ASSIGNMENT REPORT

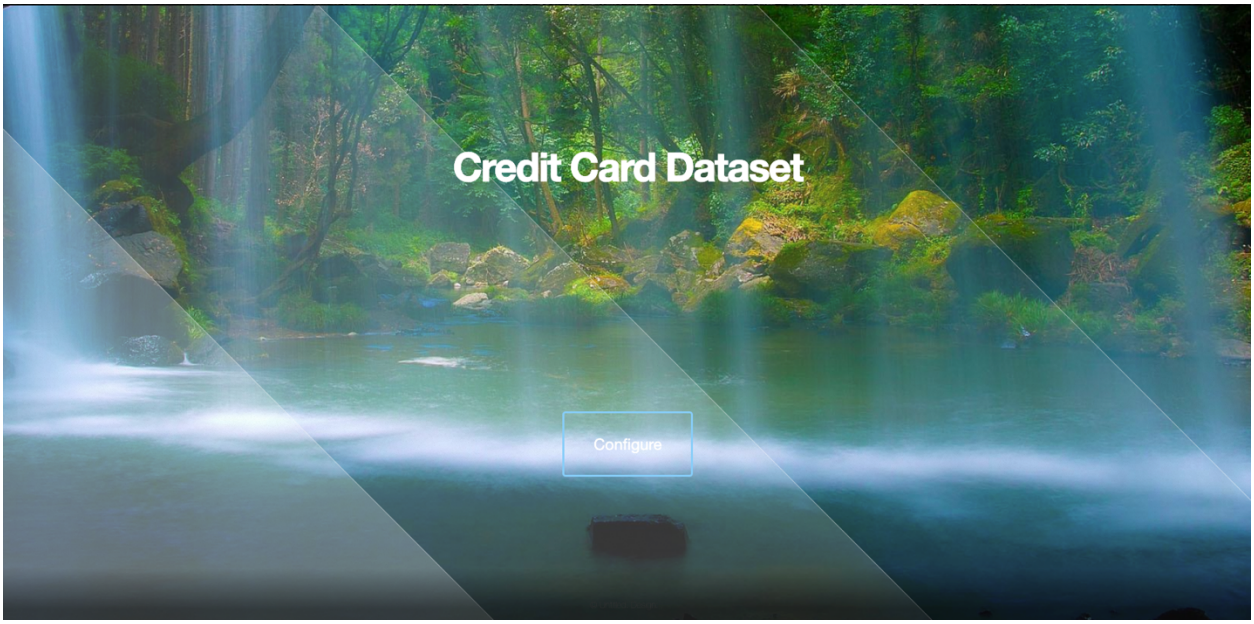
Vishwam Pandya (112669830)

Dataset Used:

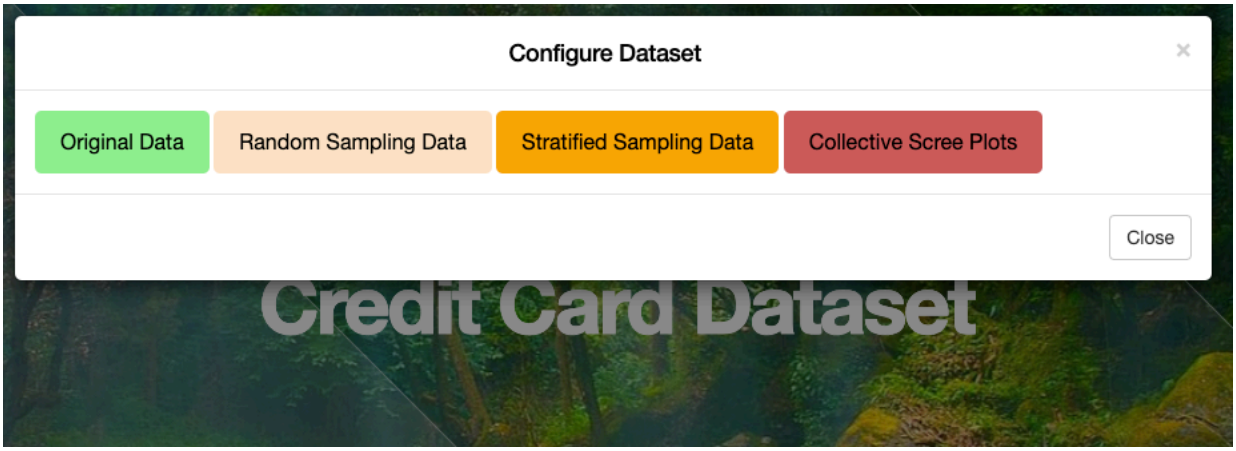
I have visualized data for the Credit Card Dataset from Kaggle.
The data can be visualized in the UI.

Let's start with the steps.

I created a UI for ease of use and understanding. I used Bootstrap "Modal" framework to build this web page from scratch.

