Visualization Assignment 2 Report

Name: Ghanendra Piplani

SBU ID: 113070787

Video Link: https://youtu.be/BDsCmpZzWhw

Code GDrive Link:
https://drive.google.com/file/d/1RJRc162BRbToKhZvXr8FD5aKXxekAlVr/view?usp=sharing

Objective:

1. Process a large CSV file to remove empty data and erroneous values.
2. Use python for carrying out random sampling and stratified sampling using K-Means clustering.
3. Use of Python to perform dimension reduction on the dataset by using Principal Component Analysis (PCA) and Multidimensional Scaling (MDS).
4. Use the D3 library to visualize the data generated by Python after applying PCA and MDS algorithms.
5. Using flask create a web application where in data is processed by the Python backend and displayed using the front-end technologies like HTML, CSS, JS.

Dataset: The dataset has been obtained from Kaggle (link). The data has 23 columns, first column is the timestamp in dd-mm-yyyy format and rest 22 columns are numerical in nature and represent the currencies. The dataset tries to capture the trend of 22 world currencies versus the US dollar from the year 2000 to 2019.

Files in ZIP:

1. Dataset present in file goibibo_com-travel_cleaned_3.csv
2. templates/index.html, pca.html, scatter_2d.html, scatter_matrix.html (contain the JS code)
3. app.py (Contains the Flask code)

Libraries used:

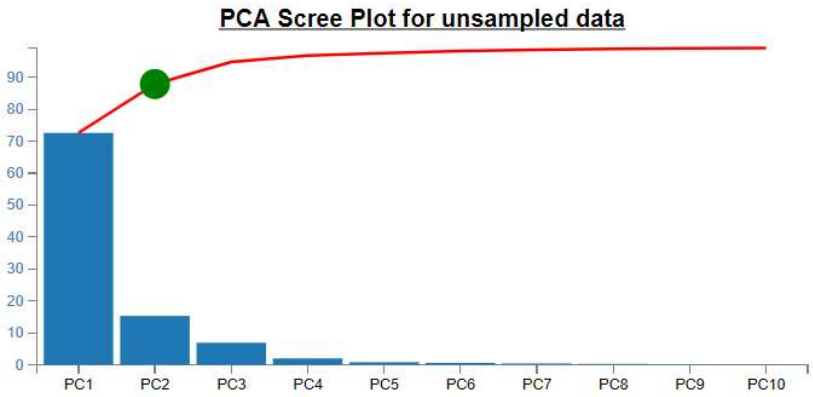
1. d3
2. Flask
3. Sklearn
4. Matplotlib
5. Pandas
6. Numpy

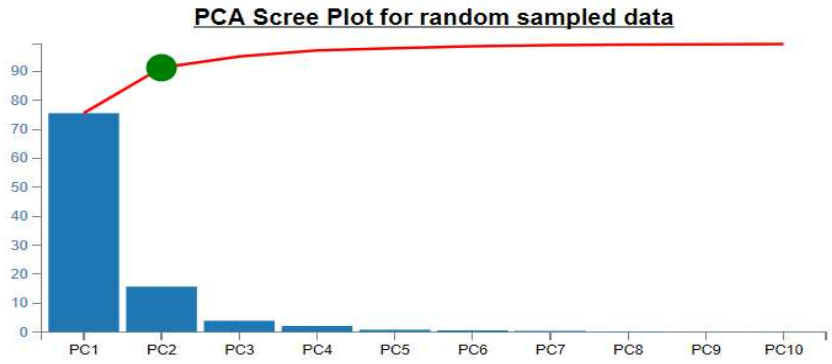
Execution Steps:

