# CSE 564 Visualization

MINI PROJECT 2 REPORT

Surabhi Sarnot

SBU Id: 112584690

# About dataset:

FIFA 19 complete player dataset. Source Kaggle.

The dataset consists of detailed attributes for every player registered in the FIFA 19. It consists of over 90 numeric and categorical attributes in total, out of which I have chosen about 15 attributes for the assignment.
Attributes that I have chosen are:

- Overall rating
- Balance
- Strength
- HeadingAccuracy
- ShortPassing
- LongPassing
- Dribbling
- BallControl
- Acceleration
- SprintSpeed
- Agility
- ShotPower
- Aggression

**Data cleaning:**

- Rows having null or NA values have been removed.
- Only numerical attributes are considered.

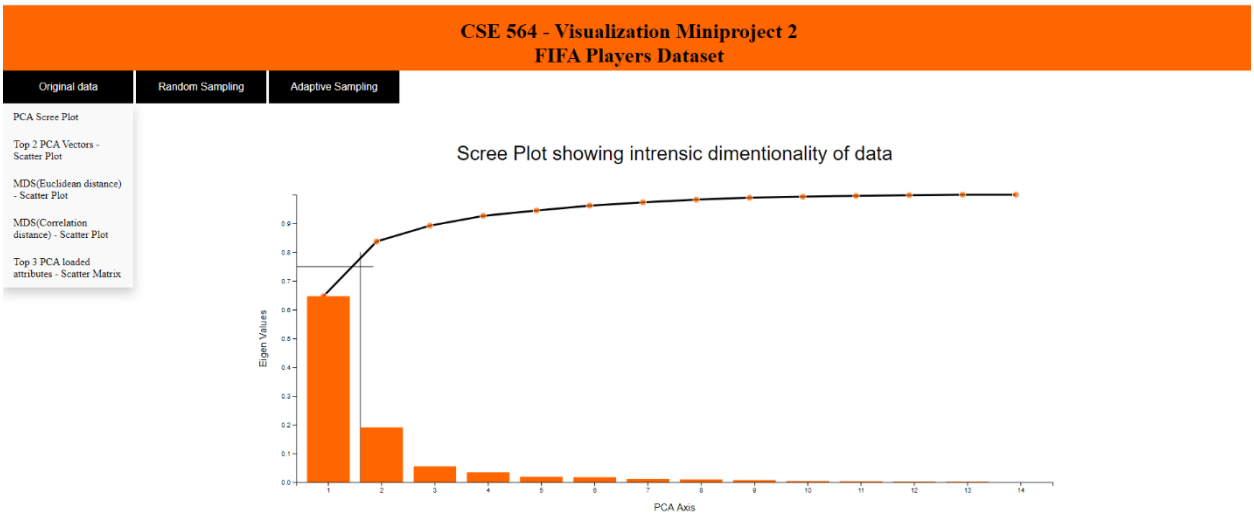
**Project directory structure:**

Project
    static
        menu.js – file containing functions for all calls from server
        matrix.js – javascript code for plotting scatter matrix
        scatterplot.js - javascript code for plotting scatter plots
        screeplot,js - javascript code for plotting scree plot
        stylesheet.css – styling sheet for the project
    templates
        index.html – index web page containing layout
    app.py - python code for all the backend data computations
    data.csv - Dataset file

The project uses python (Flask) for processing (server) and D3 for visualization (client) as mentioned in the assignment description.

**Layout:**

Below is the layout of the developed frontend.