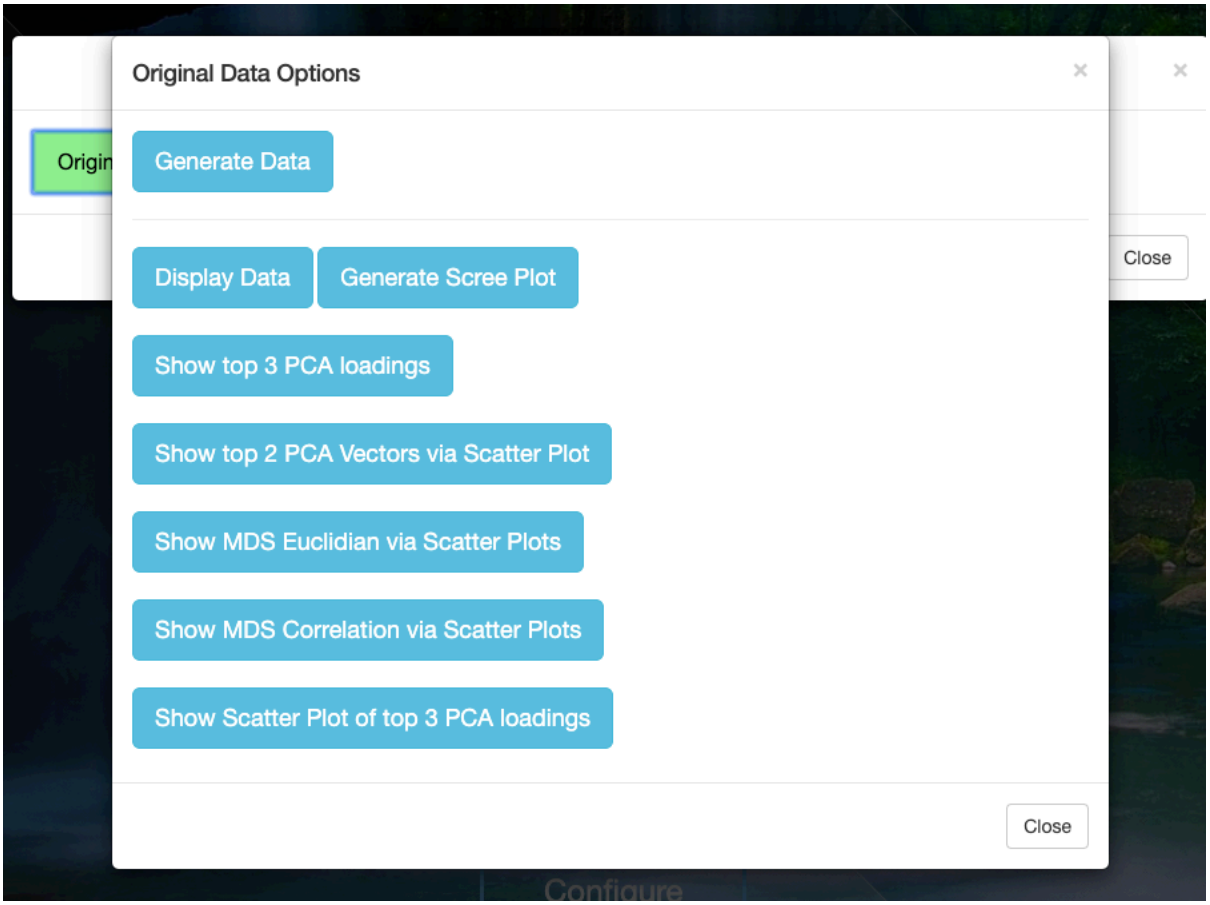


The button shown above allows for the configuration.



We have 4 options there. We have the Original Data, the Random Sampled Data, the Stratified Sampled Data and the Collective Screen Plots. The options are self-explanatory.

The first three options have the following UI screen each.



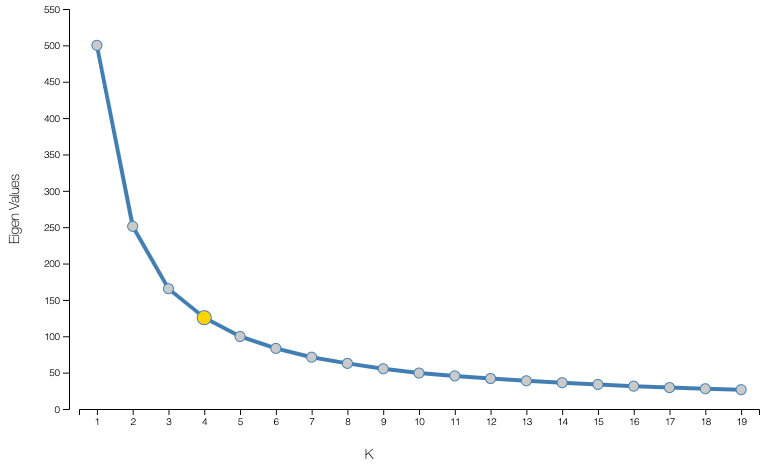
Following is the task of each button here.

- Generate Data: Cleanses the data and generates/parses the data from the Original Data, if Random or Stratified in the backend using Python
- Display Data: Used to Display the Data
- Generate Scree Plot: It will show the Scree plot after calculating the values.
- Top 3 PCA Loadings: List of top 3 PCA Loadings
- Show Top 2 PCA vectors: Scatter Plot to show the top 2 PCA vectors.
- Show MDS Euclidean: Scatter plot to show data via MDS using Euclidean Distance
- Show MDS Correlation: Scatter plot to show data via MDS using Correlation Distance
- Top 3 PCA Loadings via Scatter Plot

All the above options are same, except Stratified has an additional option to calculate value of K.

Visualizations

Finding K using Clustering



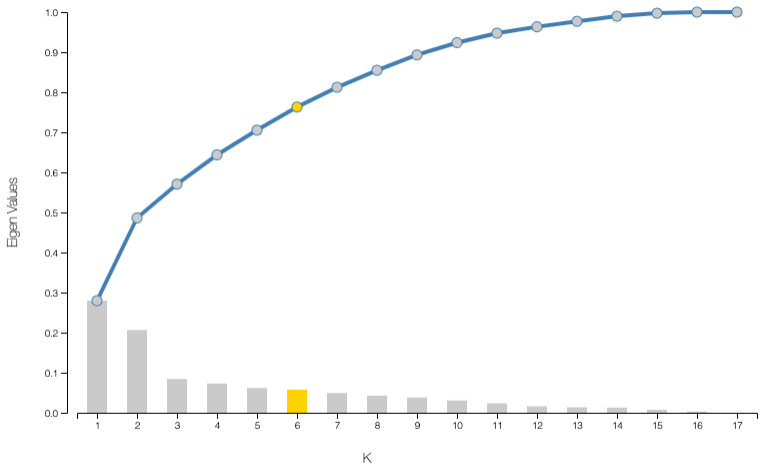
Close

```
def findK():
    df = pd.read_csv('static/data/generated_Stratdata.csv')
    k = range(1, 20)
    clusters = [KMeans(n_clusters=c, init='k-means++').fit(df) for c in k]
    centr_lst = [cc.cluster_centers_ for cc in clusters]

    k_distance = [cdist(df, cent, 'euclidean') for cent in centr_lst]
    distances = [np.min(kd, axis=1) for kd in k_distance]
    avg_within = [np.sum(dist) / df.shape[0] for dist in distances]
    kn = KneeLocator(k, avg_within, curve='convex', direction='decreasing')
    print(kn.knee)
    toBeSaved = []
    for i in range(len(avg_within)):
        toBeSaved.append((i+1,avg_within[i]))
    df = pd.DataFrame(toBeSaved)
    filename = 'static/data/clusters.csv'
    df.columns = ['size', 'value']
    df.to_csv(filename)
    return jsonify(data=str(avg_within), k=str(kn.knee))
```

Scatter Plots:

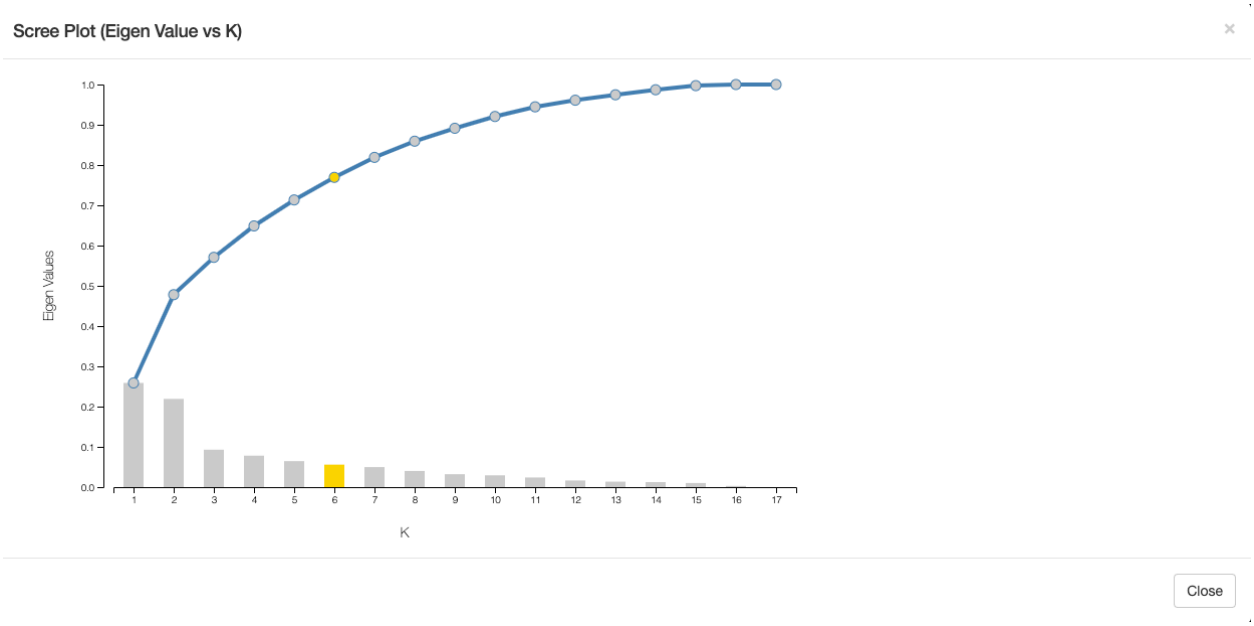
Original Data:

Scree Plot (Eigen Value vs K)

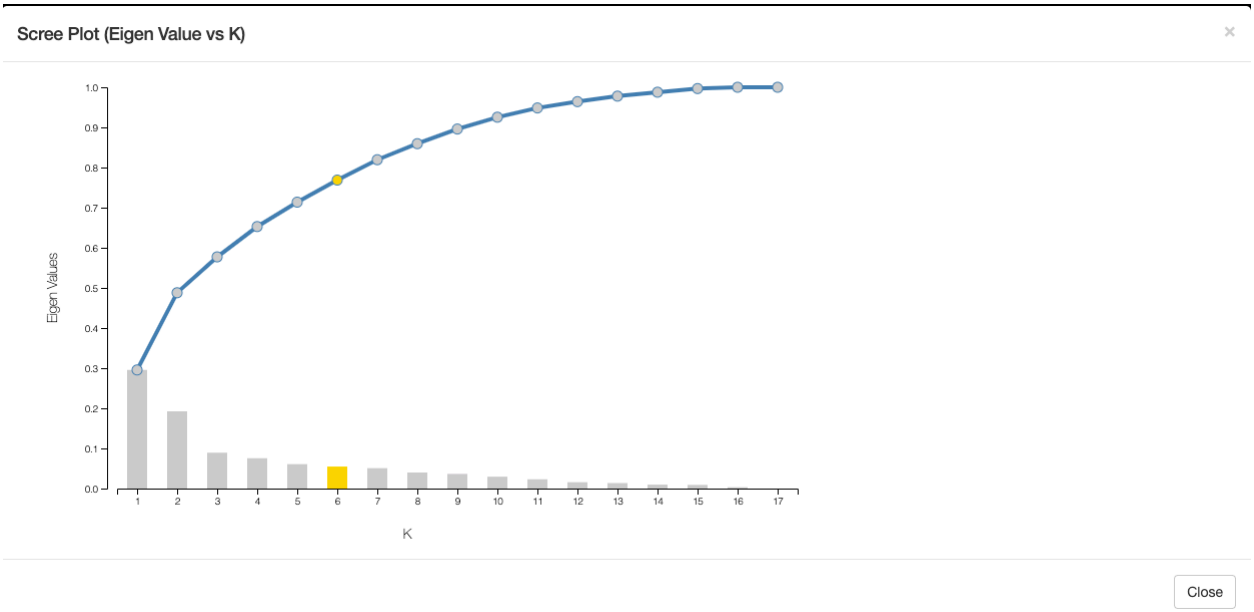


Close

Random Data:



Stratified Data:




```
var circular = graph.selectAll("circle.point1
    .data(data)

circular.enter().append("circle")
    .merge(circular)
    .attr("class", "point1")
    .style("stroke", "steelblue")
    .style("fill", function (d) {
        if (d.size == k) {
            return "#FFD700"
        }
        return "#ccc";
    })
    .attr("transform", "translate(110,0)")
    .attr("cx", function (d, i) {
        return xLine(d.size);
    })
    .attr("cy", function (d) {
        return yLine(d.value);
    })
    .attr("r", function (d) {
        if (d.size == k) {
            return 7;
        }
        return 5;
    })
```

Top 3 PCA Loadings

Original Data:

Top 3 Attributes with Highest PCA Loadings

	TENURE	BALANCE_FREQUENCY	MINIMUM_PAYMENTS
0	0.2893213201637122	-3.0496225061621627	-0.3724002759942053
1	0.2893213201637122	0.4605215426562709	0.8297323649653382
2	0.2893213201637122	0.4605215426562709	-0.16782857492639658
3	0.2893213201637122	0.4605215426562709	1.263524358280422
4	0.2893213201637122	0.4605215426562709	-0.22619411939071096
5	0.2893213201637122	0.4605215426562709	-0.3103137172634341

Random Data:

Top 3 Attributes with Highest PCA Loadings

	TENURE	PRC_FULL_PAYMENT	BALANCE_FREQUENCY
0	0.3606593939956925	-0.5255216098738006	0.5180548788646432
1	0.3606593939956925	-0.5255216098738006	0.5180548788646432
2	-0.3865400449537162	-0.5255216098738006	0.13431716667000934
3	0.3606593939956925	-0.03712026778678576	-0.6331582577192582
4	0.3606593939956925	-0.2406216816929119	-2.5518510398114813
5	0.3606593939956925	-0.5255216098738006	-2.9355887520061152

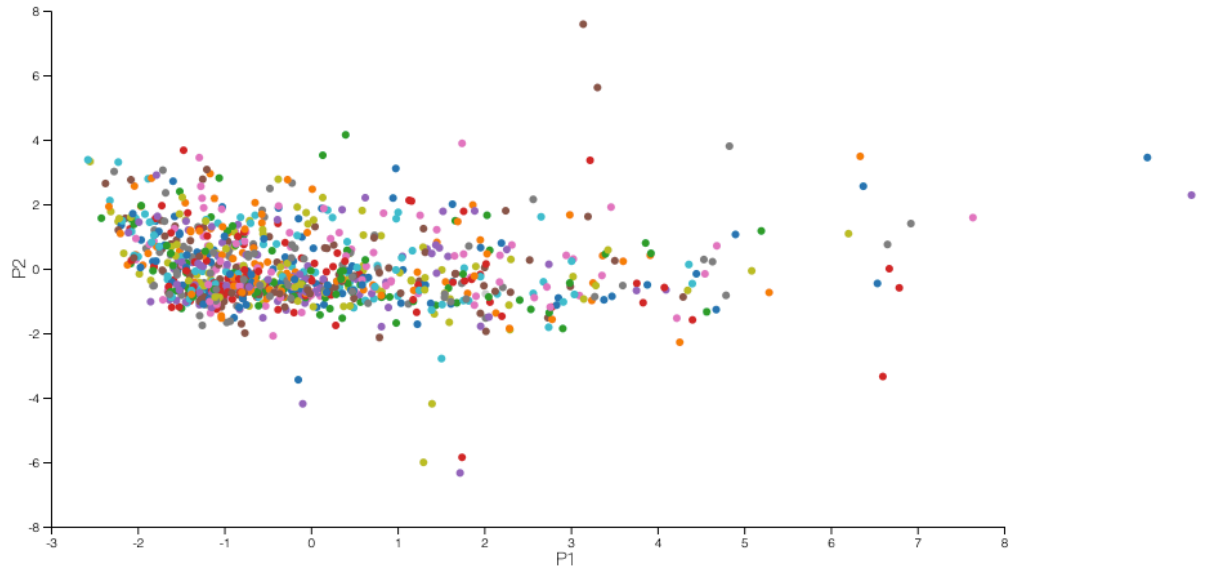
Stratified Data:

Top 3 Attributes with Highest PCA Loadings

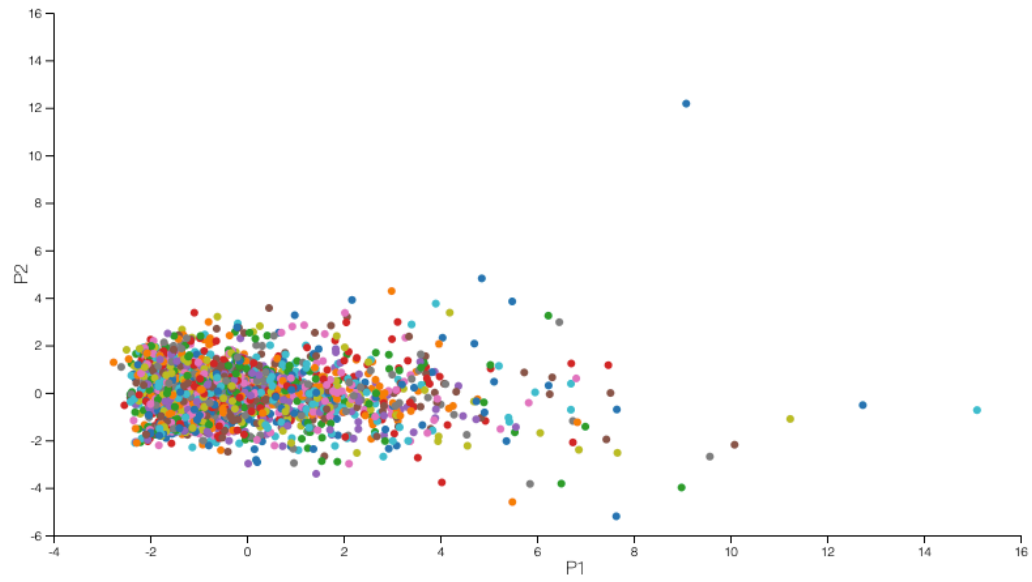
	TENURE	BALANCE_FREQUENCY	MINIMUM_PAYMENTS
0	0.28850291862106736	0.4635733650619195	0.9332237237032944
1	0.28850291862106736	0.4635733650619195	0.1334814923542229
2	0.28850291862106736	0.05245184971416938	0.3922078689502038
3	0.28850291862106736	0.4635733650619195	0.5580543236857413
4	0.28850291862106736	0.4635733650619195	2.0515099457084554
5	0.28850291862106736	0.4635733650619195	0.7235718180984583
6	0.28850291862106736	0.4635733650619195	-0.3582838433025041
7	0.28850291862106736	0.4635733650619195	0.4336238943470505

Top 2 PCA Vectors via Scatter Plot

Original Data:

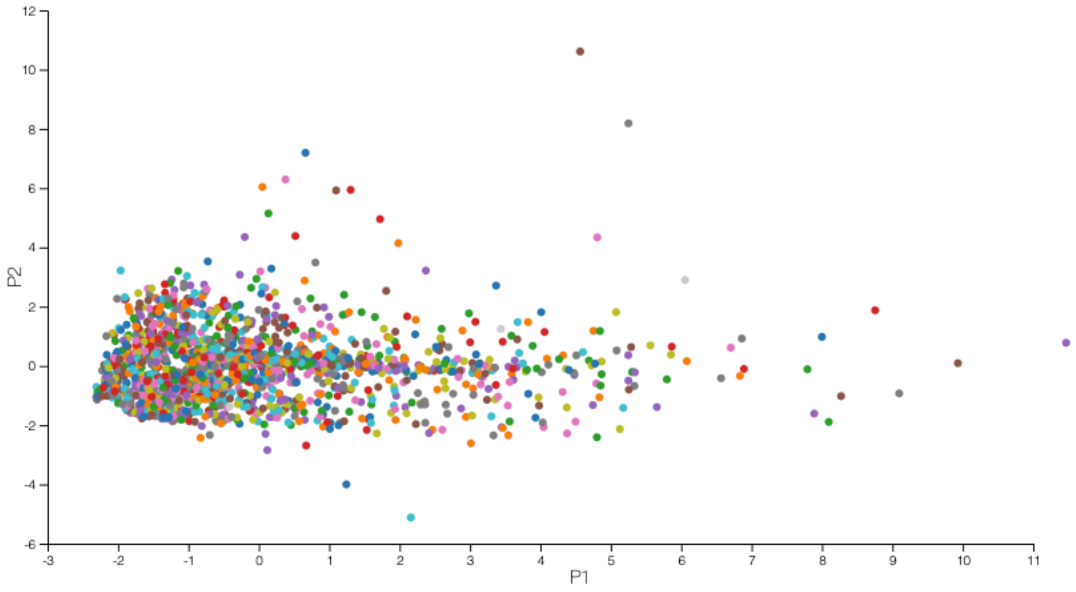


Random Data:



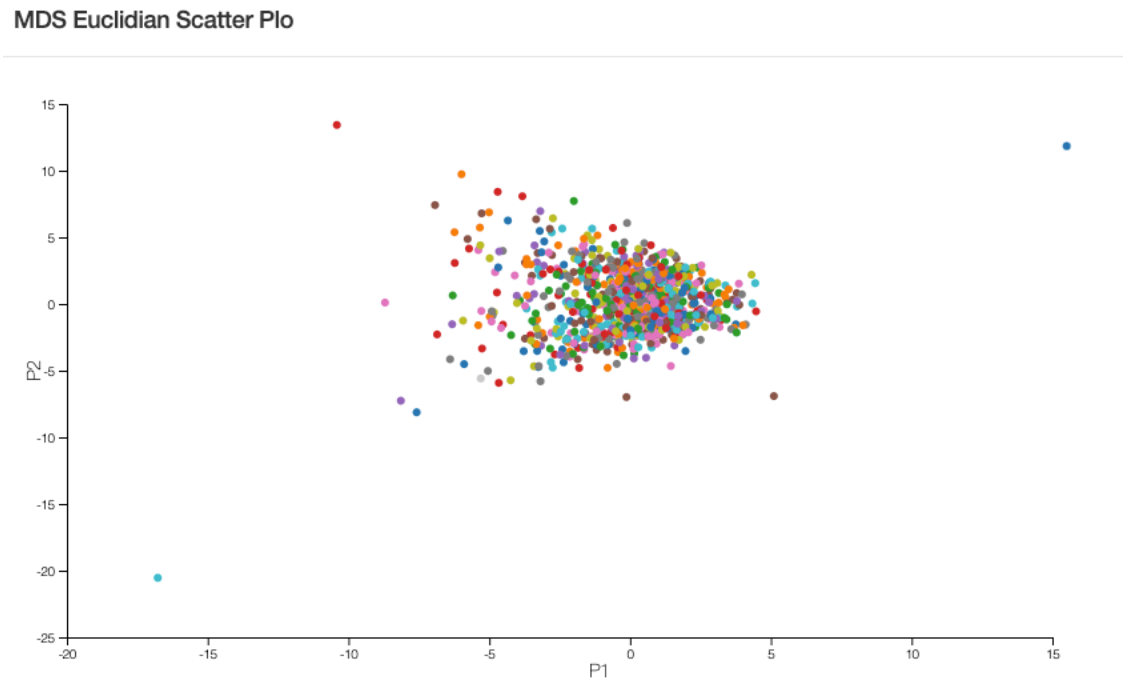
Stratified Data:

Top 2 PCA Scatter Plot Display



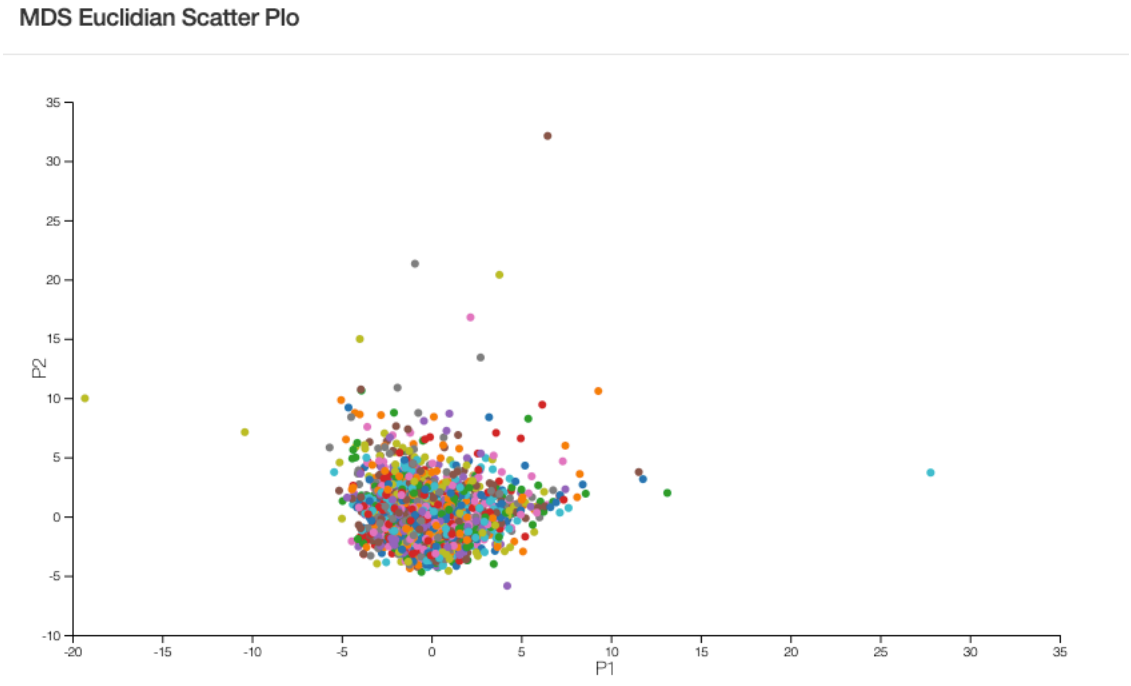

```
// draw dots
svg.selectAll("circle.point1")
    .data(data)
    .enter().append("circle")
    .attr("class", "dot")
    .attr("r", 3)
    .attr("cx", function (d) {
        return xScale(d.P1);
    })
    .style("fill", function (d, i) {
        return color(i);
    })
    .attr("cy", function (d) {
        return yScale(d.P2);
    })
    // .style("fill", function (d, i) {
    //     return color(i)
    // })
    .on("mouseover", function (d) {
        div.transition()
            .duration(500)
            .style("opacity", 0);
        div.transition()
            .duration(200)
            .style("opacity", .9);
        div.html("Point: " +
            parseFloat(d.P1).toFixed( fractionDigits: 2
            .style("left", (d3.event.pageX) + "px")
            .style("top", (d3.event.pageY - 28) + "

        d3.select(this).style("fill", "red");
        d3.select(this).attr("r", 8);
    })
    .on("mouseout", function (d) {
```

MDS Euclidian

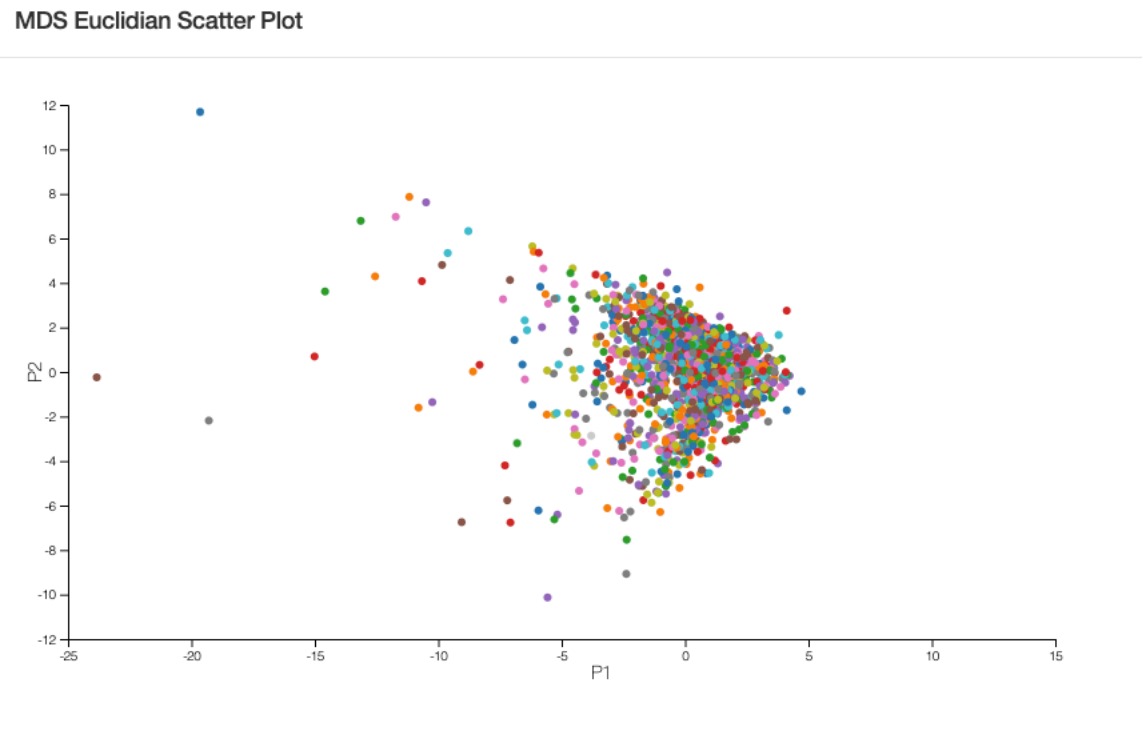


Original:

Random:

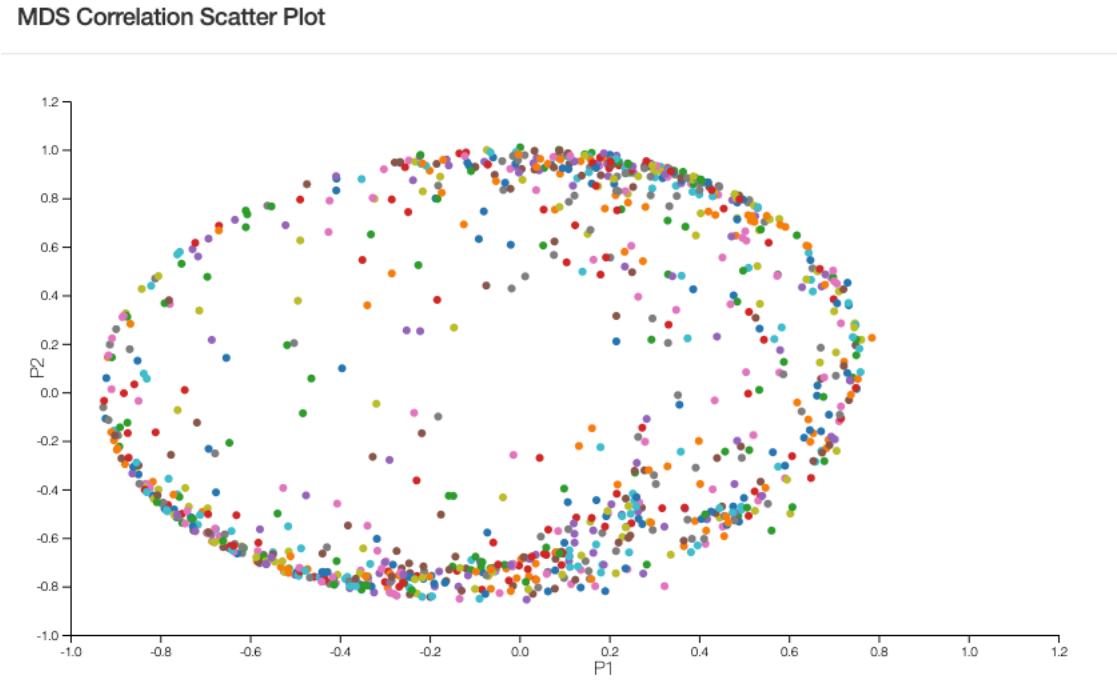


Stratified:



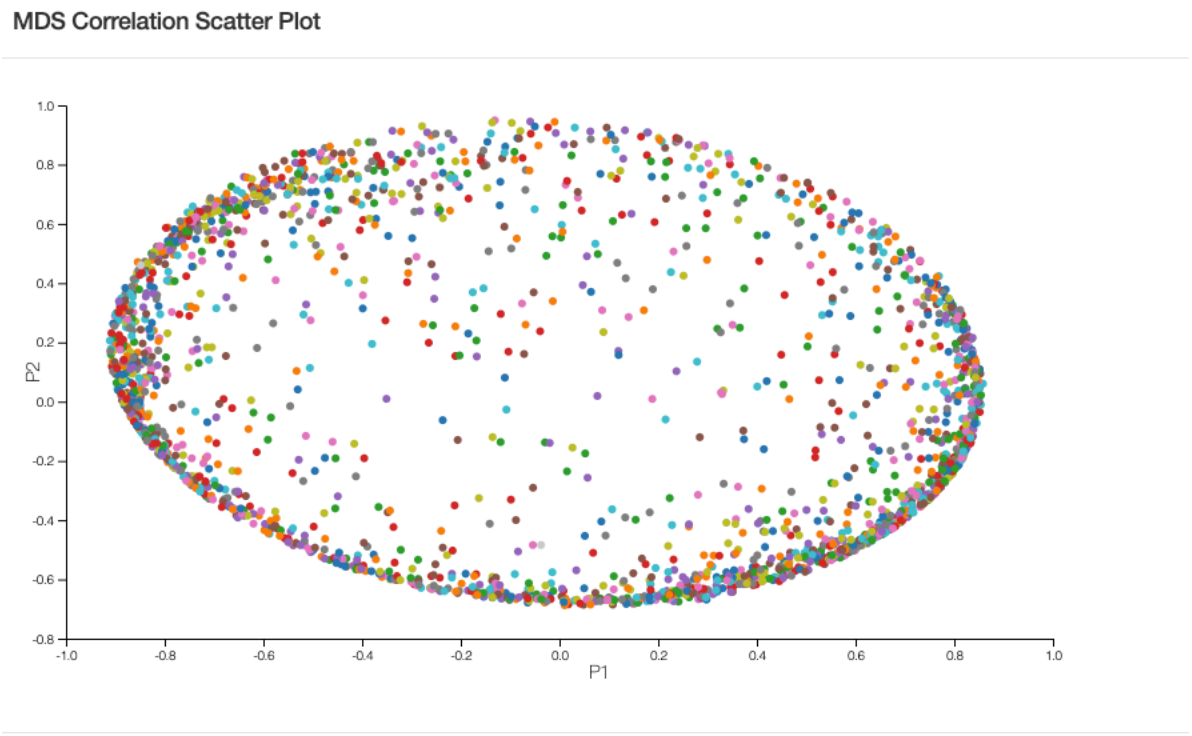

```
# MDS Euclidean
euclidFilename = 'static/data/MDSEuclidean_' + type + '.csv'
if not os.path.isfile(euclidFilename):
    print("MDS Euclid not present")
    mds = MDS(n_components=2, dissimilarity='euclidean')
    df_new = pd.DataFrame(mds.fit_transform(data_frame))
    df_new.columns = ['PC1', 'PC2']
    df_new.to_csv('static/data/MDSEuclidean_' + type + '.csv')
```