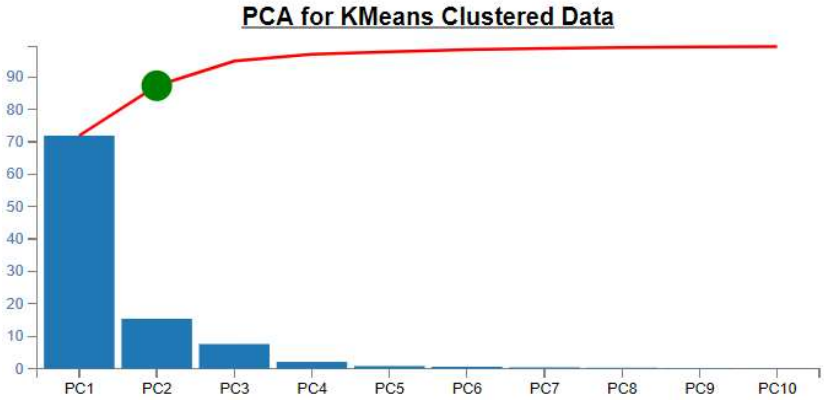
1. Download code from the google drive link if the packages don't work in the version uploaded on blackboard.
2. Open code in ide such as pycharm, go to app.py and click on run.
3. Or from cmd line type python -m app.py to run the code.
4. Use the drop down on the index html page to choose the type of visualization to observe.

Charts and Code Walkthrough:

1. Scree Plot for Unsampled data, Random sampled data and KMeans Clustered Data:

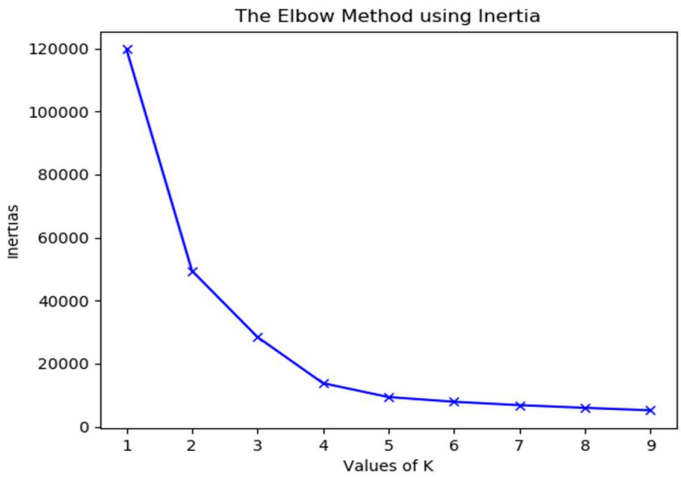







```
def pca(df_=None):
    pca_obj = PCA()
    scaled_data = preprocessing.scale(df_)
    pca_obj.fit(scaled_data)
    per_var = np.round(pca_obj.explained_variance_ratio_ * 100, decimals=1).tolist()[0:10]
    labels = ['PC' + str(x) for x in range(1, 11)]
    sp_df = pd.DataFrame({'label': labels, 'eigenvalue': per_var})
    sp_df['cumulative_eigenvalue'] = sp_df['eigenvalue'].cumsum()
    chart_data = sp_df.to_dict(orient='records')
    chart_data = json.dumps(chart_data)
```

