VISUALISATION MINI PROJECT 2

I have selected a MOVIE dataset with 15 attributes (9 categorical and 6 numerical variables). These were initially 5 different datasets that I downloaded from Kaggle:
1. Movie_Actors. 2. Movie_Genre 3. Movie_Writer
4. Movie_Movies 5. AdditionalData
I selected some columns from each of these datasets and merged them using the imdbID attribute and randomly sampled 250 tuples for this assignment.

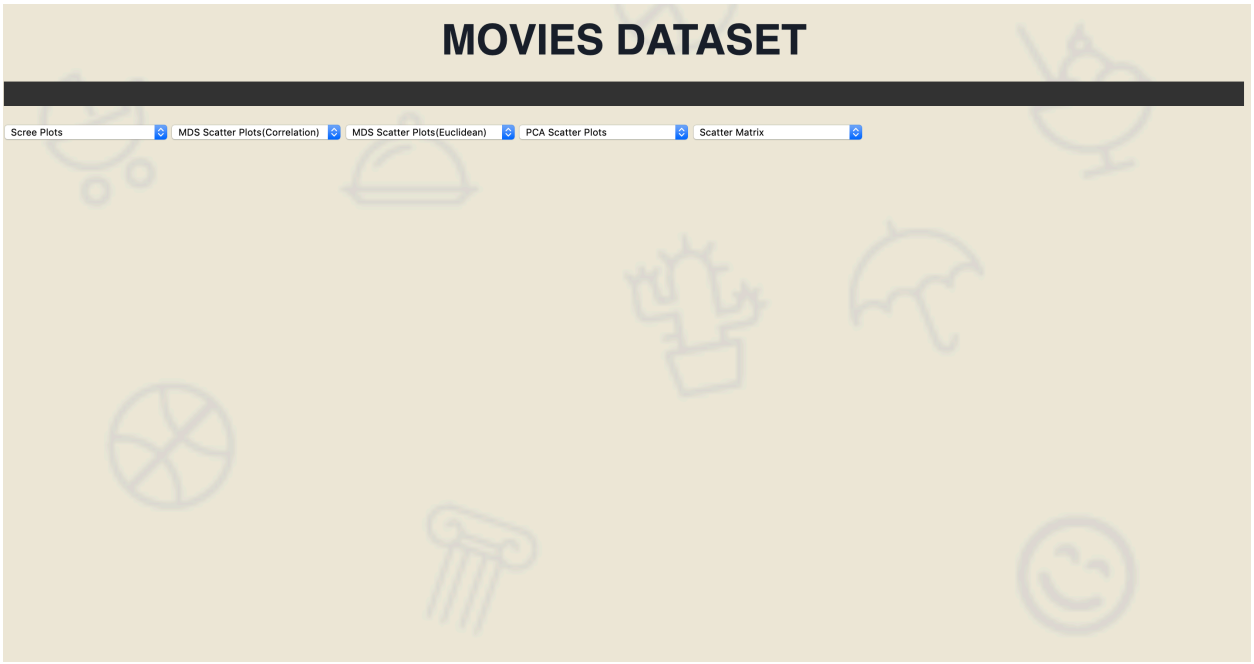
This is the same dataset that I used for Project 1.

IMPLEMENTATION:

Server: I have used XAMPP server for this assignment.
I have used **HTML**,**CSS** and **Javascript** in this assignment.

1. In the html file(**index.html**), I have added all the components of the web page i.e the heading and the buttons.
2. In the CSS file(**style.css**), I have done the formatting and styling of the components added on the web page.
3. In **barr.js** is the implemention of the plots that I have made is added.
4. I have used Flask as the server and used python(**app.py**) at the backend.

Following is the screenshot of my website:



Initially I used one hot encode to encode all the variables but it resulted in 67 variables and a lot of 0's and 1's therefore I used label encoding for encoding the categorical variables.

I have used menu buttons as it is easier to select the options from the dropdown menu rather then using navigation menu bar as we had a lot of options in this experiment.

