ENF1170: INTRODUCTORY COMPUTER SCIENCES 2023 SPRING
PROJECT OF GROUP 8

2	17023027	ÇAĞRI	KARSLI	8	ÇAĞRI KARSLI
7	20022070	FATMA ZEHRA	USLU	8	
15	21023020	ÖZGÜR	TAĞIL	8	
38	21061503	MERYEM	EKİNCİ	8	
45	22023605	ENES	BOZACI	8	
52	22023631	HANİFE	KÖSE	8	

We decided to analyze this data set. Specially we used these two files named "steam.csv" and "steamspy_tag_data.csv"

First we came up with two hypothesis reading through the data. Following is what we wanted to see if they are actually true:

Hypothesis 1: Most played genre on average would likely be multiplayer games because they are ongoing games and playing with others means it's never the same gameplay ever thus increasing the chances of an average person playing them more than they would games from different genres.

Hypothesis 2: Average positive ratings would increase with the said games being available on more than one platform. If this is true it might indicate when a good games popularity increases developers port them to other platforms.

Now to prove these hypotheses we used data exploratory visualization techniques. Let's look at the code we have written.
(Note: you will need the libraries in the provided "requirements.txt" file to execute these codes. They should be imported using import command.)

Grouping the data by genre and calculating the sum of average playtime:

```
most_played_by_genre = steam_data.groupby('genres')['average_playtime'].mean()
```

Filtering the genres with average playtime above 50 hours (this is done because in the dataset there are likely way too many games with for example only 0-10 hours of gameplay) :

```
most_played_by_genre = most_played_by_genre[most_played_by_genre > 50]
```

And let's sort the genres based on total playtime in descending order:

```
most_played_by_genre = most_played_by_genre.sort_values(ascending=False)
```

display the most played games by genre:

```
print(most_played_by_genre)
```

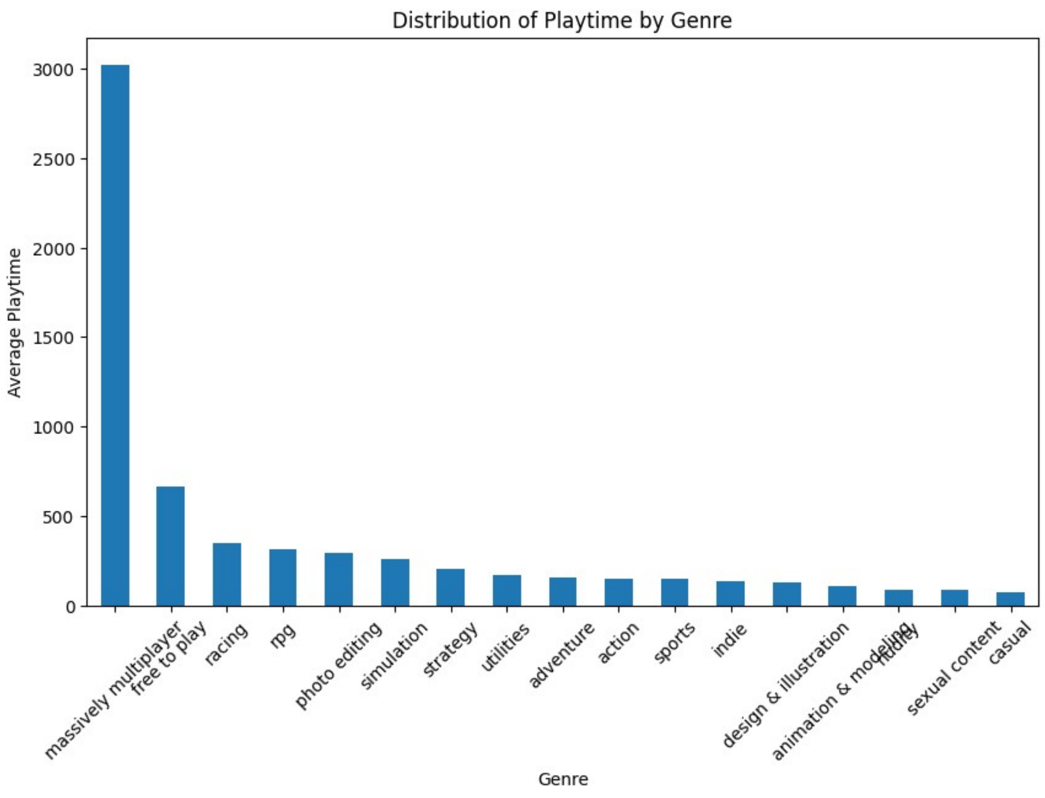
Then we've visualized this information:

a bar chart:

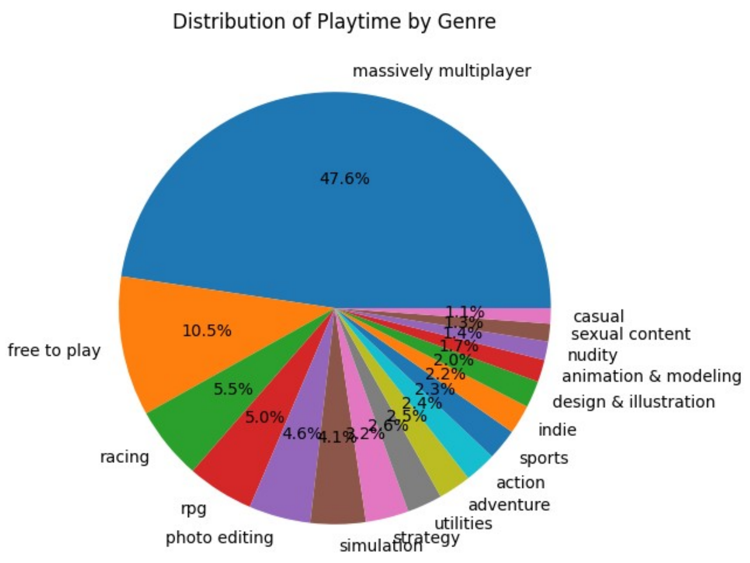
```
plt.figure(figsize=(10, 6))
most_played_by_genre.plot(kind='bar')

plt.xlabel('Genre')
plt.ylabel('Average Playtime')
plt.title('Distribution of Playtime by Genre')
plt.xticks(rotation=45)
plt.show()
```

Which gave us this chart:



We added an additional pie chart because pie charts are cool!



This proved our first hypothesis which was multiplayer genre was played more on the average. Of course why that is , while based on real life experiences we thought it’s because of the social aspect of these games and always changing, is not something that this dataset could prove.

Next we tried to prove our second hypothesis.

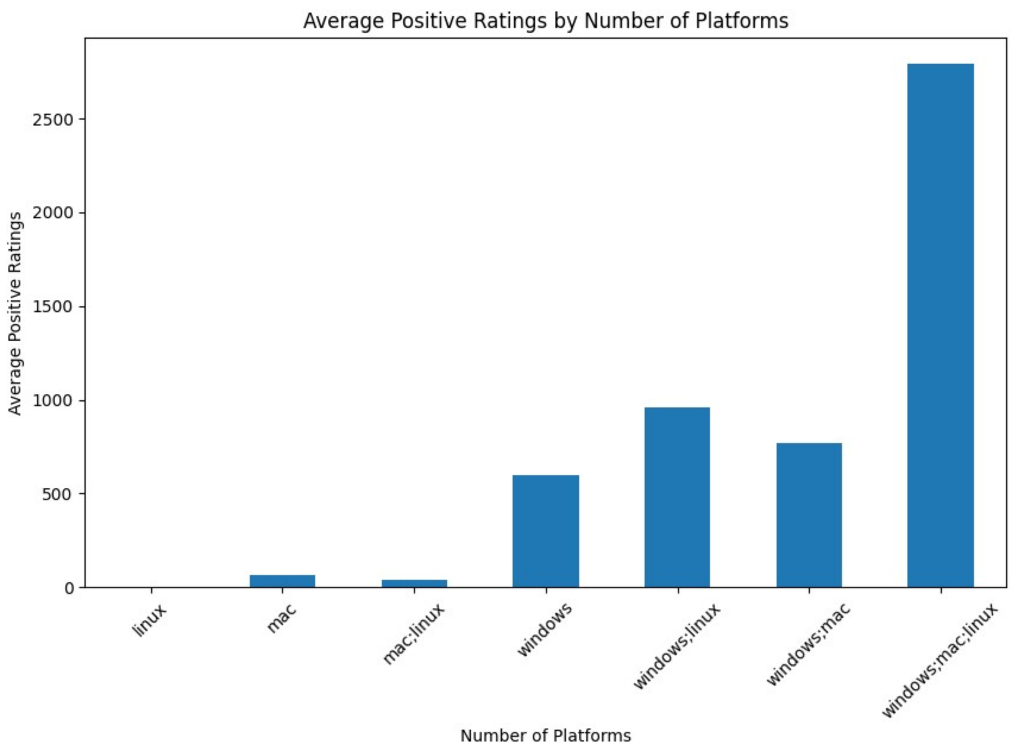
Grouping the games by number of platforms and calculating the mean positive ratings

```
average_positive_ratings_by_platforms = steam_data.groupby('platforms')
['positive_ratings'].mean()
```

