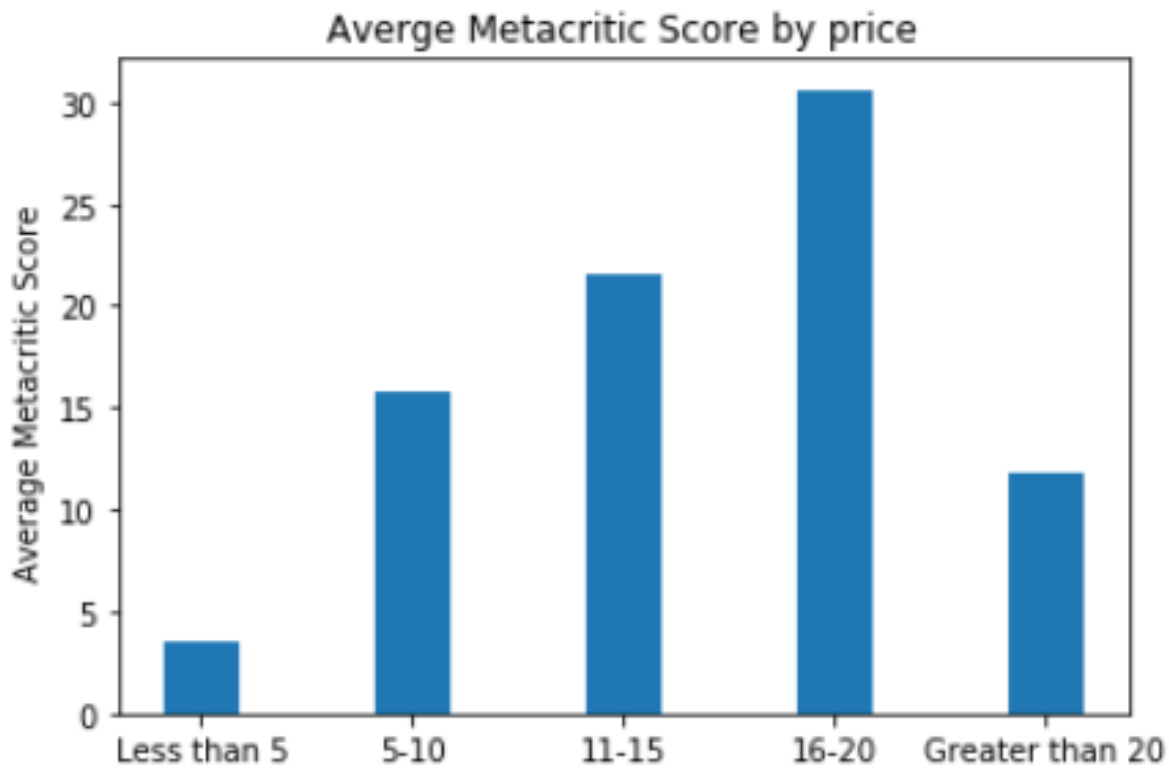
Interpretation: The Category with the most recommendations is the mmo Category.



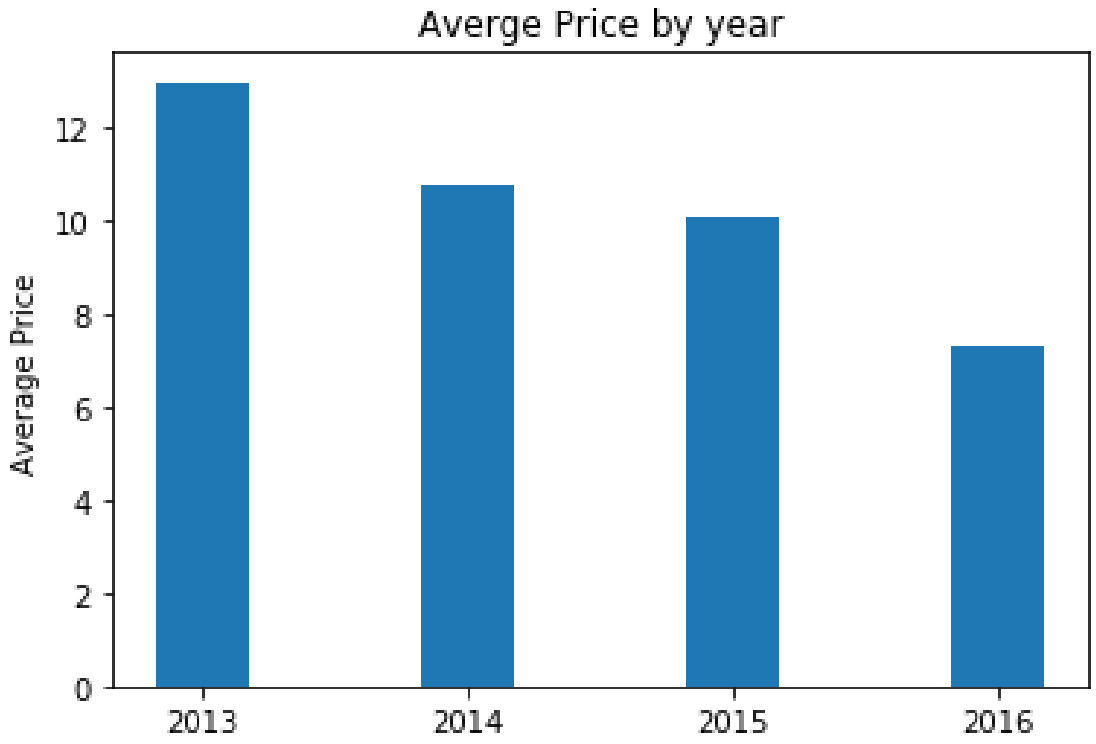
Interpretation: Coop has the highest average price yet the second highest average Metacritic score and Recommendations Count. While In app purchase has the lowest average price fourth highest average Metacritic score and the highest average recommendations. The visuals don't seem to suggest that price, Metacritic score and Recommendations are connected.



