Mini Project 2

Omkar Bhatt  
(112715227)

**Video Link:** <https://youtu.be/Ra-5aCn13Sg>

**d3 Version:** v4

**File Structure:**

* Data:
  + PS2\_GameSales.csv: The main dataset being used. This has 1000+ attributes and 11 dimensions.
  + main\_data.csv: copy of the main data which is not being used anywhere in the code.
  + main\_data\_onehot.csv: The main data which has been modified used Standard Scalar in python, so it follows a standard deviation and makes it easy to compute other metrics easily on.
  + main\_data\_random.csv: The data used for random sampling
  + main\_data\_stratified.csv: The data after stratified sampling
* Static:
  + Main.css: Has all the styles for all the elements in the application
  + pca\_graph.js: Responsible for creating the PCA graph which should bar charts and line charts together rot compare the 3 datasets and their PCAs.
  + pca\_scatter.js: A dynamic function which takes in dataset and constructs a scatter plot on it.
  + pca\_top3\_scatter.js: A dynamic function to create scatter plot matrix for a given data set. This makes it easy to call it for all 3 datasets.
  + UI\_control.js: Responsible to manage dropdown menus and call the correct js file when dropdown option is selected. Also has the GET calls to get the right data set from the app.py file.
* Templates:
  + Base.html: Have a base layout of an html page, which can form as a parent to any html page going forward. Useful when having multiple pages and having same UI for all of them. Not quite useful in the current implementation.
  + Index.html: The main HTML page which contains and calls all the js scripts and showcases svgs for the visualizations. This is being call by app.py on start.
* Python Scripts:
  + App.py: Forms the control between processing python scripts and the main page. Handles all the GET interactions from main and returns needed data for visualizations.
  + data\_sampling.py: Has the code which reads the main dataset and samples it into random sampling and stratified sampling. It also has a few functions which help to cluster and check the elbow for best clustering.
  + pca\_analysis.py: Does PCA and MDS calculations. Responsible to return PCA Component values, Top 2 PCAs, Top 3 PCA datasets and MDS
* Report.docx: this.file();

**Data:**

A new dataset has been used than the previous project. Tried to find a dataset with maximum numerical values as this will help to give output of nice PCAs and compare the biases produced by the different sampling methods.

The dataset being used is called Video Game Sales with Ratings and contains various gaming titles with their sales and rating information. The dataset had 11k rows, but this would not help is our hypothesis as MDS would work really slow on so many values. The first task was to reduce the data in length. So, only game titles on PS2 were selected form this dataset which helps us get the data set down to 1k rows.

Further, to keep all columns numericals, the Rating column was converted from categorical to numerical. It has the values Everyone (Kids-Adults), 10+, Teen, Mature, Adults Only, which were converted to a 1-5 scale. In this case it was easy to align a score to them as they can be easily differentiated in an ascending order. Otherwise one hot encoding would have been a better approach.

The data can be found here:

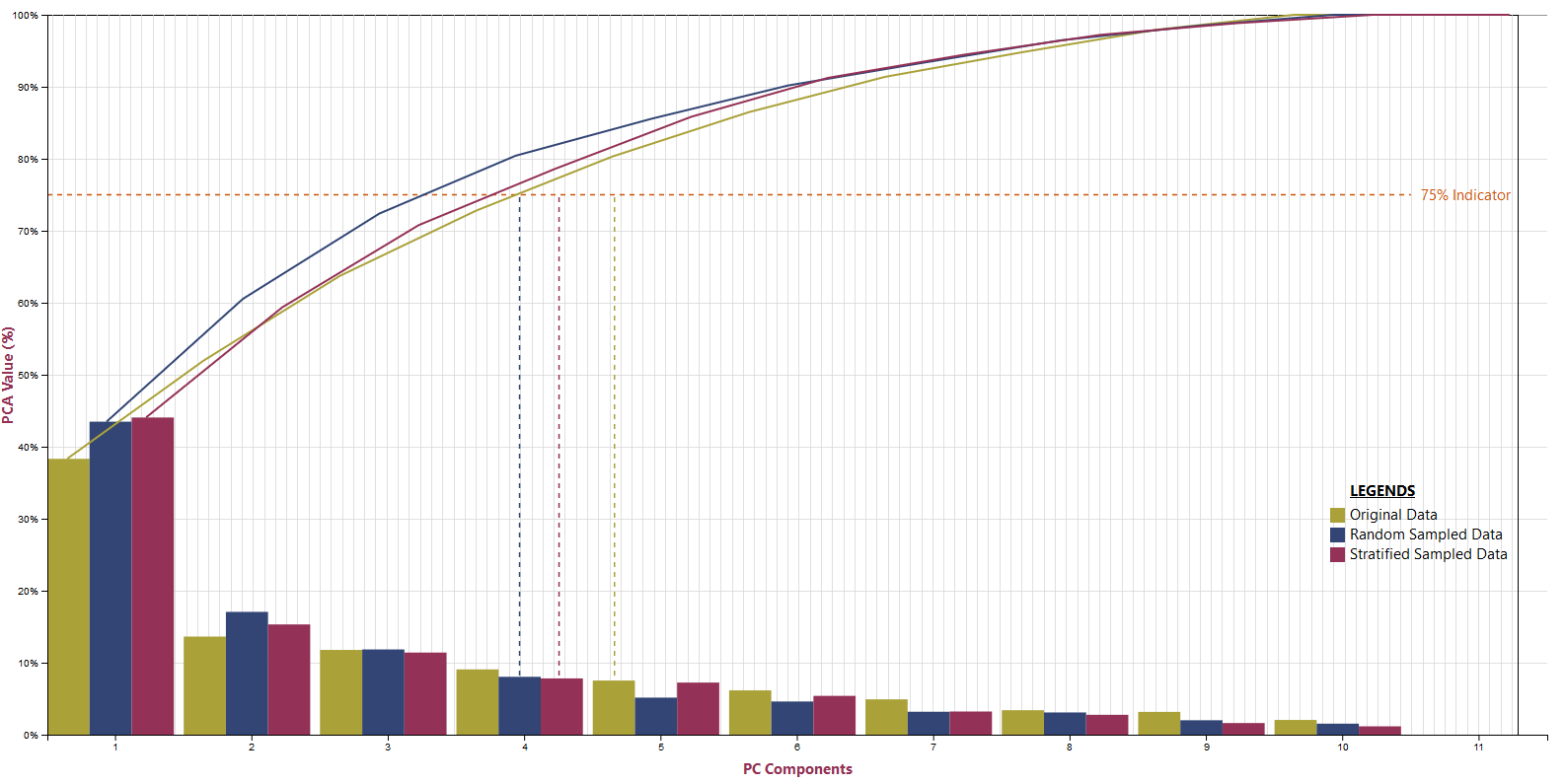
<https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings>

The dimensions used are:

1. Year\_of\_Release: When was the game released
2. NA\_Sales: Total sales in North America
3. EU\_Sales: Total sales in Europe
4. JP\_Sales: Total sales in Japan (not the best metric for PS2 games)
5. Other\_Sales: Total sales for the rest of the globe
6. Global\_Sales: Total sales for the globe
7. Critic\_Score: Score provided by Metacritic critics, out of 100
8. Critic\_Count: Total critic votes
9. User\_Score: Score provided by Metacritic users, out of 10
10. User\_Count: Total user votes
11. Rating: Content rating given to the game