The project consists of the following task:

Task1:

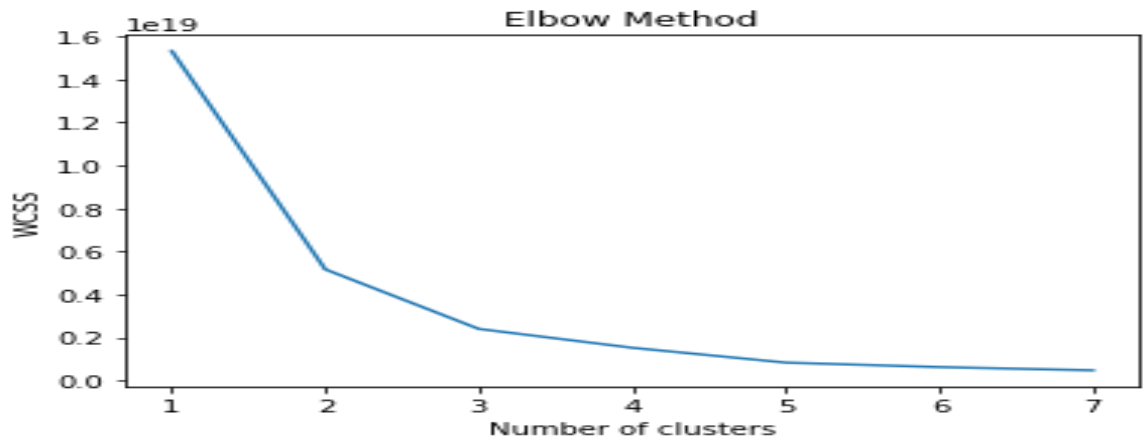
Task 1: data clustering and decimation :
Implement random sampling and stratified sampling

Random Sampling:
For random sampling I used **DataFrame.sample** method which selects a given fraction of data from the given dataset.

Stratified Sampling:
For this sampling we first need to find appropriate number of clusters and then randomly sample each cluster of data.
To find the correct number of clusters we use elbow method.

Following is the code snippet and output of the elbow:

```
def find_elbow():
    wcss=[]
    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
        kmeans.fit(result1)
        wcss.append(kmeans.inertia_)
    plt.plot(range(1, 11), wcss)
    plt.title('Elbow Method')
    plt.xlabel('Number of clusters')
    plt.ylabel('WCSS')
    plt.show()
```



From the image we can it was a little confusing so I ran accuracy test on both 2 and 3 clusters and finally got the number of clusters as 3.
So after task 1, I got 3 type of data:
Original_data
Random_data
Stratified_data
Task 2:

1. dimension reduction on both org and 2 types of reduced data
2 find the intrinsic dimensionality of the data using PCA
3 produce scree plot visualization and mark the intrinsic dimensionality
4 show the scree plots before/after sampling to assess the bias introduced
5 obtain the three attributes with highest PCA loadings

2.1:
For dimensionality reduction I have used 2 algorithms PCA and MDS on both all the types of data.