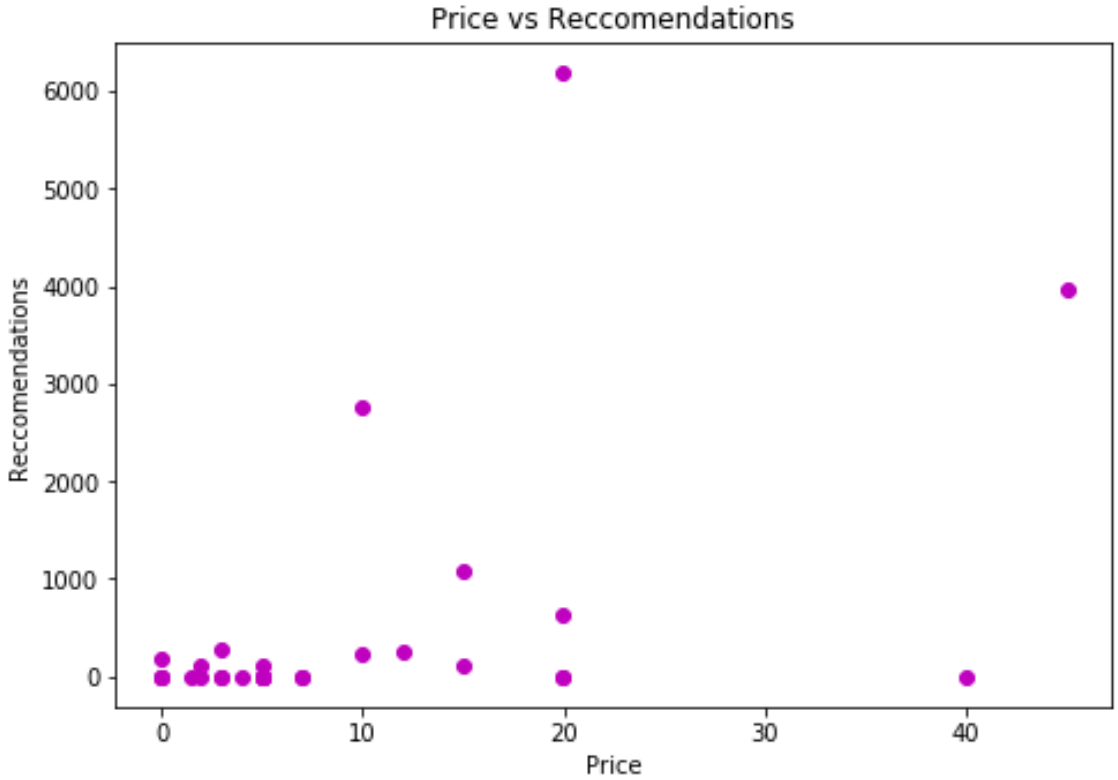
Interpretation: The average most recommended price is between 16-20. The visual almost suggests that when the price increases the recommendation's increase as well until the price exceeds twenty.



Interpretation: Similarly, to the previous visual it suggests that the Metacritic score will increase with price until the price exceeds twenty.

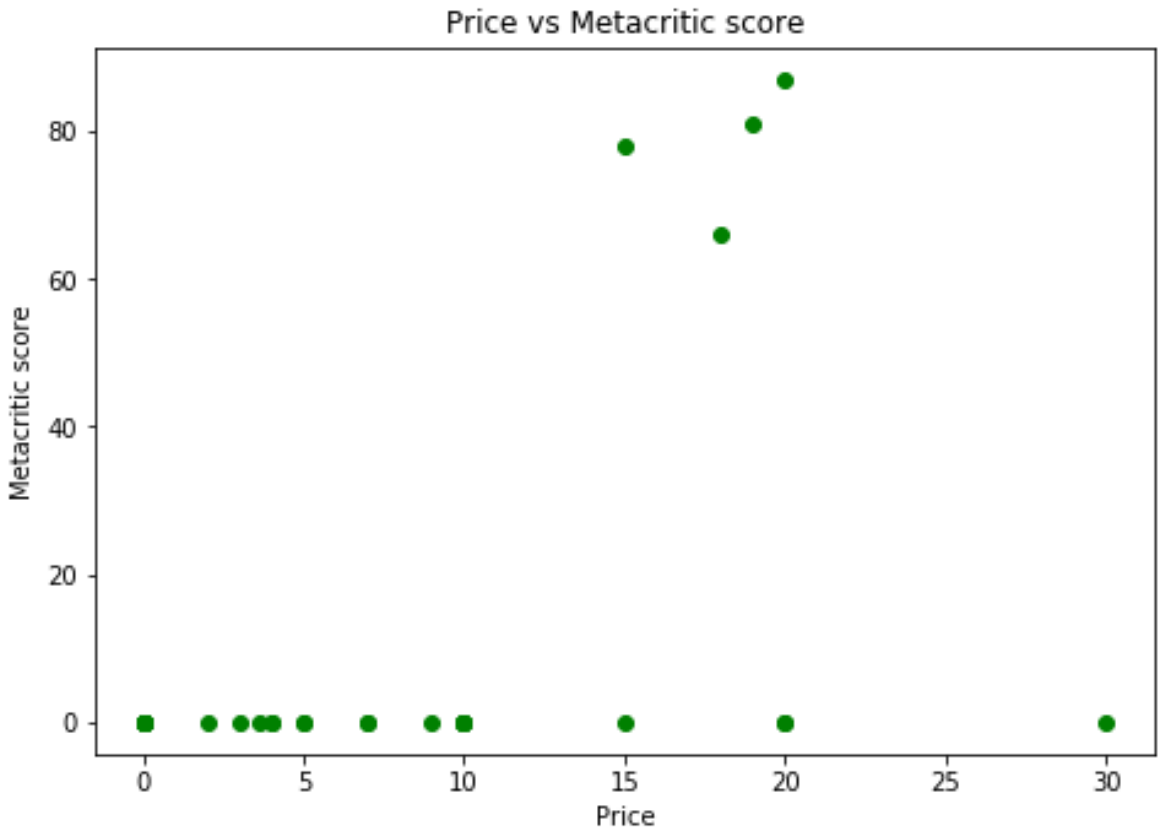


Interpretation: The average price of steam games appears to be decreasing every year.