# Task 1: Data clustering and decimation

**Sampling - Implement random sampling and stratified sampling (remove 75% of data)**

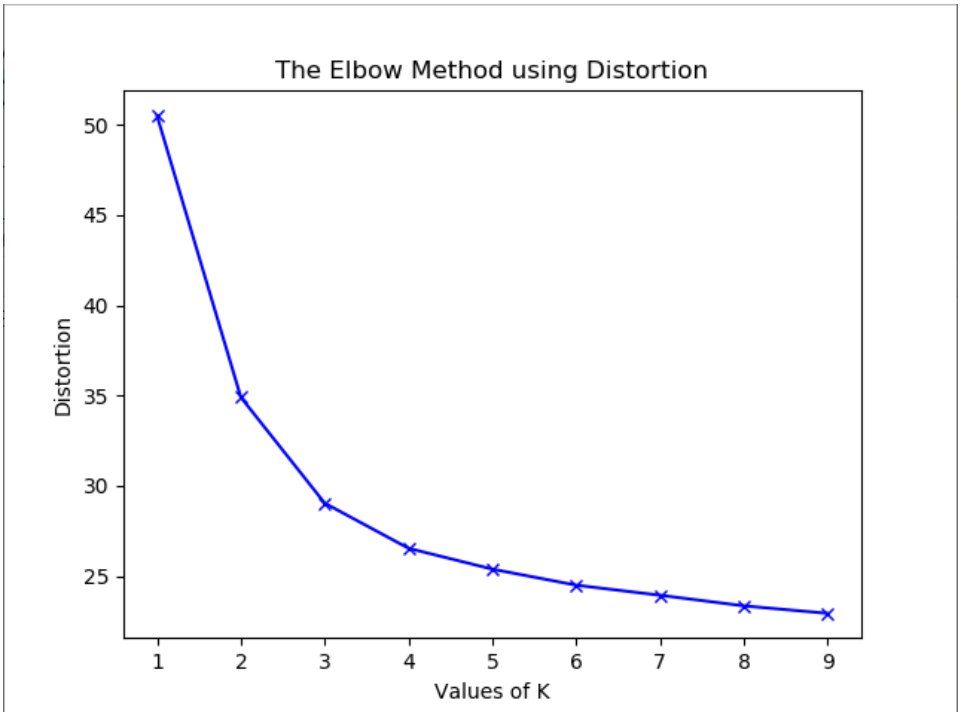
- **Random Sampling**

```
# function to calculate random sampling
def random_sampling(data):
    return data.sample(frac=0.25)
```

I have used dataframe.sample function from pandas library that randomly select rows from the dataframe based on the given percentage.

- **Stratified Sampling**

The data is divided into clusters and random records are selected from each of these clusters. We can find the appropriate number of clusters for the data using Kmeans algorithm and plotting elbow as below.




From this I have taken value of k as 3, ie, 3 clusters will be used to categorise the data and select 25% records from each cluster.

```
km = KMeans(n_clusters=3)
km.fit(df)
df['Label'] = km.labels_
for i in range(4):
    cluster_records = df[df['Label'] == i]
    sampled_df = sampled_df.append(cluster_records.sample(frac=0.25))
return sampled_df
```

In above code, I have taken number of clusters as three and taken 25% of data from each cluster and formed the stratified sampling data.

## Task 2: Dimension reduction on both org and 2 types of reduced data

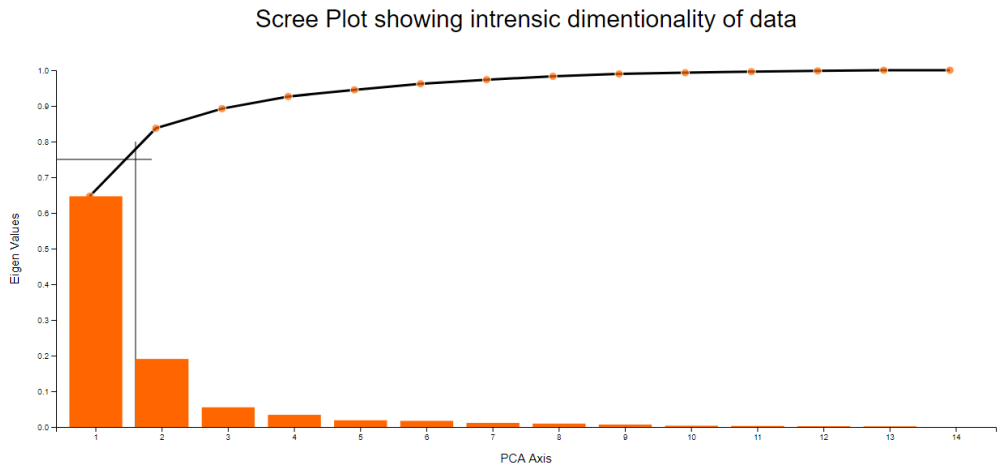
1) **Find the intrinsic dimensionality of the data using PCA**
2) **Produce scree plot visualization and mark the intrinsic dimensionality**

To find the intrinsic dimensions of the data we need to find eigen values, ie, Variance using PCA.
We find the loadings from pca of data and plot them in the scree plot to identify intrinsic dimensionality of data.