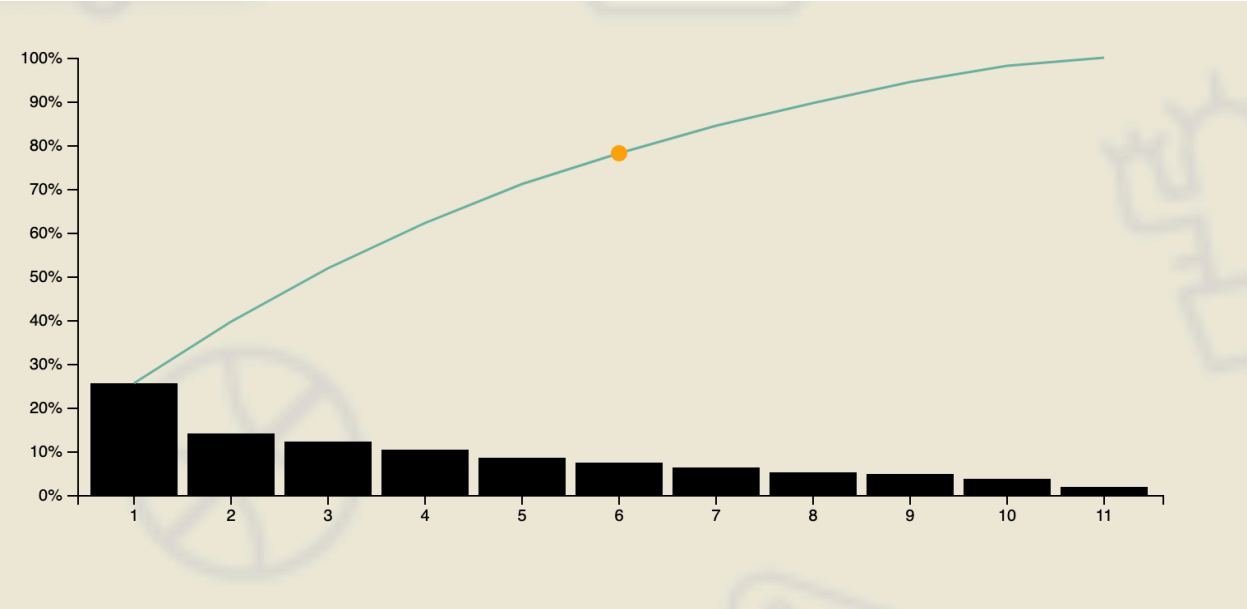
2.2 and 2.3:
For finding the intrinsic dimensionality of the of the data, we first need to find the explained variance ratio using pca and calculate the cumulative explained variance ratio and the point where the cumulative value crossed 75% is our intrinsic value.

Folloowing the code snippet of finding the intrinsic value of the original data:

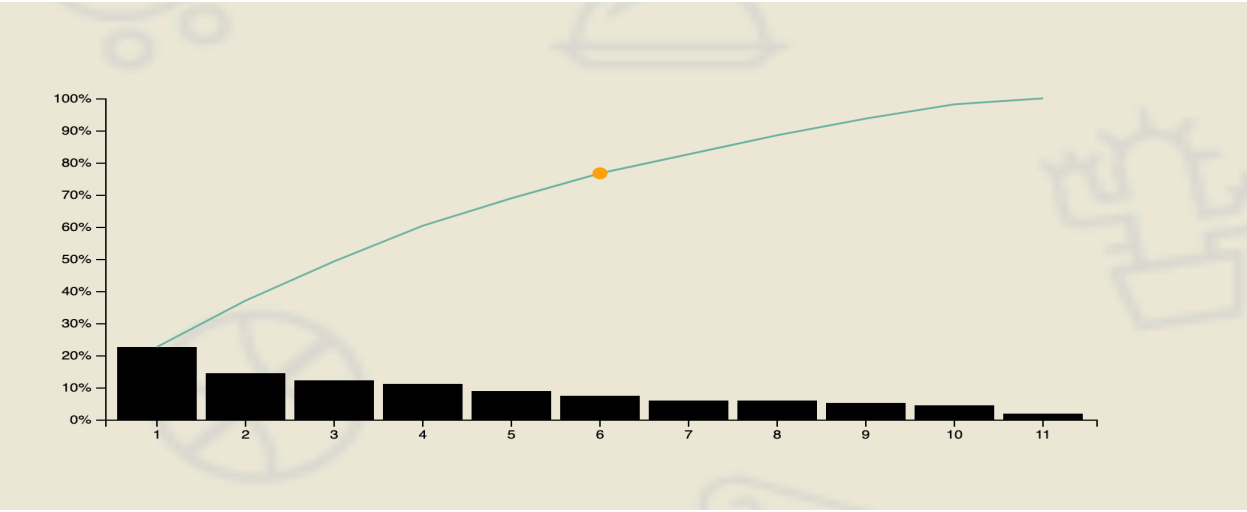
```
@app.route('/original_data_scree',methods=['POST'])
def original_data_scree():
    pca = PCA()
    principalComponents = pca.fit_transform(original_data)
    features = range(pca.n_components_)
    x=pca.explained_variance_ratio_*100
    X1=x.tolist()

    cum=[]
    sum=0;
    for i in range(0,11):
        sum=sum+a[i]
        cum.append(sum)
    intrin=0;
    for i in range(0,11):
        if cum[i]>=75:
            intrin=i+1
            break
```