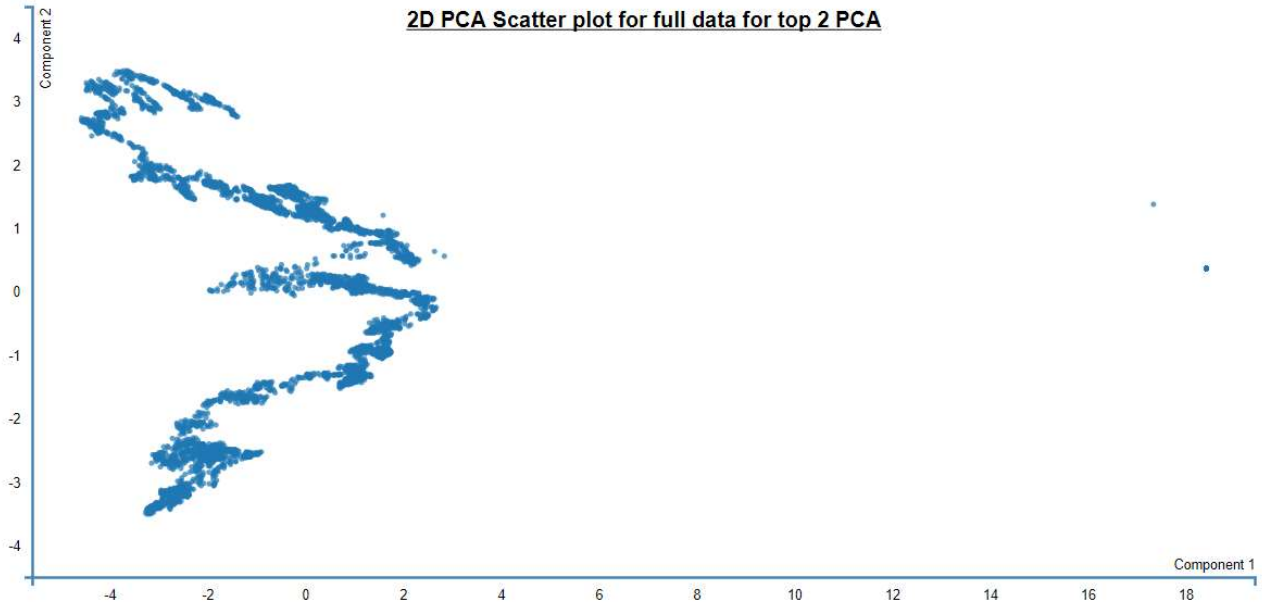
```
    data = {'chart_data': chart_data}
    return data
```



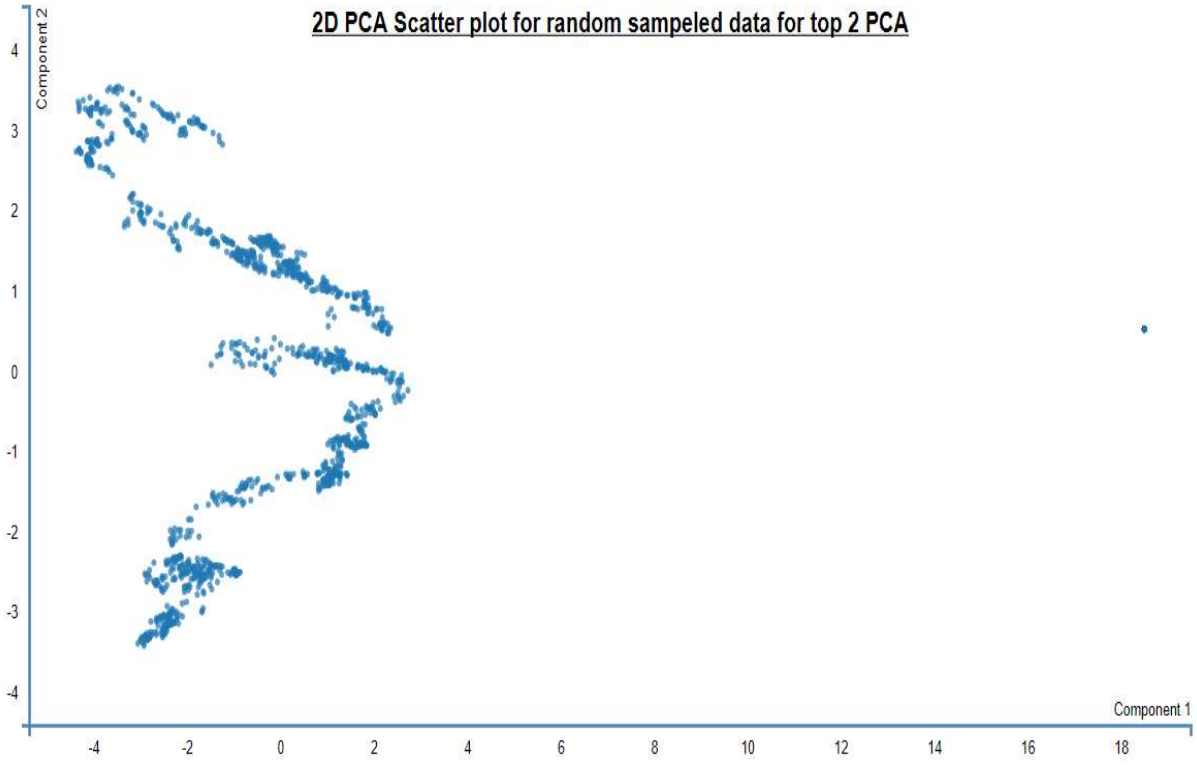
The elbow has been found by applying K-means algorithm, it comes out to be 3. Which has been used for finding the optimal PCA values.

```
for k in k_arr:
    kmeanModel = KMeans(n_clusters=k).fit(x_zip)
    inertia = kmeanModel.inertia_
    inertias.append(inertia)
    mapping_inert[k] = inertia

print("inertias {} ".format(inertias))
plt.plot(k_arr, inertias, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Inertias')
plt.title('The Elbow Method using Inertia')
plt.show()
```