Note: This scatter plot is made up of a random sample of 30 Games where the Game is not free.

Interpretation: The visual seems to show a weak positive correlation between the price of the game and its price.



Note: This scatter plot is made up of a random sample of 30 Games where the Game is not free.

Interpretation: Again, similarly this visual also suggests a positive correlation between the games price and it's Metacritic score.

Challenges

While I was plotting the Metacritic scores across the various visuals, I noticed a large amount of them were zero I then realized that they were zero not because they got that score is was because they had not been rated by Metacritic. I eventually decided to leave these values as they were as if a game hadn't gotten enough attention.

While plotting some of the prices I noticed that some of the games that were labeled as free still had a price, so I looked up some of these games in the steam store and found that in fact that were not free. I decided to change there IsFree column to false as the data was inaccurate.

```
df.loc[(df.PriceFinal>0) & (df.IsFree),'IsFree']=False
df.loc[(df.PriceFinal>0) & (df.GenreIsFreeToPlay),'GenreIsFreeToPlay']=False
```

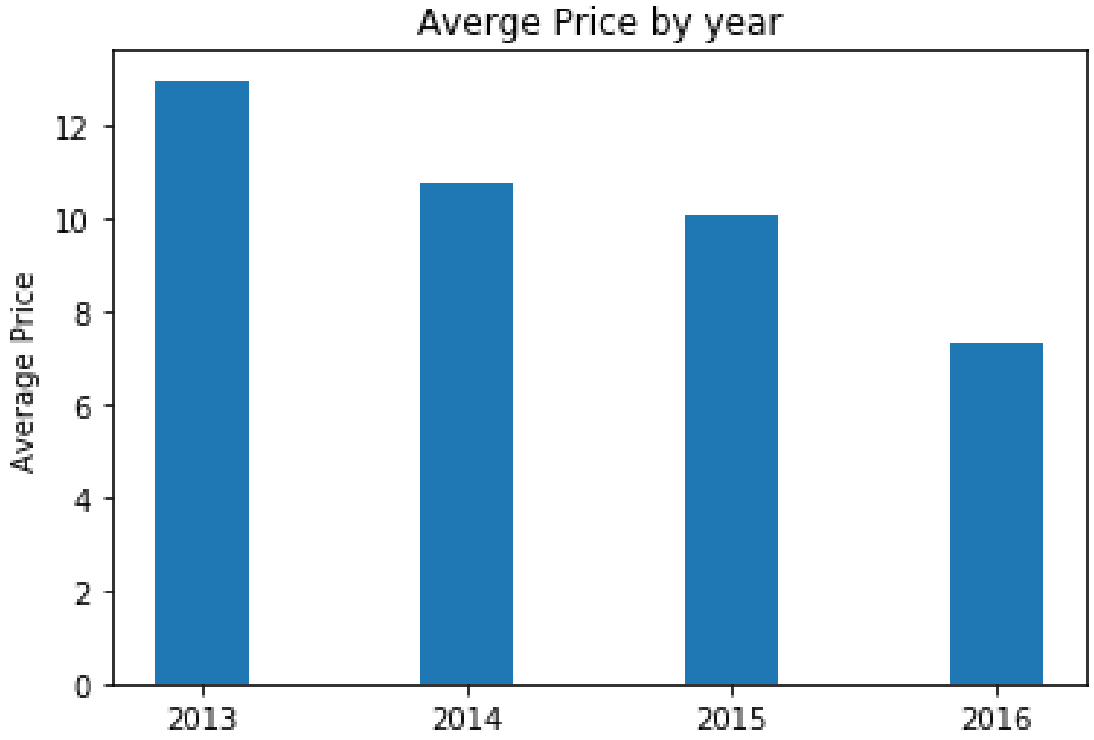
3.Questions

The questions I aim to test are

1. Does the data suggest that the average price of steam games is dropping every year?
2. Does the data suggest that the in-app purchases category has on average the highest recommendations count?

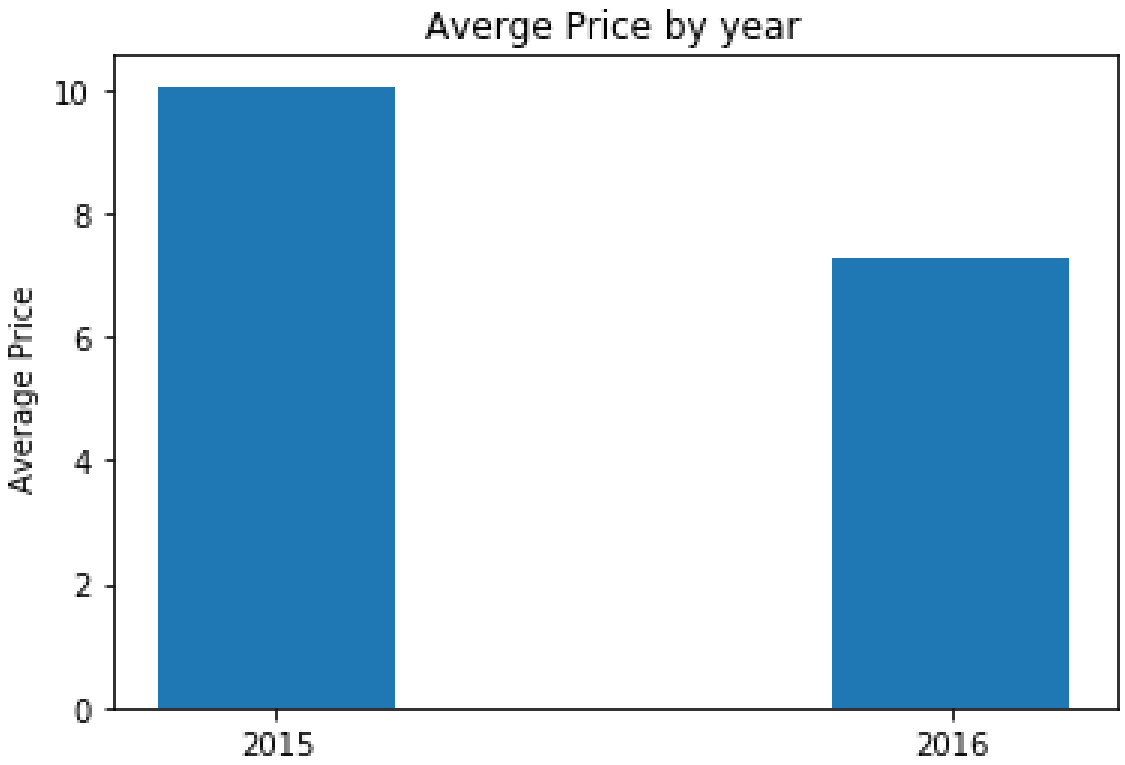
Does the data suggest that the average price of steam games is dropping every year?