Following is the plot that marks the intrinsic parameter on the scree plot:

Scree plot is the plot of PCA components vs the explained_variance_ratio.
From the plot we can see the orange point which marks the intrinsic parameter for the original data and the value is 6 in this case.
The line shows the cumulative value and y axis denotes the explained variance ratio and x axis the PCA components.



Scree plot of random_data with the intrinsic paramter



Scree plot of stratified_data with the intrinsic parameter

2.4 So we can see that there is not much difference in the plot and the intrinsic value is the same.

2.5 To calculate the highest loading_attributes I used loading:

```
loadings = np.sum(np.square(pca.components_), axis=0)
indices_of_top_3_attributes = loadings.argsort()[-3:][::-1]
top_two_components = pca.components_[:2]
print(indices_of_top_3_attributes)
```

So the above code snippet shows the code for calculating the top 3 attributes .
And after running it on all the 3 datasets we got almost the same results .

For the original data:

```
[ ]  pca = PCA()
     principalComponents = pca.fit_transform(original)


     loadings = np.sum(np.square(pca.components_), axis=0)
     indices_of_top_3_attributes = loadings.argsort()[-3:][::-1]
     top_two_components = pca.components_[:2]
     print(indices_of_top_3_attributes)
```

```
[7 2 1]
```

Where 7 is Responsibility,
2 is Language and
1 is the country

For random_data:

```
     pca = PCA()
     principalComponents = pca.fit_transform(stratified_data)


     loadings = np.sum(np.square(pca.components_), axis=0)
     indices_of_top_3_attributes = loadings.argsort()[-3:][::-1]
     top_two_components = pca.components_[:2]
     print(indices_of_top_3_attributes)
```

```
[9 7 2]
```

Where 9 is the totalVotes.
We cans ee that 7 and 2 are common in both

And similarly for stratified sampling I got 7,2,6.

So at the end of task 2, I made the scree plot, calculated the intrinsic parameters and also the top 3 attributes.