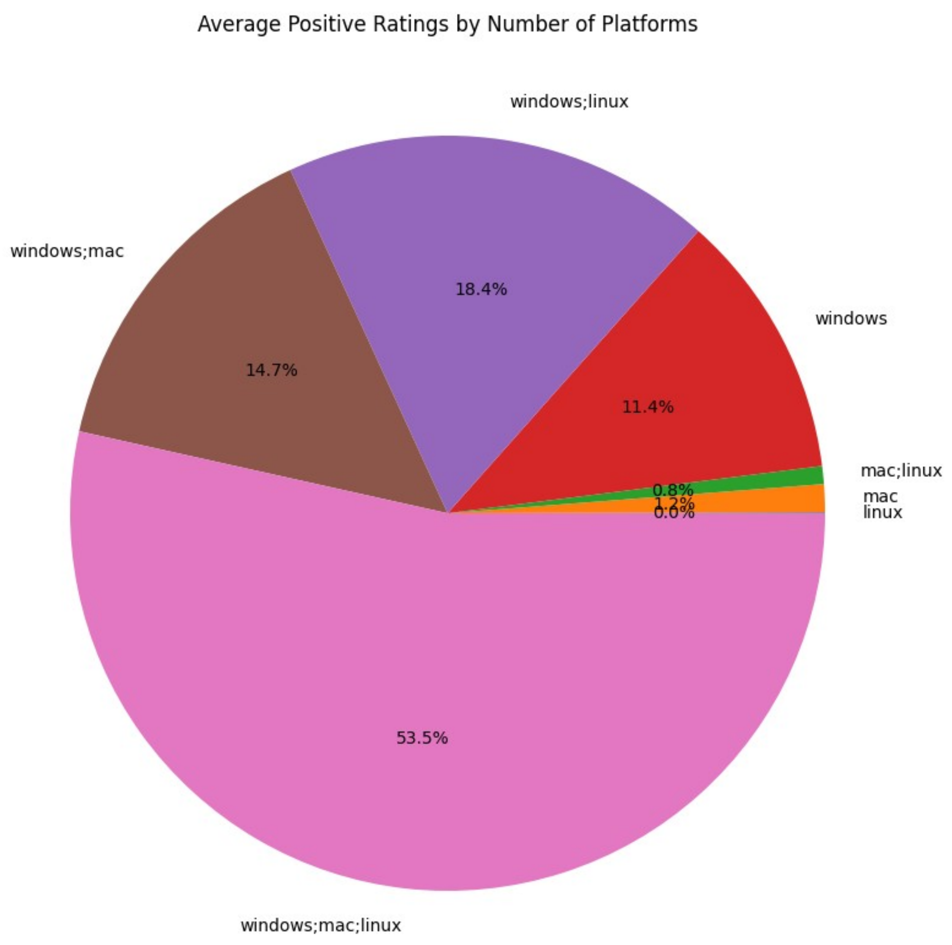
Creating a bar plot:

```
plt.figure(figsize=(10, 6))
average_positive_ratings_by_platforms.plot(kind='bar')
plt.xlabel('Number of Platforms')
plt.ylabel('Average Positive Ratings')
plt.title('Average Positive Ratings by Number of Platforms')
plt.xticks(rotation=45)
plt.show()
```

The results are the following chart:



Another way of showing this our favorite pie chart:



This also confirmed our hypothesis the more platforms a game averaging on a good score has more it has more people giving it good ratings.

This concludes our answer to the Question 1.

For the second question we chose K means clustering as our method. Let’s briefly explain it first:

It means to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
Centroid point is computed as the mean or average of all the data points that belong to a particular cluster. So it’s like a representative.
Centroids helps in distinguishing one cluster from another based on their proximity or distance to the other centroids.

This way we will try to discover any patterns. It will provide insights into the relationships between different data instances.

Thinking of the Iris dataset, k-means clustering can be used to group the iris flowers based on their features such as sepal length, sepal width, petal length, and petal width. This could help in understanding the natural groupings of the iris flowers and maybe even uncovering different species or variations within the dataset.

The purpose of the K-means clustering method is to partition a given dataset into K distinct clusters, where K is a pre-defined number chosen by the user. The goal is to minimize the within-cluster variance, which measures the similarity of data points within each cluster while maximizing the dissimilarity between clusters.

The formula for K-means clustering involves two key steps: assignment and update.

1. Assignment Step:

For each data point, calculate its distance to each cluster centroid using a distance metric, commonly the Euclidean distance. The formula for the Euclidean distance between a data point x and a centroid c is:

d(x, c) = sqrt((x1 - c1)^2 + (x2 - c2)^2 + ... + (xn - cn)^2)

Here, (x1, x2, ..., xn) and (c1, c2, ..., cn) represent the coordinates of the data point and centroid, respectively.

Assign each data point to the nearest centroid by selecting the cluster with the minimum distance.