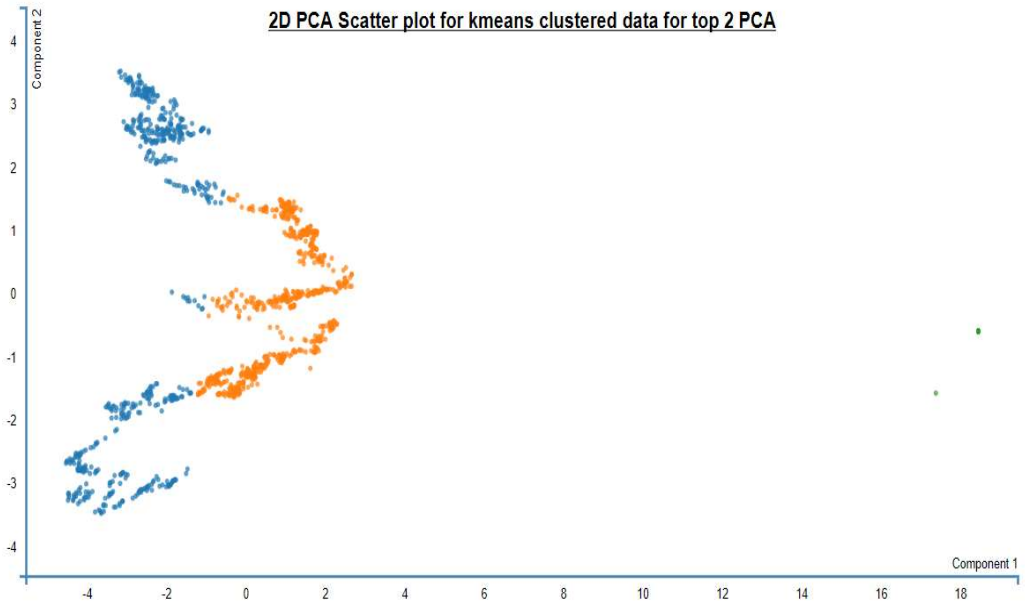
2. The Principal Component Analysis method is used for finding the features of the dataset which heavily influence it. Top 2 PCA have been visualized using scatter plot for the unsampled data, random sampled data and KMeans clustered data.



```
@app.route('/scatter_2d_fulldata')
def scatter_fulldata_plot():
    global df_main
    pca_data = PCA(n_components=2)
    pca_data.fit(df_main)
    df_ = pca_data.transform(df_main)
    data_final = pd.DataFrame(df_)
    loadings = np.sum(np.square(pca_data.components_), axis=0)
    indices = loadings.argsort()[-2:][::-1]
    print(indices)
    for i in range(len(indices)):
        data_final[df_main_random.columns[indices[i]]] = df_main_random[
            df_main_random.columns[indices[i]]]
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D PCA Scatter plot for full data for top 2 PCA"}))
```