TASK 3: Visualization of both original and 2 types of reduced data

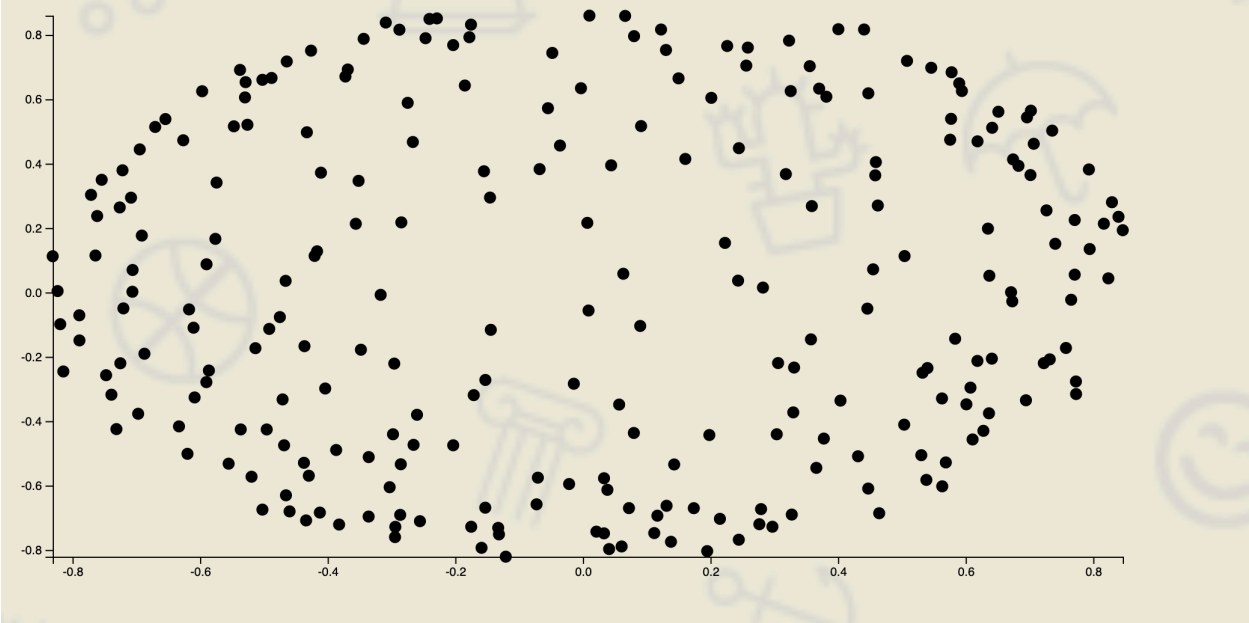
1. Visualize the data projected into the top two PCA vectors via 2D scatterplot
2. Visualize the data via MDS (Euclidian & correlation distance) in 2D scatterplots
3. Visualize the scatterplot matrix of the three highest PCA loaded attributes

This task was to visualize the data using scatter plots. We made 3 types of scatter plots with 3kinds of data.

1. PCA scatter plot:
   In this we used pca for dimensionality reduction and then plotted the top 2 attributes on a scatter plot using all the 3 types of data.

   Following is the code snippet and screenshot of the plot of the random_data:

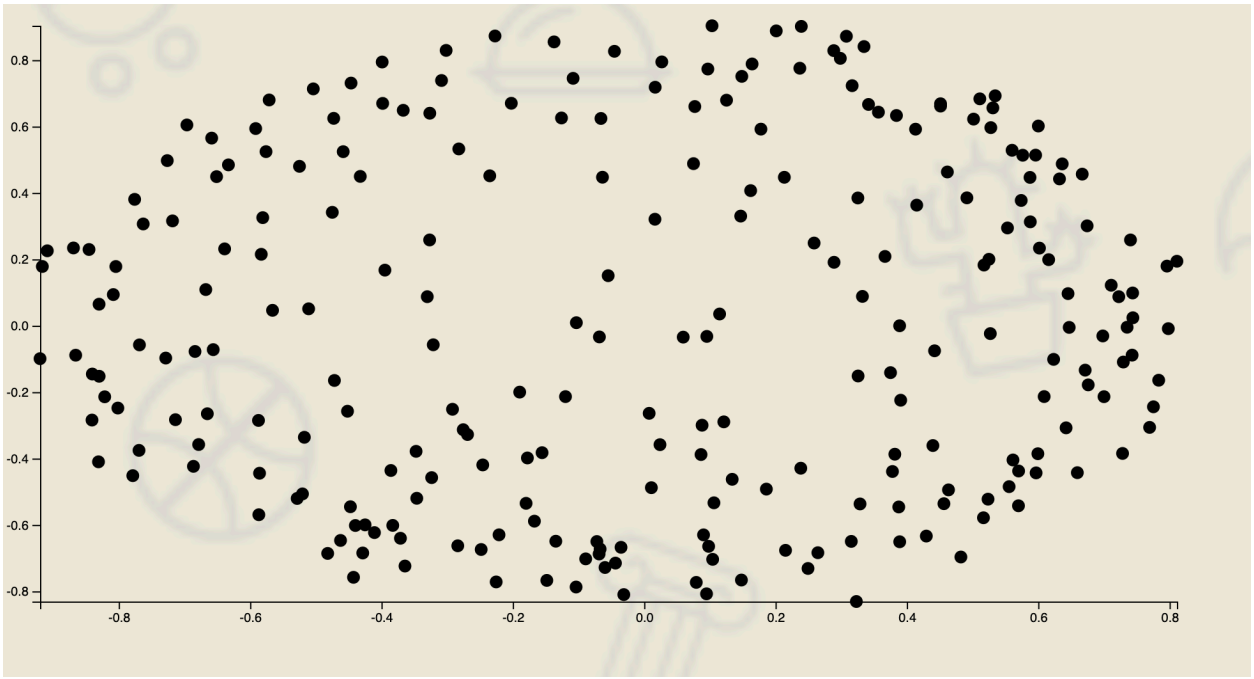
```
4
5   @app.route('/pca_scatter_random',methods=['POST'])
6   def pca_scatter_random():
7       random_data = original_data.sample(frac =.5)
8       pca = PCA(n_components=2)
9       out = pca.fit_transform(random_data)
0       output1=out[:,0].tolist()
1       output2=out[:,1].tolist()
2       final=[]
3       for i in range(len(output1)):
4           final.append({'A':output1[i],'B':output2[i]})
5       print(final)
6
7       return jsonify(final)
8
```