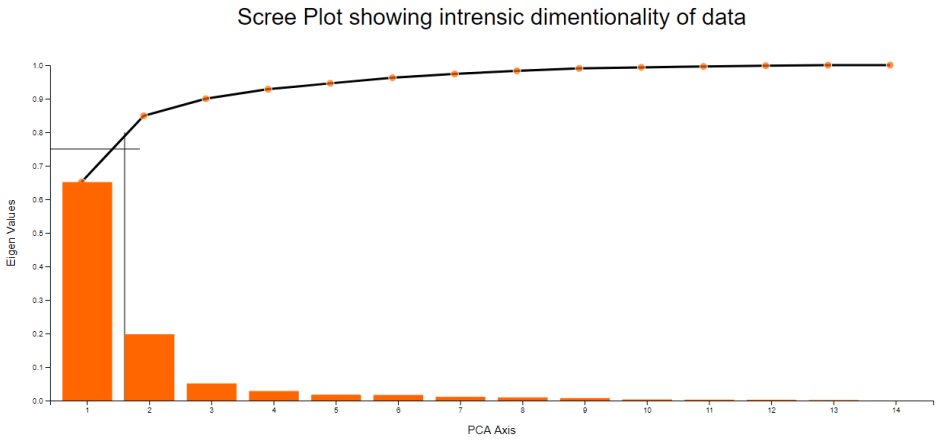
```
def cal_pca(data):
    pca = decomposition.PCA()
    pca.fit(data)
    loadings = np.sum(np.square(pca.components_), axis=0)
    indices_of_top_3_attributes = loadings.argsort()[-3:][::-1]
    return pca.explained_variance_ratio_, indices_of_top_3_attributes
```

3) **Show the scree plots before/after sampling to assess the bias introduced**
For plotting scree plots I have used bar chart and line() function in d3.
I have plotted two lines that intersect at 75% of the data coverage. Below are the scree plots.

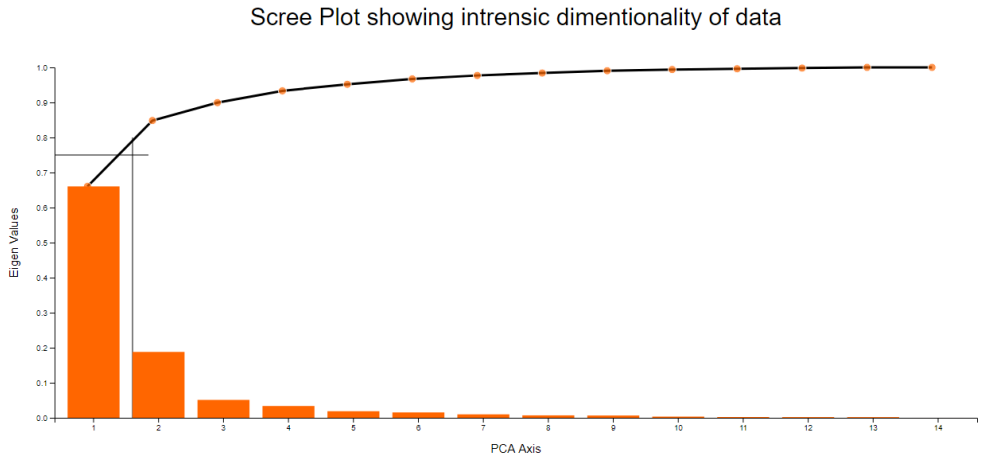
**Original data:**




**Random Sampling:**




**Adaptive Sampling:**




**Observation:**

All the three plots clearly show that the intrinsic dimensionality of the data is 2. First two attributes cover over 85% of the data.
Hence, intrinsic dimensionality of my data is 2.

**4) Obtain the three attributes with highest PCA loadings**

We take sum of squared terms of the PCA components, ie PCA and then sort them to find the top three attributes with highest PCA loadings.
I have printed the values of these loadings on the python console as shown below.

```
127.0.0.1 - - [07/Apr/2020 20:47:18] "GET /scree_plot_original HTTP/1.1" 200 -
Top three PCA attributes are ['Overall', 'Strength', 'Agility']
127.0.0.1 - - [07/Apr/2020 20:47:20] "GET /scree_plot_random HTTP/1.1" 200 -
Top three PCA attributes are ['HeadingAccuracy', 'Agility', 'Aggression']
```