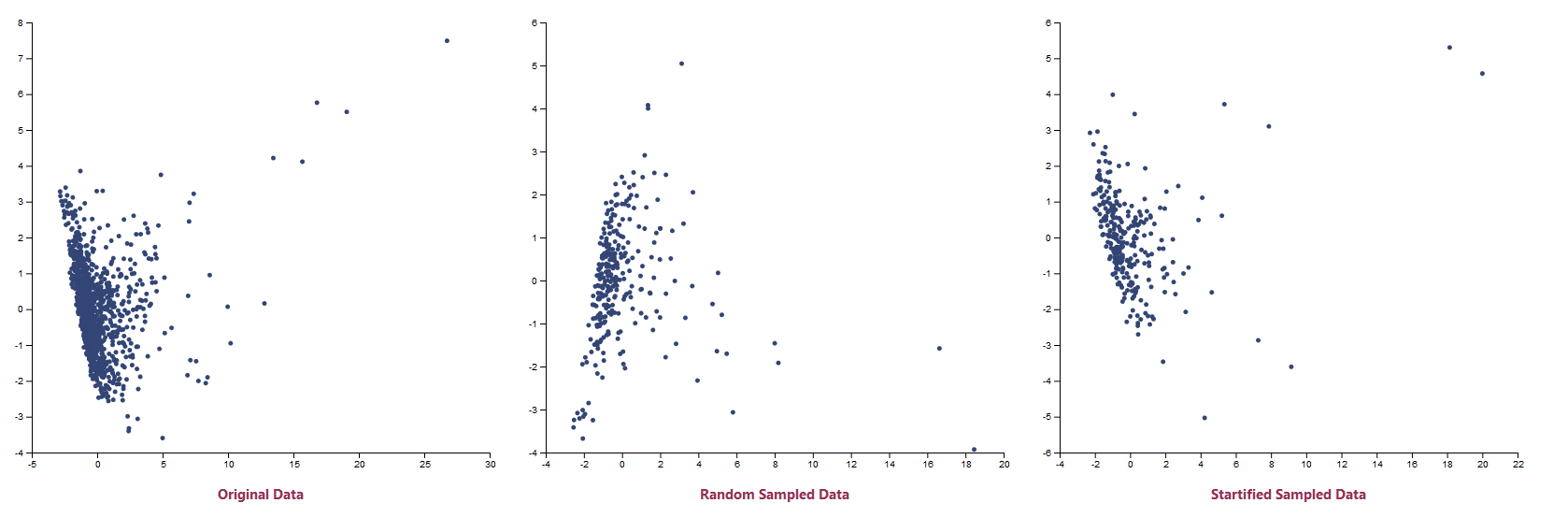
**Code Implementation:**

1. Flask – app.py:
   1. This is the main file which is called when flask runs a virtual server to run the website with visualizations.
   2. This file is responsible to read the data, process it and keep it stored so that various calls from front end can be satisfied by responding with the data.
   3. Currently, this file checks for the data files, and if it does not find them, it calculates the new data files and samples them on the go into random and stratified; also saves them. This way, it will always have data ready.
   4. It has a main route, which return the index html page and waits for any request from the html page.
   5. Rest of the code in here acts like a GET request. It waits for a call from the js and knows what data needs to be returned at what call.
2. Data Sampling:
   1. The main data is read and stored in Pandas.
   2. The data is then sent into a standard scalar, which is a python function, which converts the data into standard values, making it easy for them to compare to each other, using mean and standard deviation.
   3. Random: 25% of the total data is randomly is selected and put into a new pandas object for further use.
   4. Clustering: Next, we check the error achieved in the main dataset when we cluster it on multiple k clustering values. An elbow is found on this curve and selected as a good clustering number.
   5. All the data is now clustered into these n clusters.
   6. For our current data, the best n value found was 4, meaning 4 clusters would be best to separate our data.
   7. From each cluster, we select 25% values and make a new dataset to form an evenly distributed dataset of the original where clustering ratios are maintained.
3. PCA Calculations:
   1. Intrinsic Dimensionality: From any given data, a PCA is fit on it to find the PCA components responsible for the clustering. Or something that helps to sperate the data well.
   2. We take taken 11 PCA components, which is the same as each column. Thought the PCAs do not exactly correspond to the columns, it helps to get a good division on the PCA values.
   3. This gives use intrinsic dimensionality for each PCA and this can be summed up to get a cumulative intrinsic dimensionality for all the PCA which can help produce a scree plot.
   4. The plot shall help in understanding which PCAs are helping give out a better output.
   5. This also gets the PCA value for each PCA columns, helping to get the Top2 PCAs which can be later plotted as a scatter graph to understand how much the contribute to data’s uniqueness.
   6. When all the PCA values are squared and added at column level, we can also get the top attributes responsible for making the top PCAs. This can be used to get the Top 3 attributes which affect our data the most.
   7. The data for these top 3 attributes can then be used to make a scatter plot matrix to understand the uniqueness and how the bias can be introduced in the data.
4. MDS calculations:
   1. MDS helps to reduce from a higher dimensionality to a lower dimensionality.
   2. In our case, its being converted from 11 to 2.
   3. This is being done by creating a dissimilarity or distance matrix between all points.
   4. Then this is being fitted into a 2-dimensional plane to give it more meaning and understand how the data points can be similar.
   5. This can be eventually plotted on a scatter plot to show meaning inference.
5. PCA\_Graph – Scree Plot:
   1. We get the intrinsic dimensionalities for each PCA and their cumulative intrinsic dimensionalities, which will be used to make this graph.
   2. First the svg is created to make the bar graph for intrinsic dimensionality.
   3. Then a line graph is used to plot the cumulative intrinsic dimensionalities. We see this line rising from the first bar graph and becoming 1 eventually.
   4. This shows us how the top 4 PCAs are helping create 75% of all the uniqueness in out data.
   5. The graph shows a line at 75% and plots the graph for each dataset which contribute to the 75%.