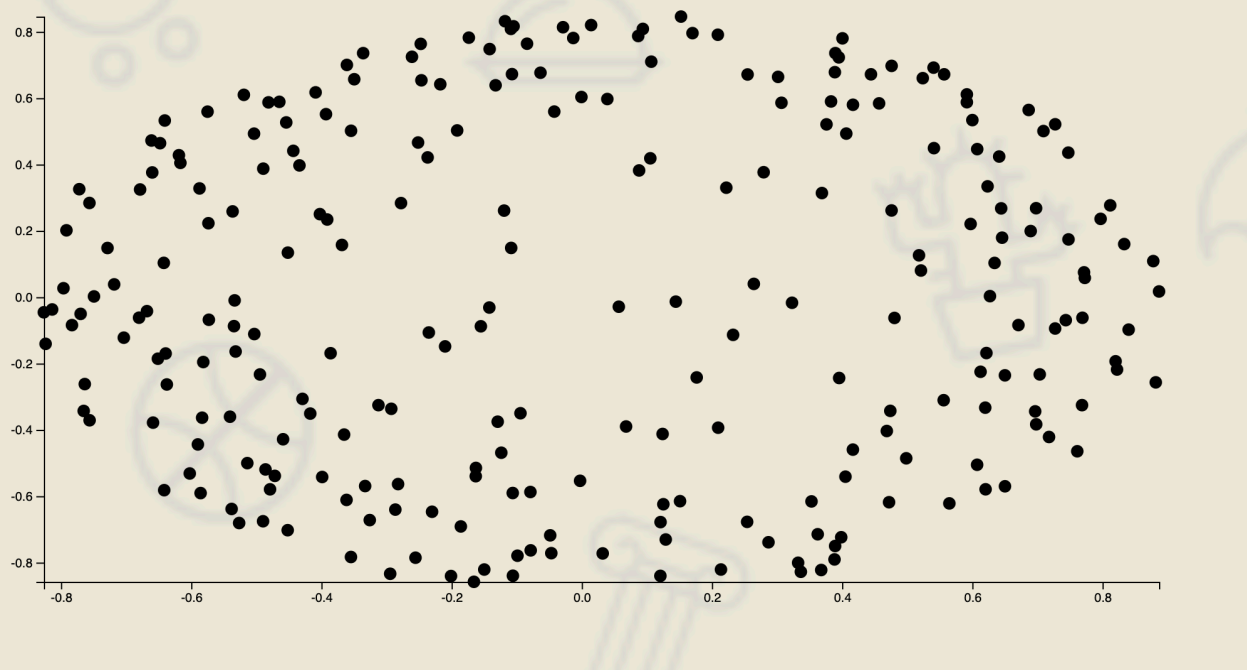
We get almost the same plot using stratified data.