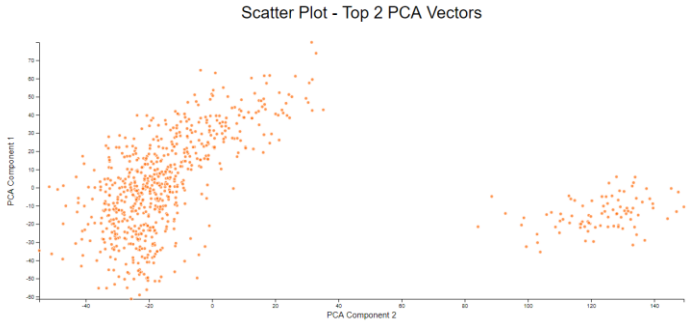
# Task 3: Visualization of both original and 2 types of reduced data

1) Visualize the data projected into the top two PCA vectors via 2D scatterplot
   To obtain data in the top two PCA components, I run PCA function for two components.

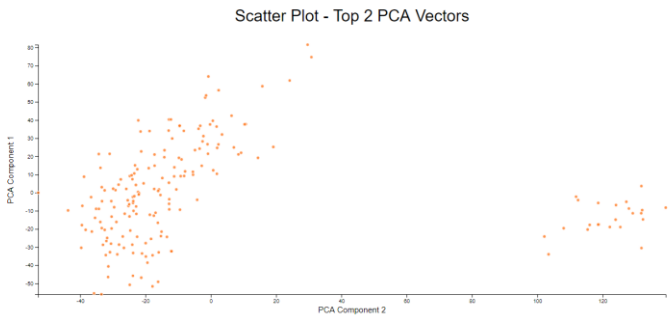
```
def get_top_pca_two_components(data):
    pca = decomposition.PCA(n_components=2)
    return {'pca_components': (pca.fit_transform(data)).tolist()}
```

Then using scatterplot, I have plotted this data. This is done to visualize original, adaptive and random sampling data.

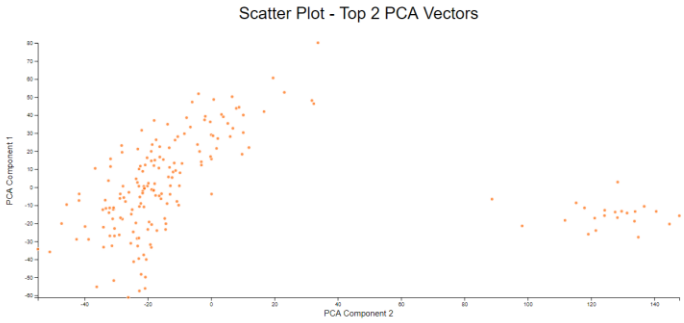
**Original data:**




**Random Sampling**




**Adaptive sampling**




**Observation:**

I all the plots we can clearly see that two distinct clusters are formed. This means that both PCA components have high weightage.

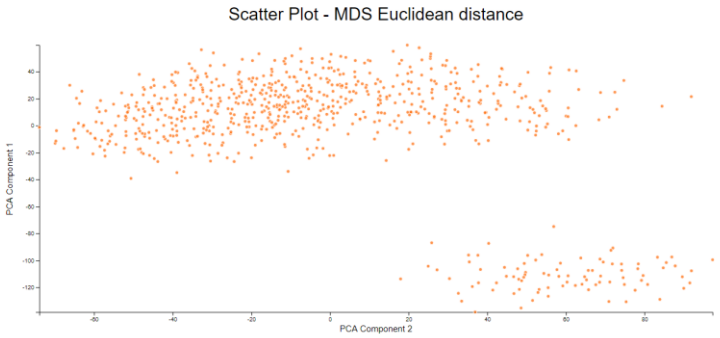
2) **Visualize the data via MDS (Euclidian & correlation distance) in 2D scatterplots**
   **MDS** arranges the points between the two components on the **plot** such that the distance between the points best resembles how dissimilar they are.