MDS Correlation

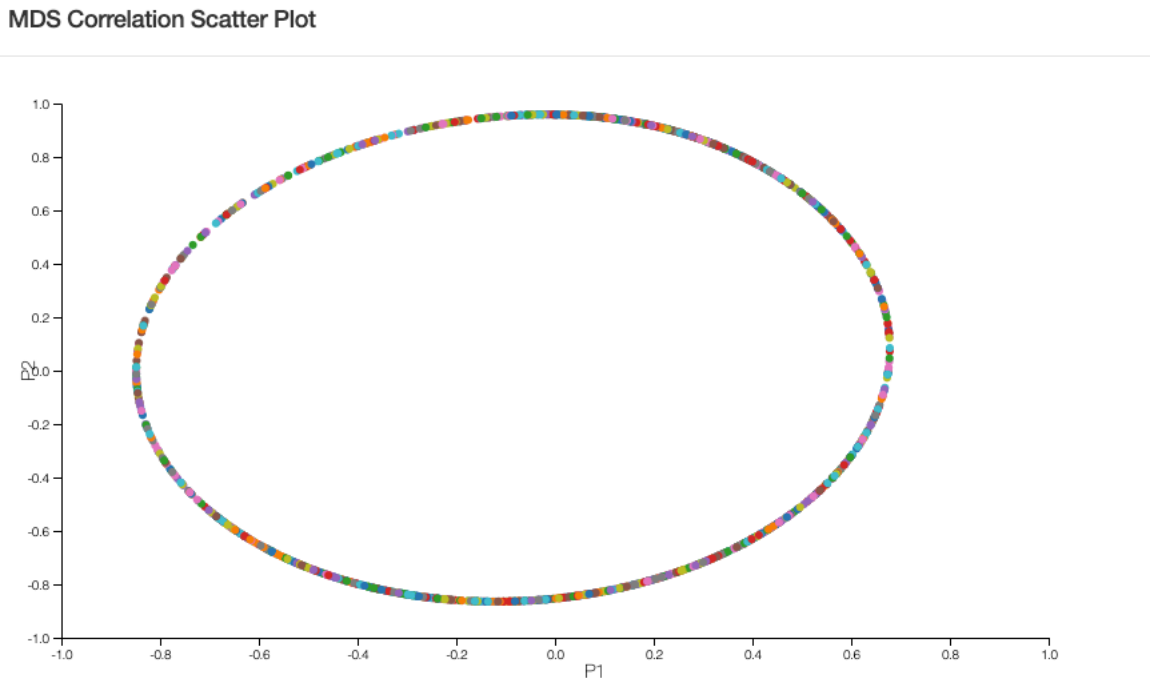


Original: _____

Random:



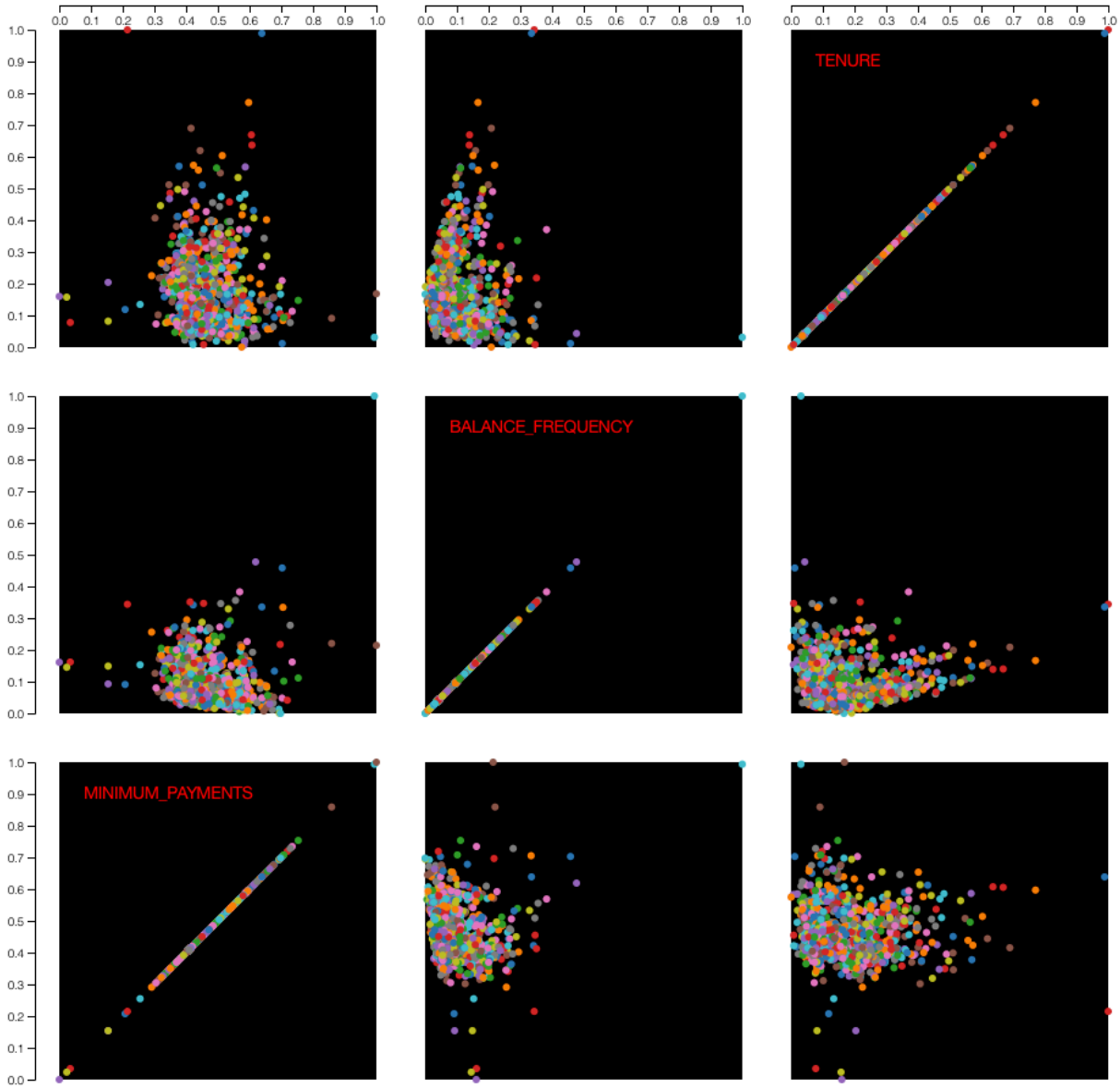
Stratified:



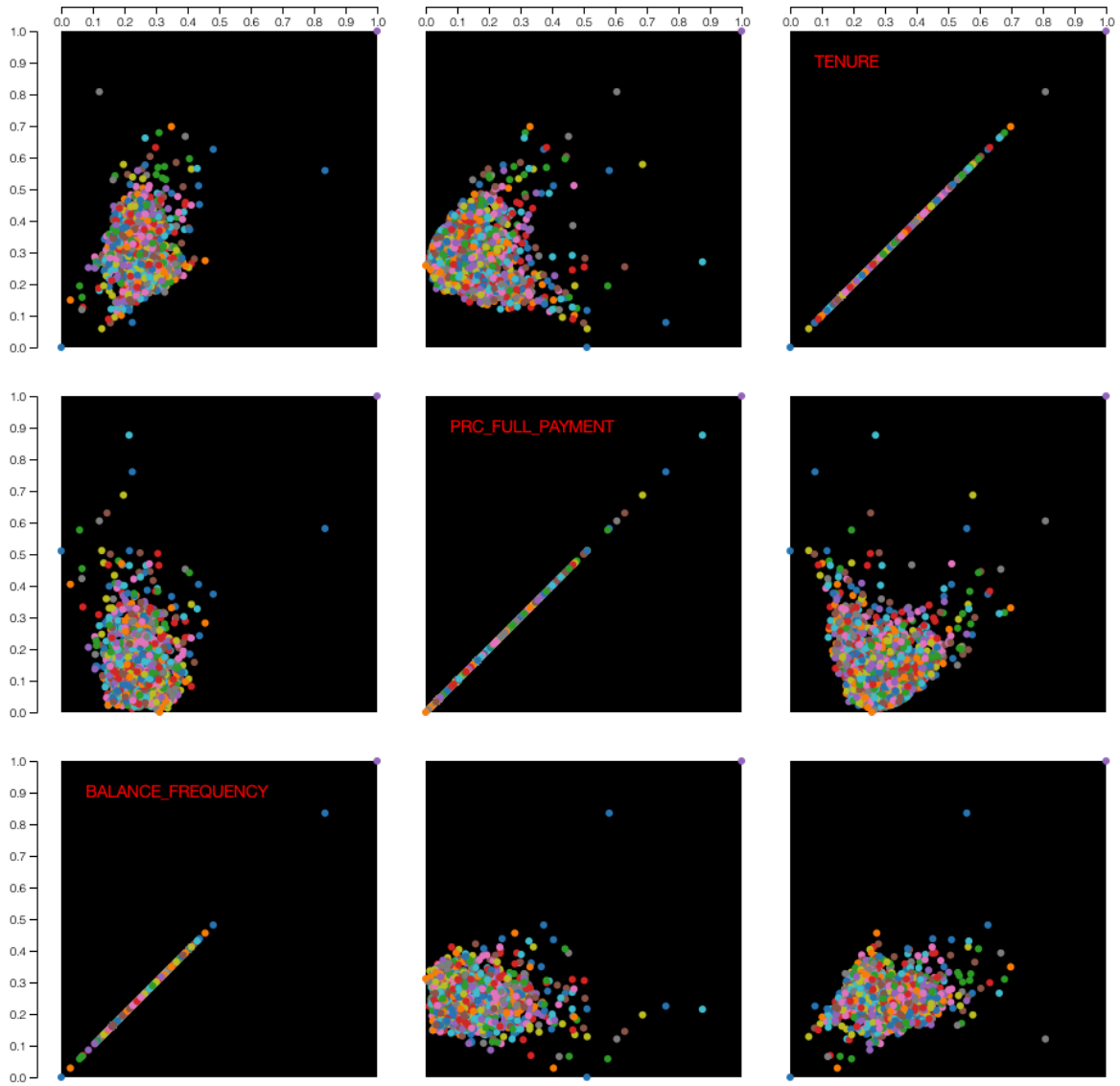

```
# MDS Correlation
corrFilename = 'static/data/MDSCorrelation_' + type + '.csv'
if not os.path.isfile(corrFilename):
    print("MDS Corr not present")
    dis_mat = metrics.pairwise_distances(data_frame, metric='correlation')
    mds = MDS(n_components=2, dissimilarity='precomputed')
    df = pd.DataFrame(mds.fit_transform(dis_mat))
    df.columns = ['PC1', 'PC2']
    df.to_csv('static/data/MDSCorrelation_' + type + '.csv')
```

Top 3 PCA Loadings Scatter Plot

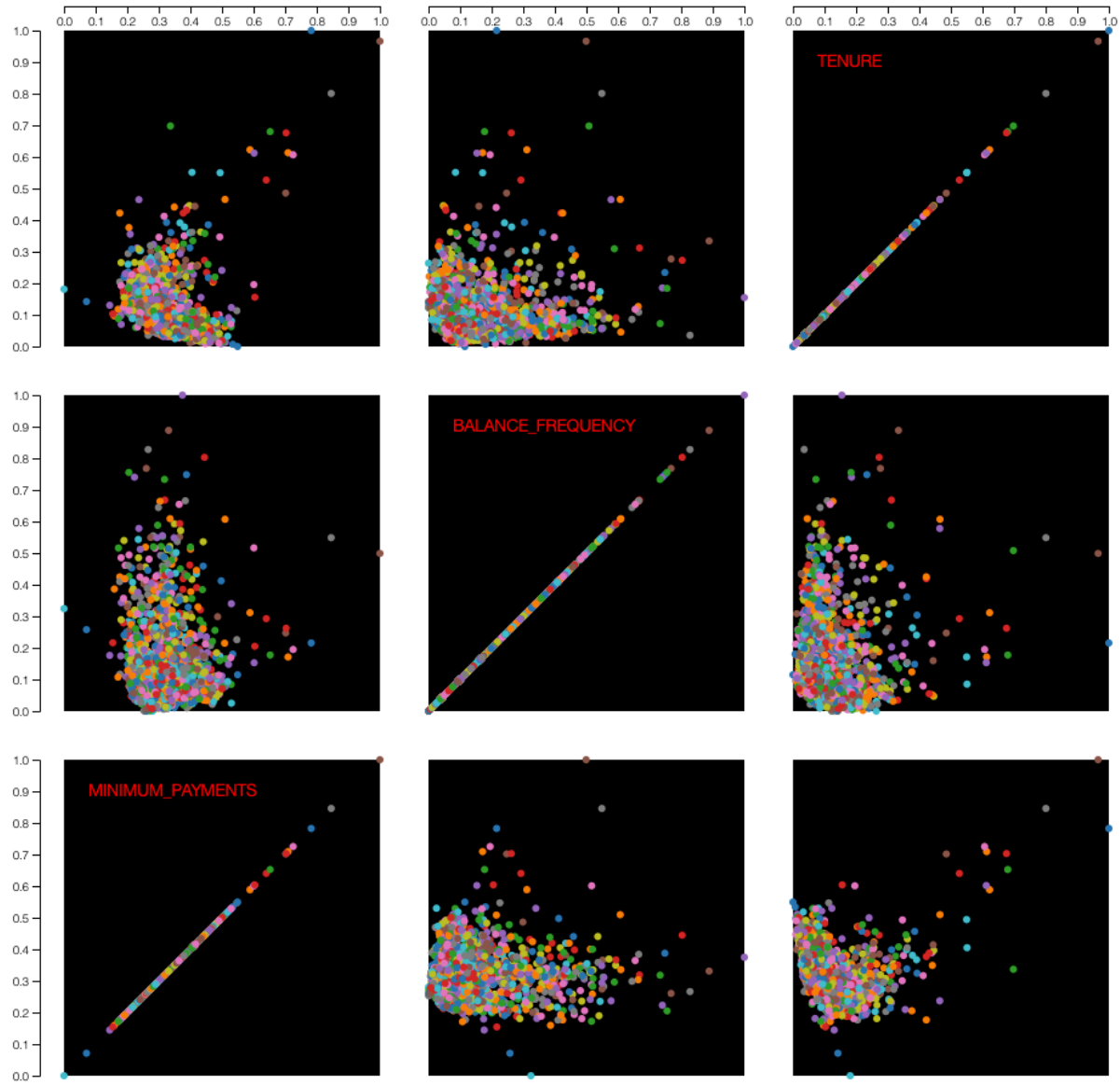
Original:



Random:



Stratified:



Observations:

The data I used is very tightly bounded between its columns. The PC1 and PC2 components when plotted, do not show any visible cluster difference between them.

This result was verified by use of MDS. Both the distance metrics showed a very closed cluster, thereby confirming my previous assumption.

Video Link:

https://youtu.be/HP6HI0QlPaw