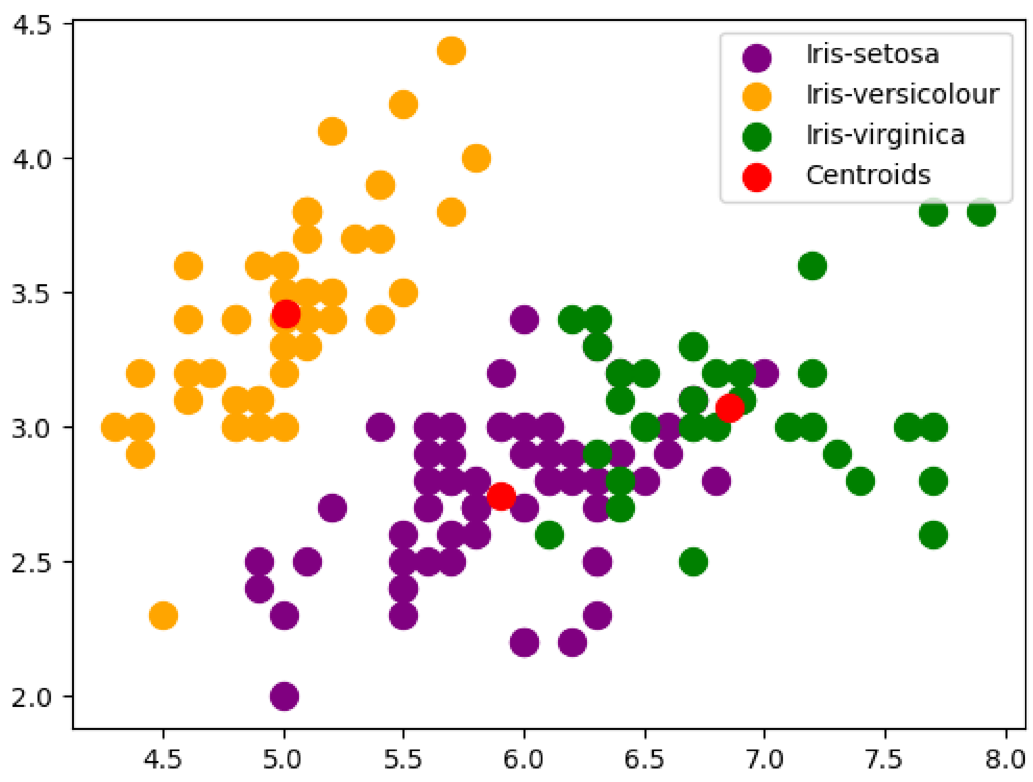
2. Update Step:

After the assignment step, update the centroids of each cluster based on the newly assigned data points. The updated centroid is calculated as the mean of all the data points belonging to that cluster. The formula for updating the centroid coordinates is:

ci = (x1 + x2 + ... + xn) / n

Here, (x1, x2, ..., xn) represents the coordinates of the data points assigned to cluster i, and n is the number of data points in that cluster.

These two steps of assignment and update are repeated iteratively until convergence is achieved, typically when the centroids no longer change significantly or a maximum number of iterations is reached. The final result is a clustering solution where each data point belongs to one of the K clusters.



The first three plt.scatter() calls plot the data points of each cluster separately.

The indexing X[predicted_clusters == i, 0] selects the x-coordinates ([:, 0]) of the data points where predicted_clusters is equal to the cluster label i.

Similarly, the indexing X[predicted_clusters == i, 1] selects the y-coordinates ([:, 1]) of the data points for that cluster.

The s parameter sets the size of the markers, and the c parameter sets the color of the markers. Each cluster is assigned a different color and labeled accordingly.

The last plt.scatter() call plots the cluster centroids obtained from the k-means clustering algorithm. kmeans.cluster_centers_[:, 0] selects the x-coordinates of the centroids, and kmeans.cluster_centers_[:, 1] selects the y-coordinates. The centroids are plotted with a red color and labeled as 'Centroids'.

Finally, plt.legend() is used to display the legend, which shows the labels and corresponding colors of the different clusters and centroids.

By visualizing the clusters and centroids in this way, you can gain insights into how the k-means algorithm has grouped the Iris data points and where the cluster centroids are located.

Some uses of k means cluster in real world include:

- Text categorization
- Face Detection
- Signature recognition
- Customer discovery

Used data sets and other sources:

https://www.kaggle.com/datasets/nikdavis/steam-store-games

https://www.kaggle.com/datasets/uciml/iris

https://neptune.ai/blog/k-means-clustering

https://www.kaggle.com/code/hamelg/python-for-data-analysis-index/notebook

https://www.youtube.com/watch?v=Z2VXrwMYHFk

https://www.linkedin.com/pulse/k-means-clustering-its-real-world-use-case-pratik-kohad-1c

https://www.knowledgehut.com/blog/data-science/eda-data-science