This question comes to me as the visuals show a decrease is average price in the years 2013-2016.



Interpretation: The average price of steam games appears to be decreasing every year.

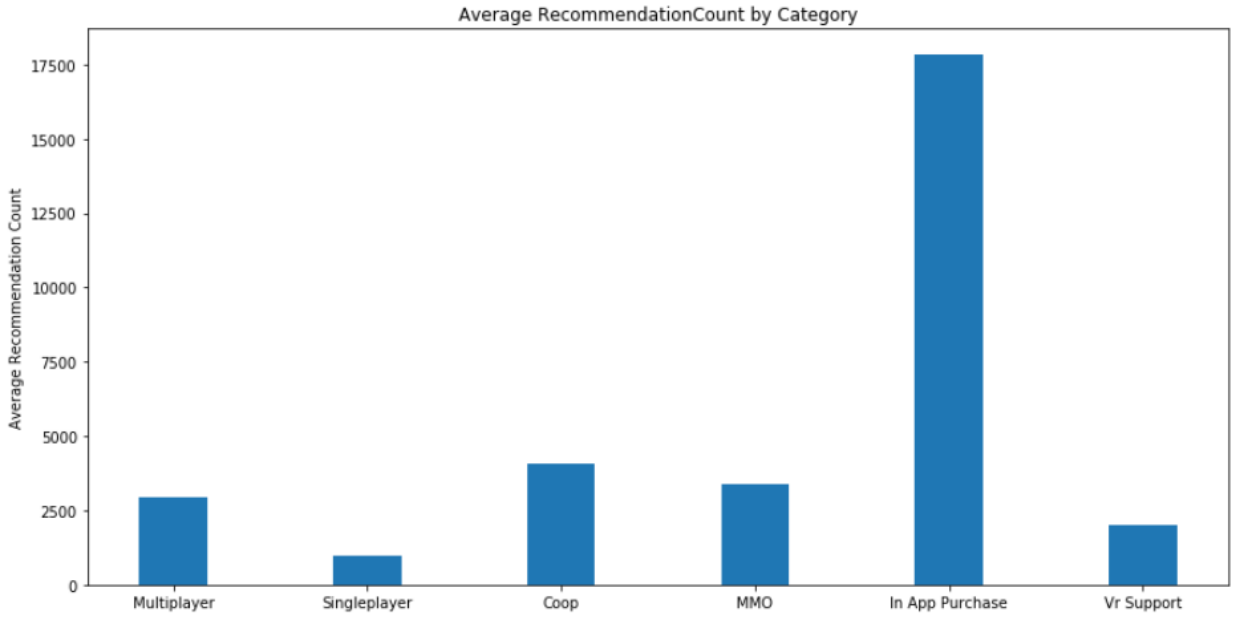
For the analysis I decided to narrow down the scope of the analysis to just the years 2015 and 2016 as these had the largest amount of games in the dataset.



Interpretation: The average price of steam games is higher in 2015 than in 2016 but is the amount statistically significant.

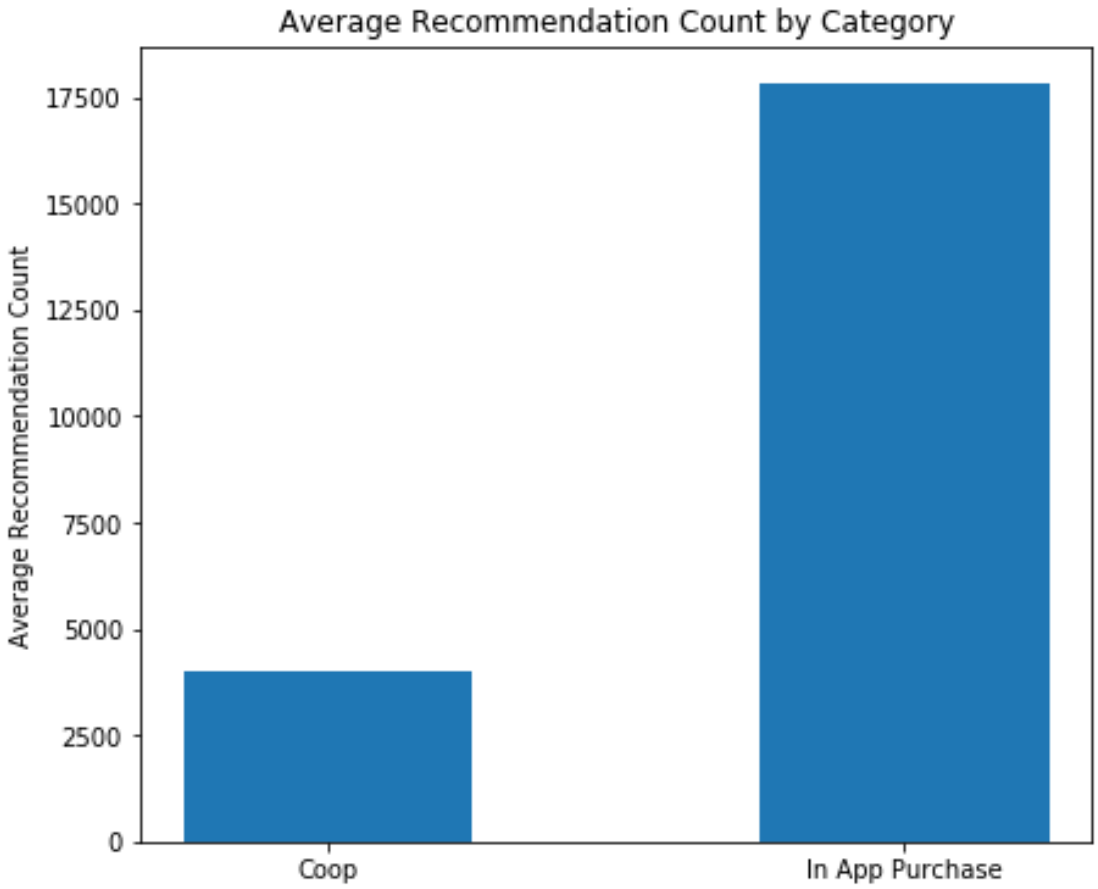
Does the data suggest that the in-app purchases category has on average the highest recommendations count?