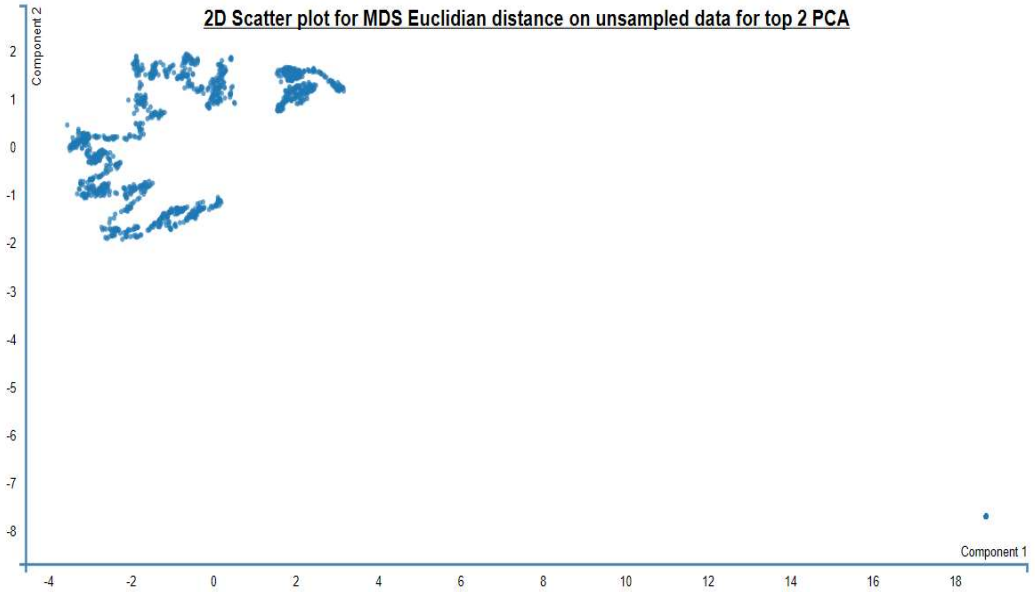



```
@app.route('/scatter_2d_random')
def scatter_random_plot():
    global df_main_random
    pca_data = PCA(n_components=2)
    pca_data.fit(df_main_random)
    df_ = pca_data.transform(df_main_random)
    data_final = pd.DataFrame(df_)
    loadings = np.sum(np.square(pca_data.components_), axis=0)
    indices = loadings.argsort()[-2:][::-1]
    print(indices)
    for i in range(len(indices)):
        data_final[df_main_random.columns[indices[i]]] = df_main_random[
            df_main_random.columns[indices[i]]]
    print(data_final)

    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D PCA Scatter plot for random sampeled data for top 2 PCA"}))
```

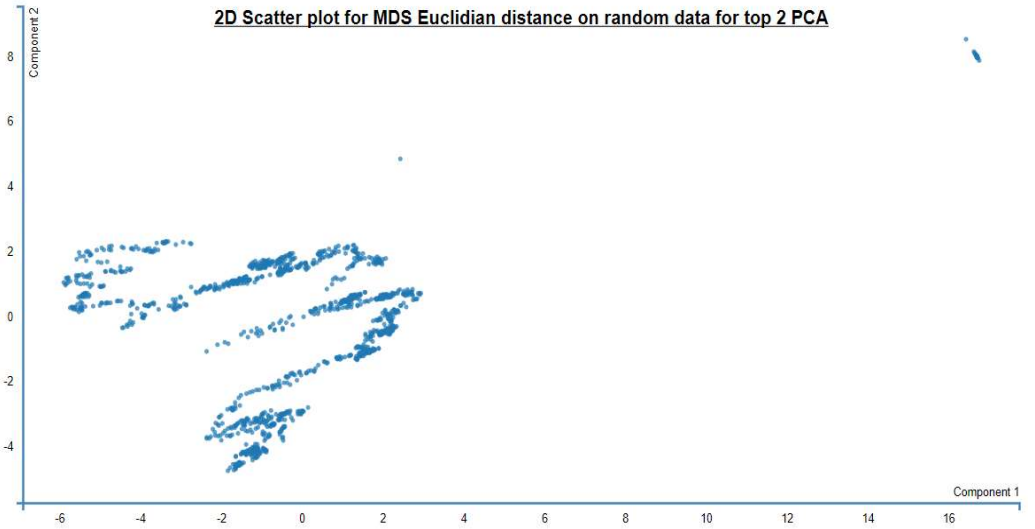



```
@app.route('/scatter_2d_kmeans')
def scatter_kmeans_plot():
    global df_main_kmeans
    pca_data = PCA(n_components=2)
    pca_data.fit(df_main_kmeans)
    df_ = pca_data.transform(df_main_kmeans)
    data_final = pd.DataFrame(df_)

    loadings = np.sum(np.square(pca_data.components_), axis=0)
    indices = loadings.argsort()[-2:][::-1]
    print(indices)
    for i in range(len(indices)):
        data_final[df_main_kmeans.columns[indices[i]]] = df_main_kmeans[
            df_main_kmeans.columns[indices[i]]]
    print(data_final)
    data_final['clusterid'] = np.nan
    x = 0
    for index, row in df_main_kmeans.iterrows():
        data_final['clusterid'][x] = row['cluster_num']
        x = x + 1
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D PCA Scatter plot for kmeans clustered data for top 2 PCA"}))
```