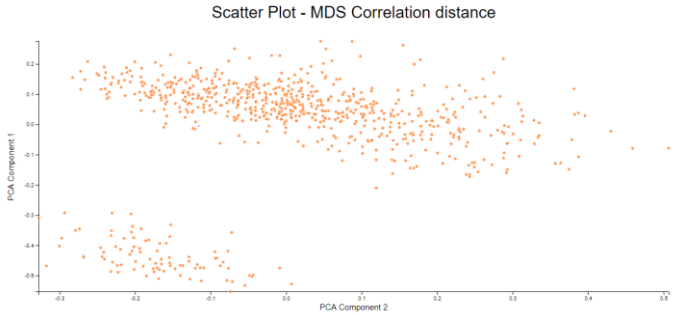
   I have computed mds data using sklearn.manifold library MDS function.
   I have computed it using both Euclidean and correlation distance. The code snippet is as below.

```
def cal_mds_by_euclidean(data):
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    return {'mds_euclidean': mds_data.fit_transform(pairwise_distances(data, metric='euclidean')).tolist()}
```

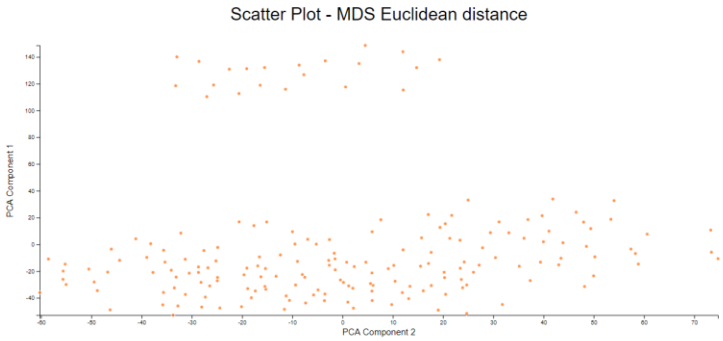
```
def cal_mds_by_correlation(data):
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    return {'mds_correlation': mds_data.fit_transform(pairwise_distances(data, metric='correlation')).tolist()}
```