The reason I chose this question was many people online and, in the media, often ridicule games with in-app purchases and microtransactions as being the worst kind of games yet this visual shows that the users are on average recommending games of this category more than the second highest.



Interpretation: In app purchases is by far has on average the most recommendations this although unexpected makes sense in context of the previous pie chart as four of the top five most recommended (all except Garrys Mod) are in the in-app purchases category.

For the analysis I focused on In-app purchases and the second top Coop.



Interpretation: The in-app purchases category has an average over four times higher than the coop category.

4.Analysis results

Does the data suggest that the in-app purchases category has on average the highest recommendations count?

Hypothesis Test

H0: there is no difference in mean price in 2015 and 2016.

H1: The mean prices have gone down.

H0: μ Prices2015 = μ2 Prices2016

HA: μ Prices2015 > μ2 Prices2016

One Tailed test

N=200

DF=200-2

T-critical = 1.653

Alpha = 0.05

Decision Rule: Reject H0 if t observed > t-critical

```
df.loc[df.ReleaseDate==2015,'PriceFinal'].sample(n=100, random_state=10)
df.loc[df.ReleaseDate==2016,'PriceFinal'].sample(n=100, random_state=10)

# tCrit is 1.653
# since scipy ind test is a two tailed test p value is divided by 2
t,p2tailed=stats.ttest_ind(value1,value2)
print(t,p2tailed/2)
```

Results: t statistic = 1.6185950911303215

Interpretation: Since the calculated t-statistic is less than the t-critical value meaning there is not a significant enough difference between them. Therefore, we do not reject H0 at a 95% confidence level.

Does the data suggest that the in-app purchases category has on average the highest recommendations count?

Hypothesis Testing

H0: there is no difference in mean recommendations between in app purchases and Coop games.

H1: The mean prices have gone down.

H0: μ in-app recommendations = μ2 coop recommendations

HA: μ in-app recommendations > μ2 recommendations

One Tailed test

N=200

DF=200-2

T-critical = 1.653

Alpha = 0.05

Decision Rule: Reject H0 if t observed > t-critical

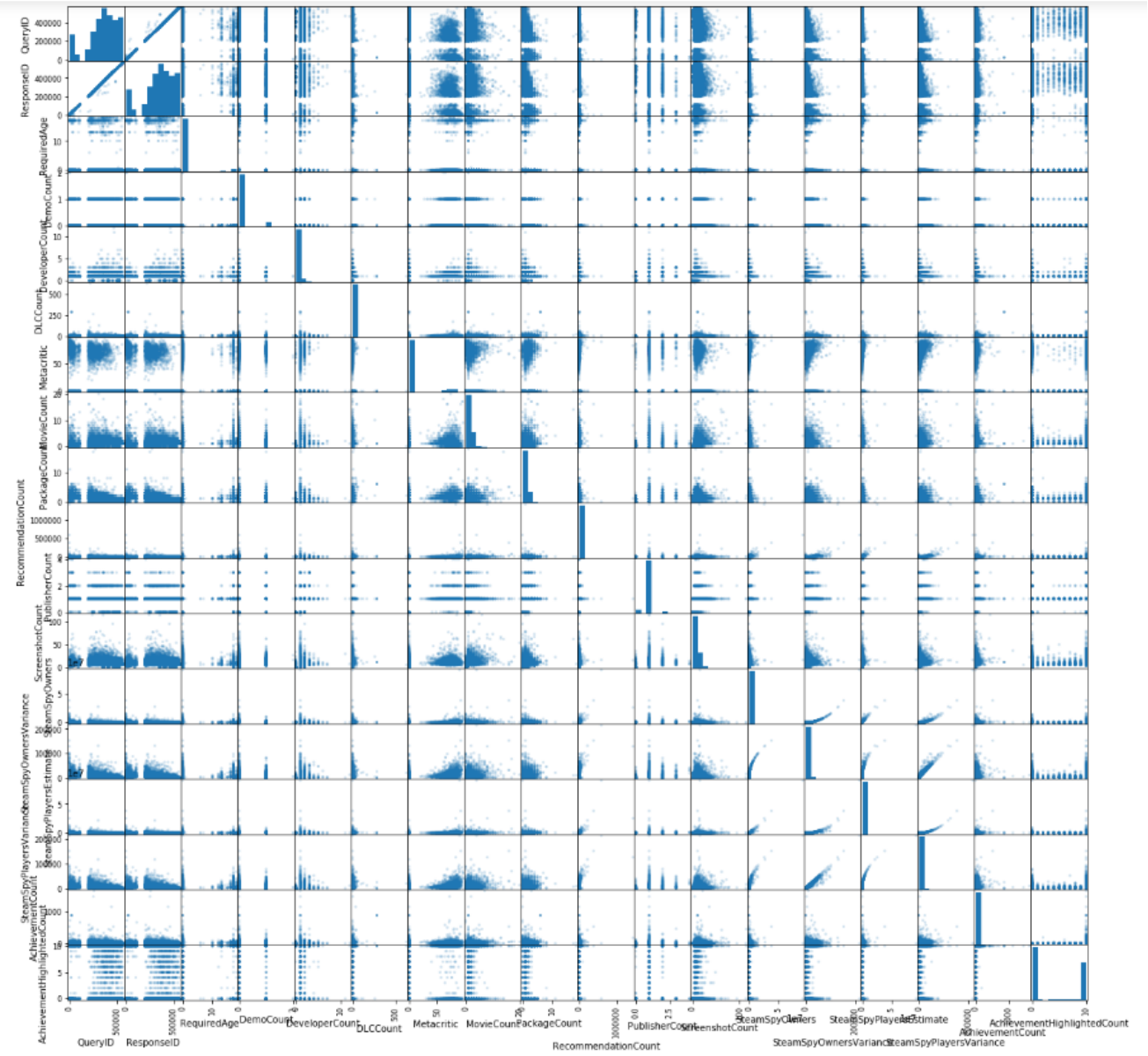
```
coop=df.loc[df.CategoryCoop ==True,'RecommendationCount'].sample(n=100, random_state=1)
inapp=df.loc[df.CategoryInAppPurchase ==True,'RecommendationCount'].sample(n=100, random_state=1)

t,p2tailed=stats.ttest_ind(inapp,coop)
print(t,p2tailed/2)
```