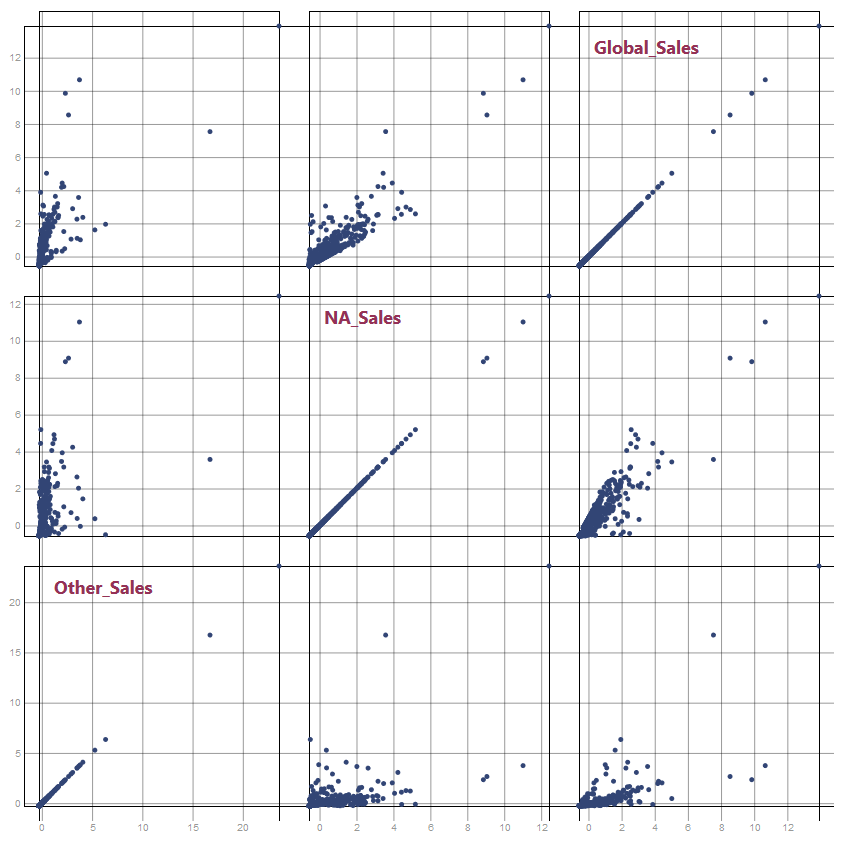


* 1. There are different colours used to show different datasets and have been mentioned in the legends.
  2. We see that in Random and Stratified, lesser PCAs are responsible to form uniqueness, while with the original data.
  3. The gridlines help to show batter which graph matches to which line point.

1. Scatter Plot:
   1. A dynamic reusable function for scatterplot has been written.
   2. This takes care of the following:
      1. Top 2 PCA components
      2. MDS with correlation
      3. MDS with Euclidean
   3. This takes in the required data and creates the required svg for the dataset.
   4. The scatter function is called for each dataset and then put together in a inline block division in html.
   5. This helps to compare the dataset outcomes.
   6. It is observed that stratified data is closer to the original dataset compared to random sample.
   7. Also, the outlier which are randomly ignored in sampled data are handled better at stratified level as they are taken from clusters.



1. Scatter Plot Matrix:
   1. This code has been taken directly from reference.
   2. The data set with the top 3 attributes is collected and used to plot it
   3. The attribute names are visible to help understand what they are
   4. It was noted that with attributes which have less unique value like no. of player (1,2,3,4) it didn’t give a good spread which would reduce the uniqueness of the data.
   5. It is seen that random data sampling takes these attributes as high PCA and hence introduces a good bias into the dataset.
   6. The stratified shows 2 common top attributes with original dataset, rather than random which shows all new attributes.



1. UI Control:
   1. This is the main control which connects the html page with the python processing.
   2. It first is responsible for the interaction with dropdowns on the UI and keep the UI consistent by updating the needed data on them.
   3. Also handles the two dropdowns with each other.
   4. Have a switch case which identifies the selected dropdown and gets the required data accordingly.
   5. It removes any current svg on the screen to accommodate new ones.
   6. Then passes the data received into the right JavaScript function to produce the needed graph.
   7. This also makes it easier to make dynamic GET links
   8. The outputs received are easy to send into js function and produce graphs easily.
   9. This is the controller to the whole website connecting calls to the server and any browser based scripts.

**References:**

1. Bar graph: <https://bl.ocks.org/d3noob/bdf28027e0ce70bd132edc64f1dd7ea4>
2. Line graph: <https://bl.ocks.org/d3noob/402dd382a51a4f6eea487f9a35566de0>
3. Scatter plot: <https://bl.ocks.org/sebg/6f7f1dd55e0c52ce5ee0dac2b2769f4b>
4. Scatter plot matrix: <https://bl.ocks.org/Fil/6d9de24b31cb870fed2e6178a120b17d>
5. Dropdowns: <https://materializecss.com/dropdown.html>