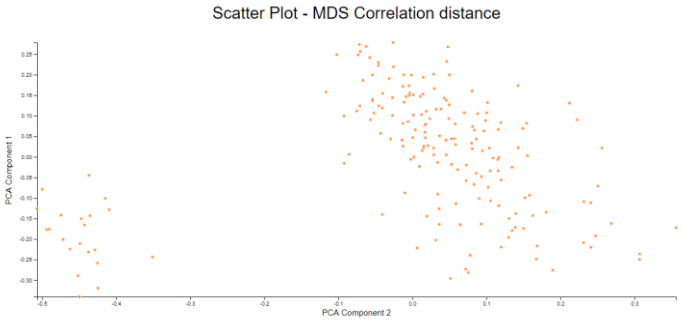
## Original Data



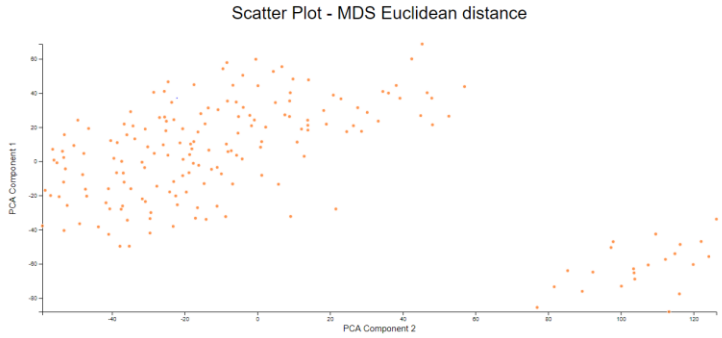



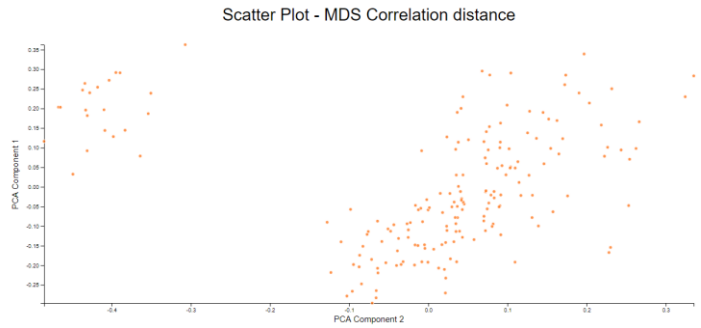

## Random Sampling



**Adaptive Sampling**




**Observation**:

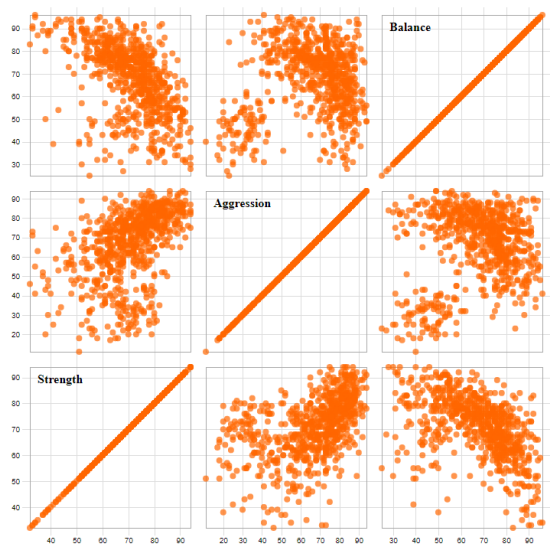
In all the three plots we can see only clear clusters formed and no other formations between the plotted data. The point that are closer are similar to each other and the points that have huge distance between them are very dissimilar to each other. The formation of clusters may signify having interesting patterns in the multivariate data set.

**3) Visualize the scatterplot matrix of the three highest PCA loaded attributes**

Scatterplot matrix is used to visualize bivariate relationships between combination of variables. We can use it to explore many relationships within one chart.

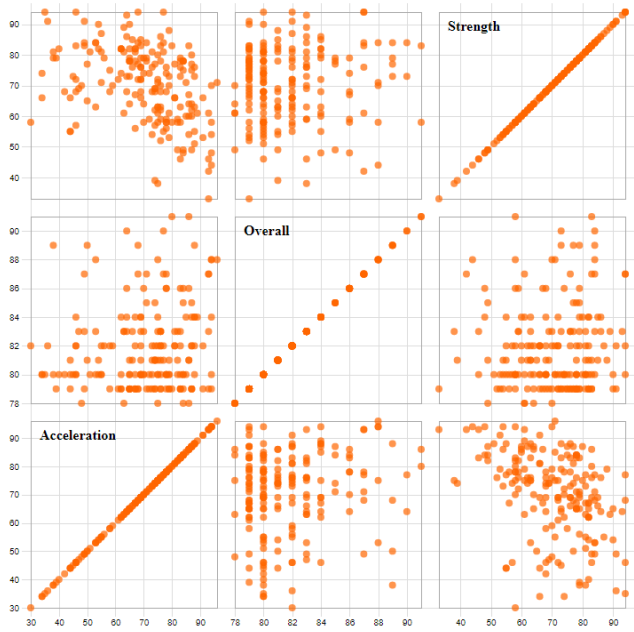
**Original Data:**

We can see that the attributes are correlated to each other.



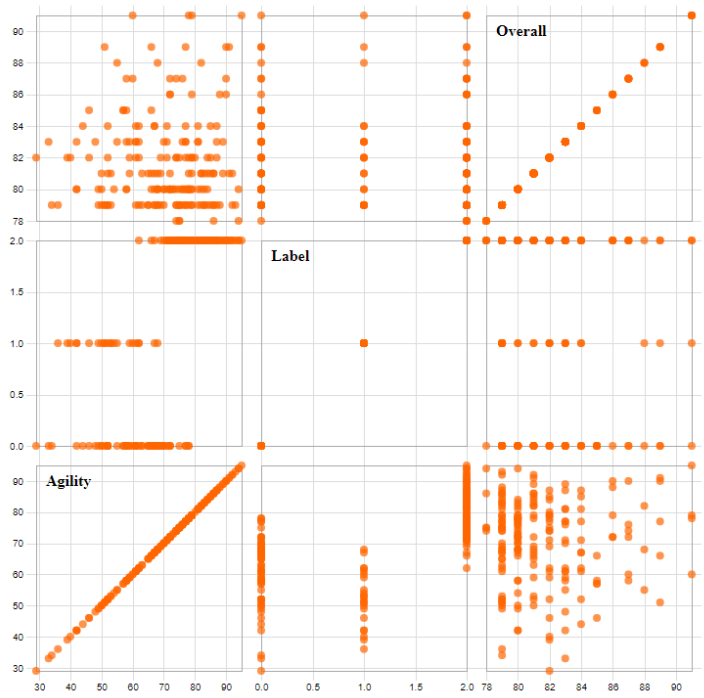

**Random sampling:**

Here correlation between the attributes is not that much clear.




**Adaptive sampling:**

When one attribute increases, the other attribute does not change. There is no correlation between attributes or very less.




Youtube video link:

https://youtu.be/FSWudf1Ss9E

References:

Referenced code template for scatter plot matrix from
https://bl.ocks.org/Fil/6d9de24b31cb870fed2e6178a120b17d