Result: t statistic = 1.466141448855674

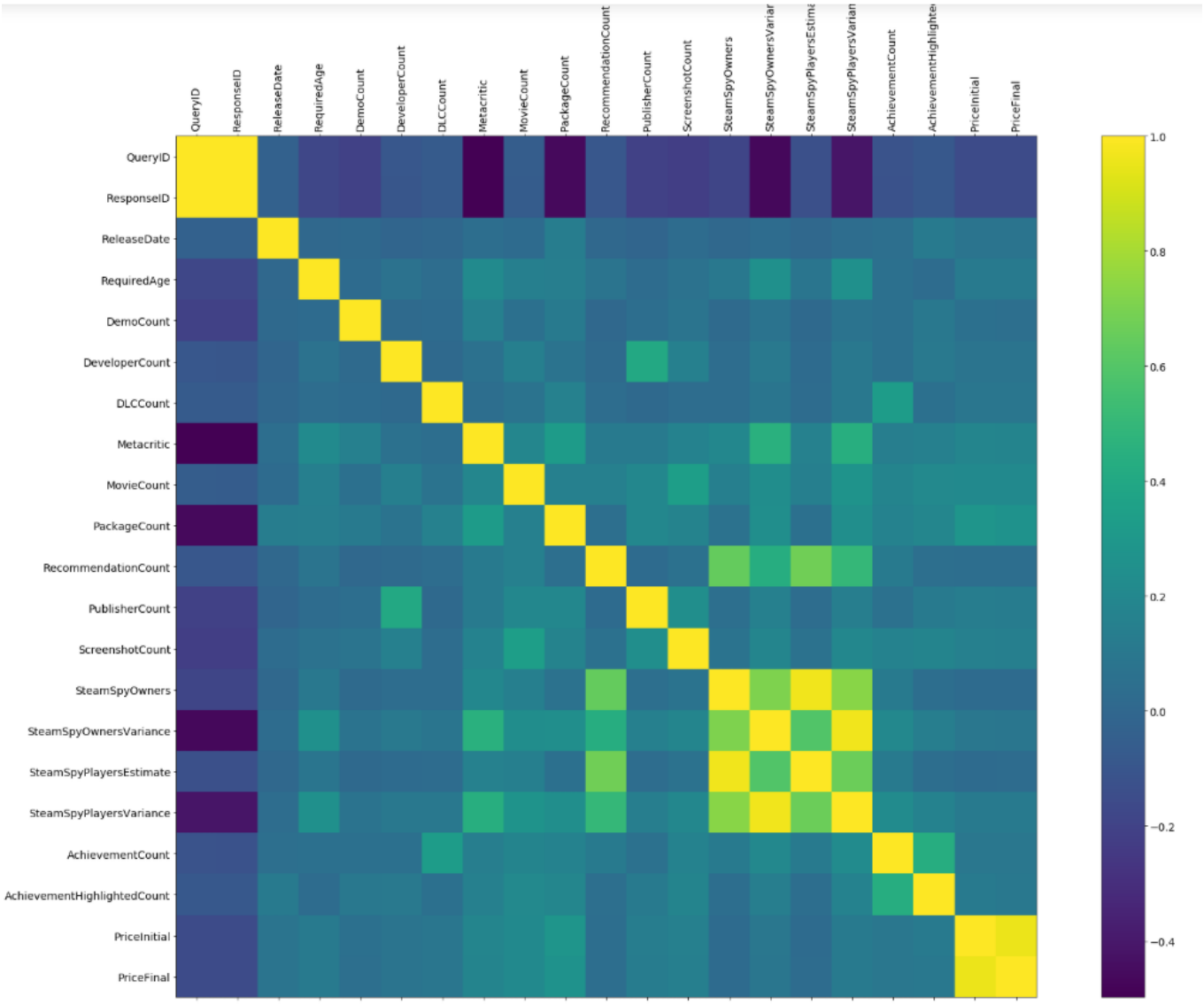
Interpretation: Similarly, to above the calculated t statistic is less than the t-critical value therefore we conclude that we cannot reject H0 at a 95% confidence level. The interesting part of this is that the visual clearly shows that the in-app category is by far higher than the coop the reason the t statistic was lower I believe was because of the lack of all the outliers thus leaving a more accurate statistic.