2. MDS using metric as Correlation and Euclidean:

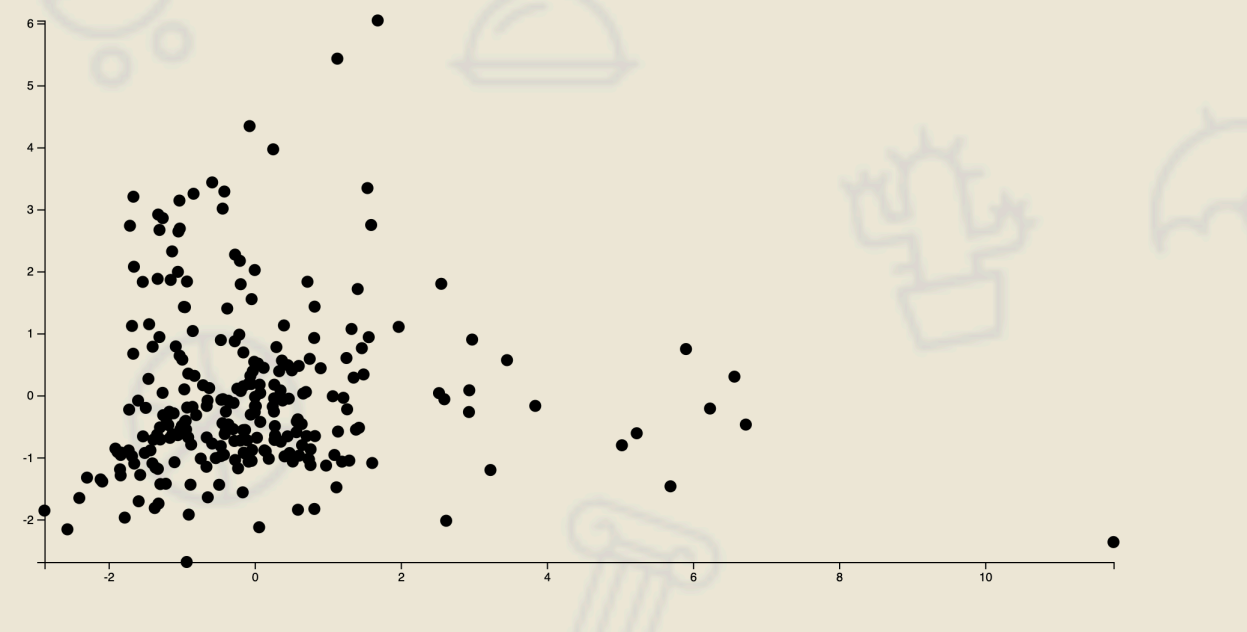
Correlation Random



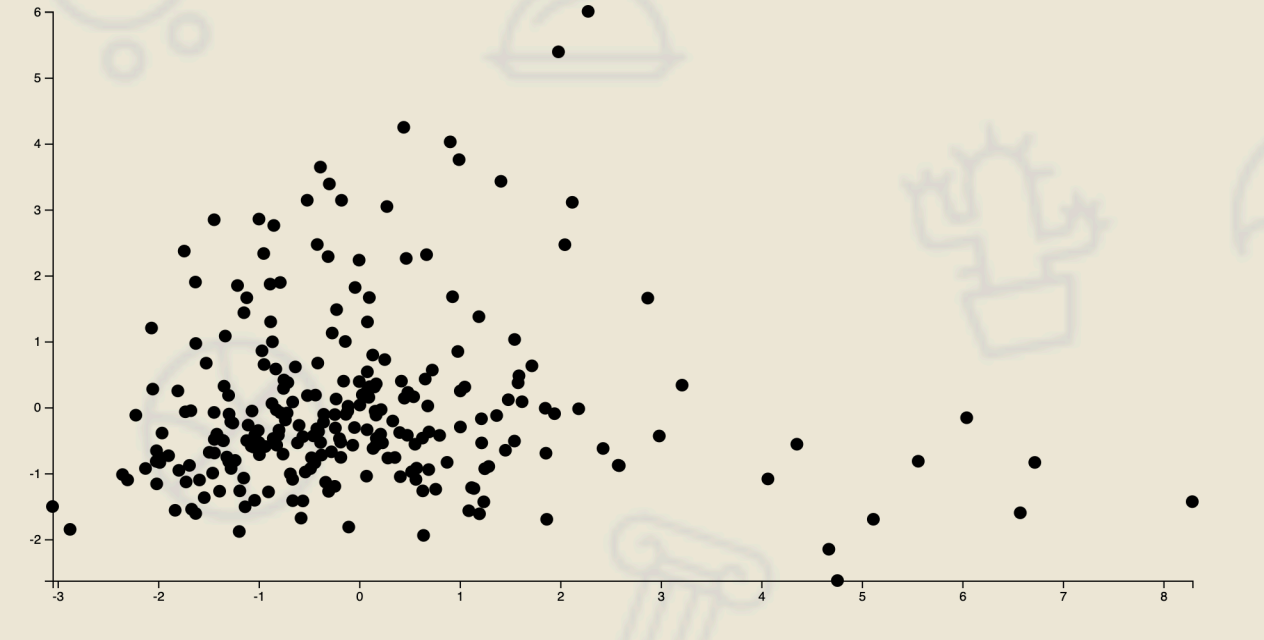
Correlation Stratified



Euclidean Random:

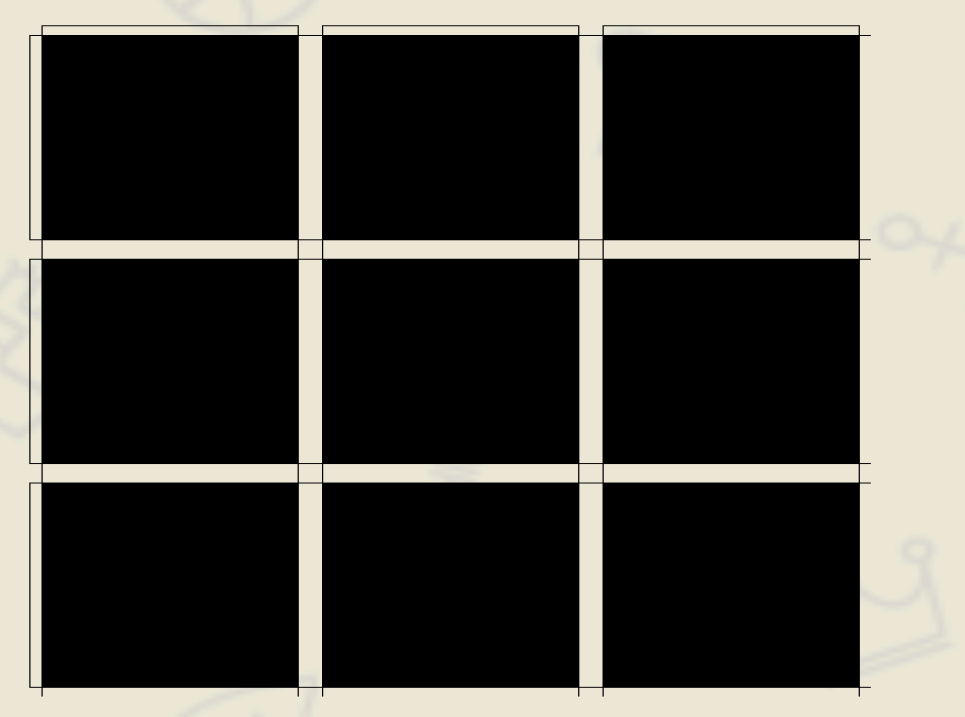


Euclidean stratified:



So from all these plot we can see that pca plots and mds plots using correlation are amost the same and are more spread out.
Whereas in mds Euclidean plot the points are closer.

2.3:
The last task was to make a scatter matrix but I was unable to complete it.I could only get the matrix.



YouTube video Link:

https://youtu.be/CPKkXhM8Cm4