Correlations

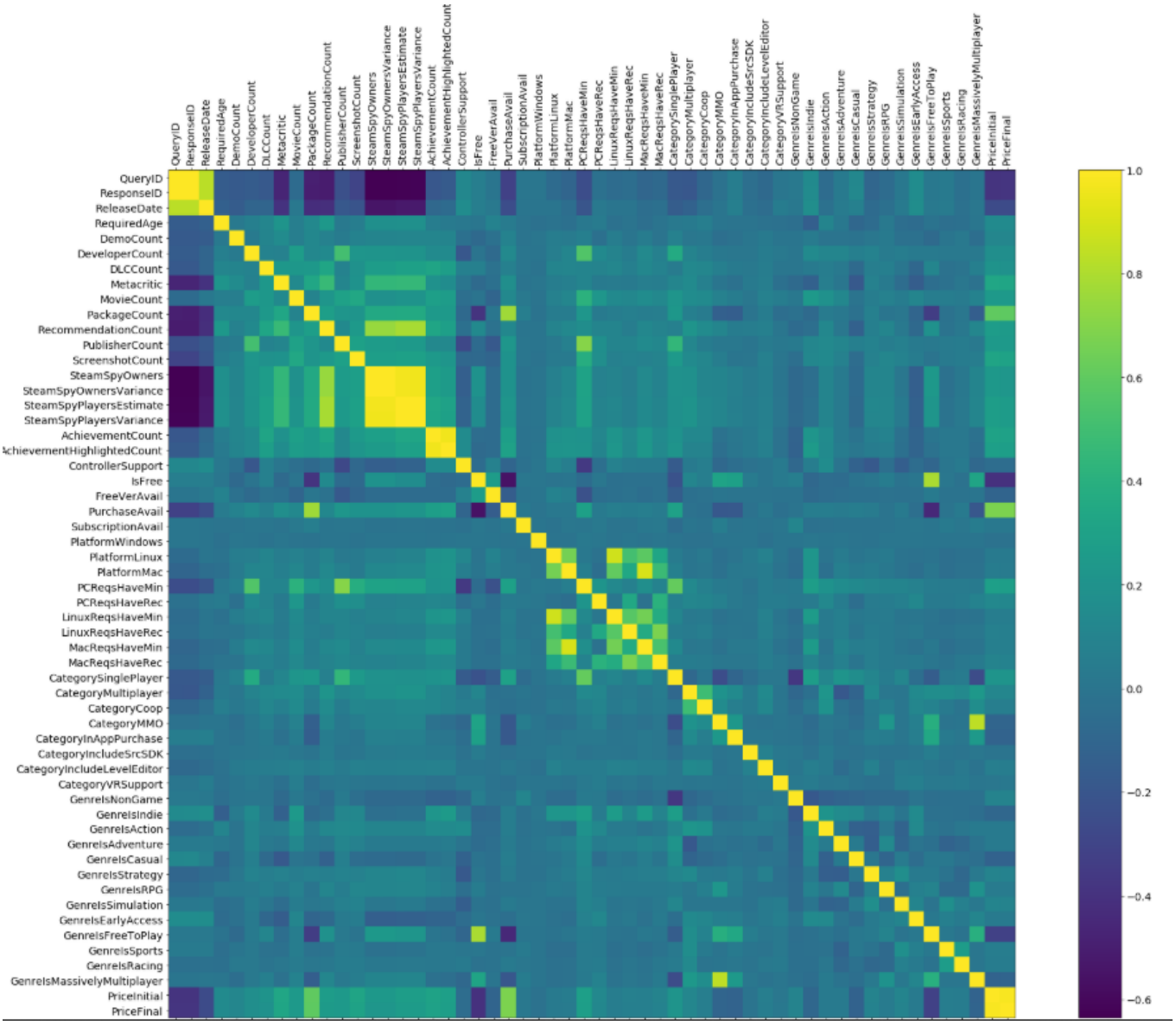
Using pandas scatter_matrix function I created this visual below to help me to spot any interesting corraltions.



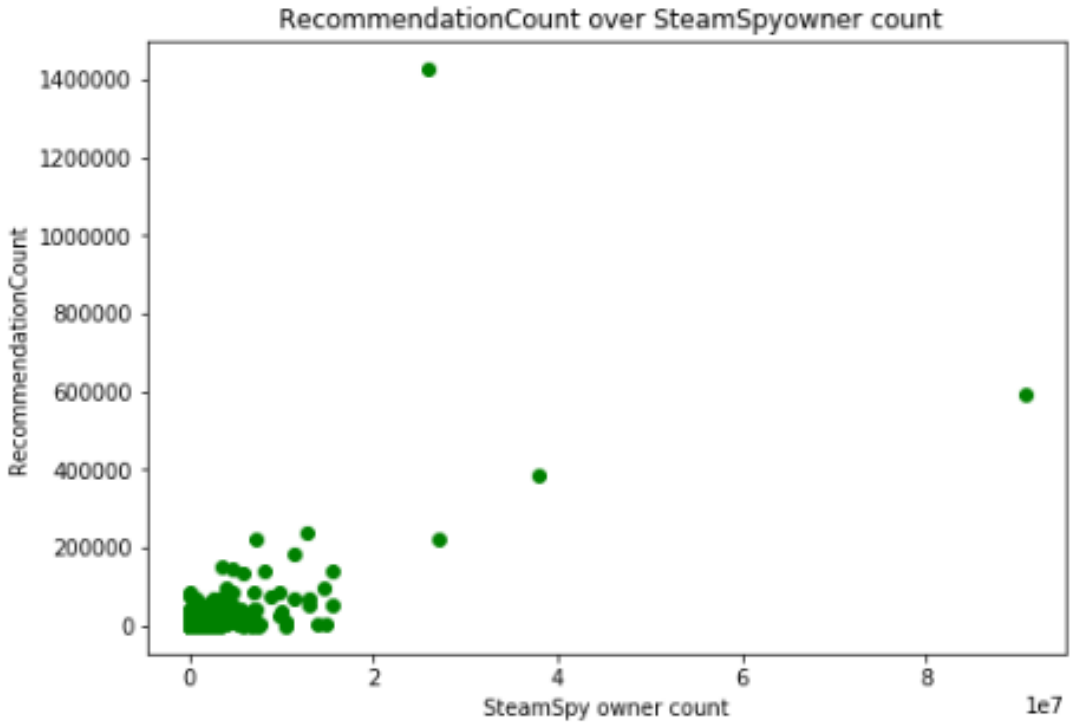
I also used pyplot's matshow to give an easier to read visual. For the first matshow I used the Pearsons coefficient I also removed all bool and object columns as Pearsons is better used for direct corralation when one rises so does the other etc.



For the second one I used Spearman’s coefficient as Spearman's is able to assess non liner relationships, I added back in the bool typed columns in the below visual.


QueryID
ResponseID
ReleaseDate
RequiredAge
DemoCount
DeveloperCount
DLCCount
Metacritic
MovieCount
PackageCount
RecommendationCount
PublisherCount
ScreenshotCount
SteamSpyOwners
SteamSpyOwnersVariance
SteamSpyPlayersEstimate
SteamSpyPlayersVariance
AchievementCount
AchievementHighlightedCount
ControllerSupport
IsFree
FreeVerAvail
PurchaseAvail
SubscriptionAvail
PlatformWindows
PlatformLinux
PlatformMac
PCReqsHaveMin
PCReqsHaveRec
LinuxReqsHaveMin
LinuxReqsHaveRec
MacReqsHaveMin
MacReqsHaveRec
CategorySinglePlayer
CategoryMultiplayer
CategoryCoop
CategoryMMO
CategoryInAppPurchase
CategoryIncludeSrcSDK
CategoryIncludeLevelEditor
CategoryVRSupport
GenreIsNonGame
GenreIsIndie
GenreIsAction
GenreIsAdventure
GenreIsCasual
GenreIsStrategy
GenreIsRPG
GenreIsSimulation
GenreIsEarlyAccess
GenreIsFreeToPlay
GenreIsSports
GenreIsRacing
GenreIsMassivelyMultiplayer
PriceInitial
PriceFinal
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

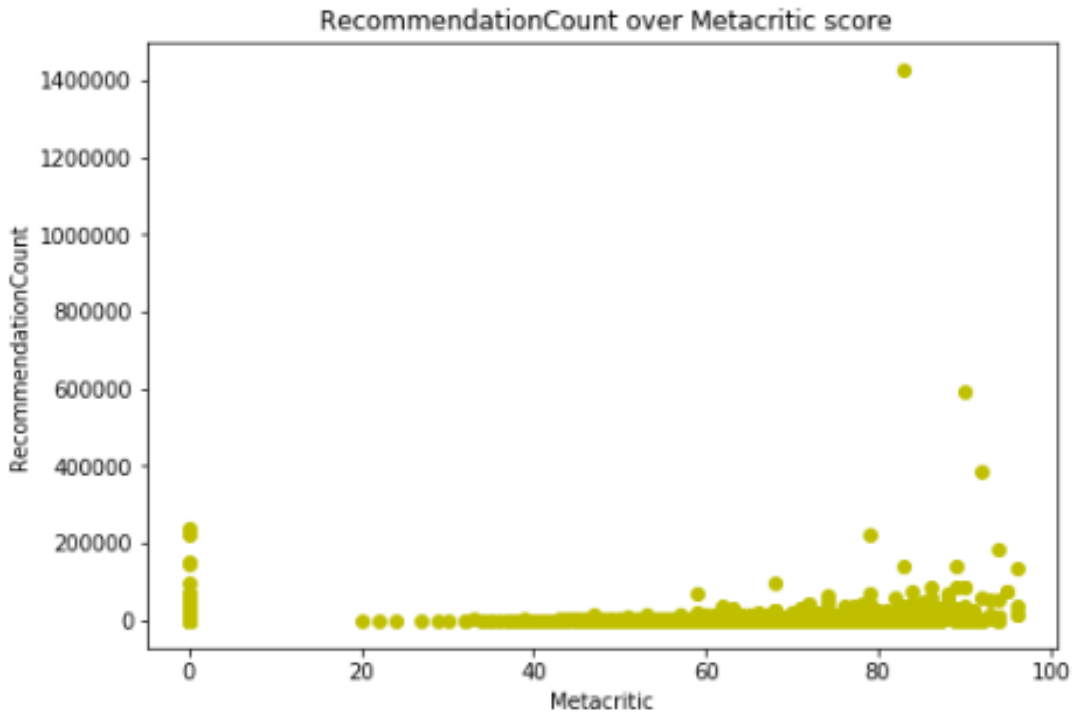
Is there a correlation between a games recommendation's count and steam spy owners?



Interpretation: The visual above shows appears to show a positive correlation between the number of steam spy owners and the number of recommendations a game has.

Results: The Pearson correlation coefficient is .6435988422112349 and the Spearman correlation coefficient.7501654776303415. Therefore, indicating a positive correlation i.e. the more recommendations a game has the more steam spy owners it will have.

Is there a correlation between a games recommendation's count and its Metacritic score?



Interpretation: The visual above shows appears to show little to no correlation between the Metacritic score and the number of recommendations a game has.

Results: The Pearson correlation coefficient is . 1181198621668242 and the Spearman correlation coefficient .4657167962074023. Therefore, indicating a weak to non-existent correlation.

Is there a correlation between a games recommendation's count and its price?

Interpretation: Similarly, to the visual above the visual appears to show a very weak correlation between the price of a game and its number of recommendations. Which is the opposite of what I would have expected.

Results: The Pearson correlation coefficient is .04378413637521766 and the Spearman correlation coefficient .2568203456038146. Therefore, indicating a weak to non-existent correlation meaning the games recommendations is not related to the price.

Regression and Prediction

As time goes on steam games will continue to receive recommendations and therefore their Steam Spy owner count will also rise. So, I used regression analysis to predict how many Steam Spy owners a game would have if it had one million recommendations.

```
x = np.array(df['RecommendationCount'])
y = np.array(df['SteamSpyOwners'])
slope, intercept, r_value, p_value, slope_std_error = stats.linregress(x, y)
prediction = intercept + slope * 1000000
```

The result was 47609086 meaning that if a game got a million recommendations is would have 47609086 steam spy owners.

5.Conclusion

Overall, I am happy with the results of my analysis of this dataset although I think I did use the optimal method of displaying the data or the most optimal way to analyze the dataset I still found a few things that surprised me such as price being a small factor in terms of a games recommendations.

In terms of code reusability, I think most of my code would need small to major refactoring in order to be used in another analysis since the most of it was written around the way the dataset was.