3. The Multidimensional Scaling plots using Euclidean distance for unsampled data, random sampled data and KMeans clustered data.

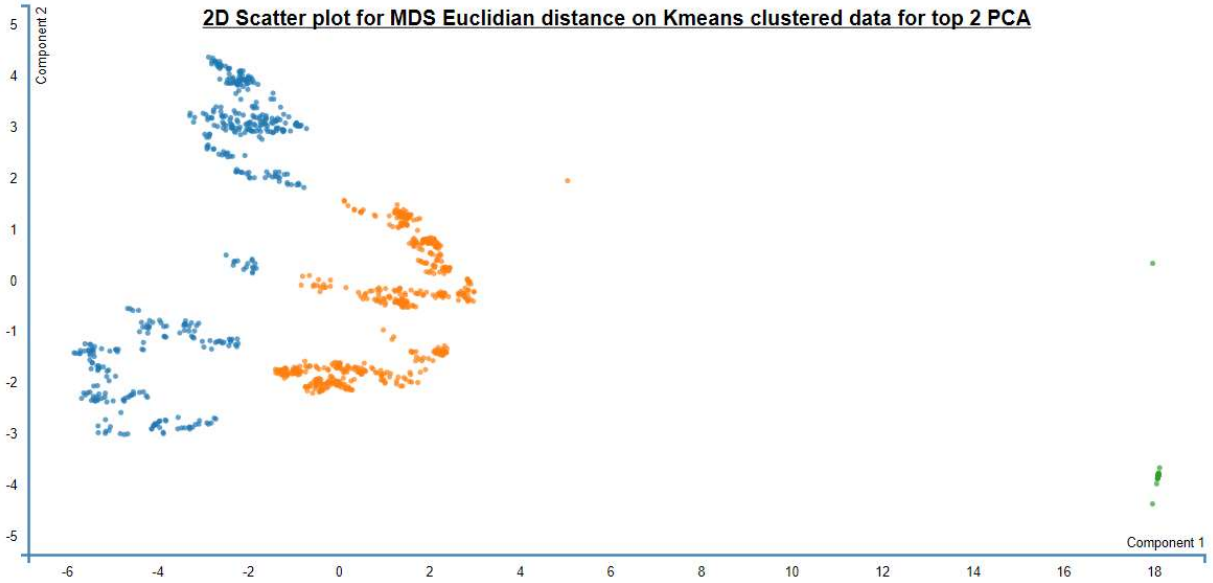



```
@app.route('/mds_euclidean_full')
def mds_euclidean_full():
    global df_main
    df_full_subset = df_main[0:int(0.1 * len(df_main))]
    print("dfdf {}".format(df_full_subset.shape))
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    similarity = pairwise_distances(df_full_subset, metric='euclidean')
    X = mds_data.fit_transform(similarity)
    data_final = pd.DataFrame(X)
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D Scatter plot for MDS Euclidian distance on unsampled data"}))
```

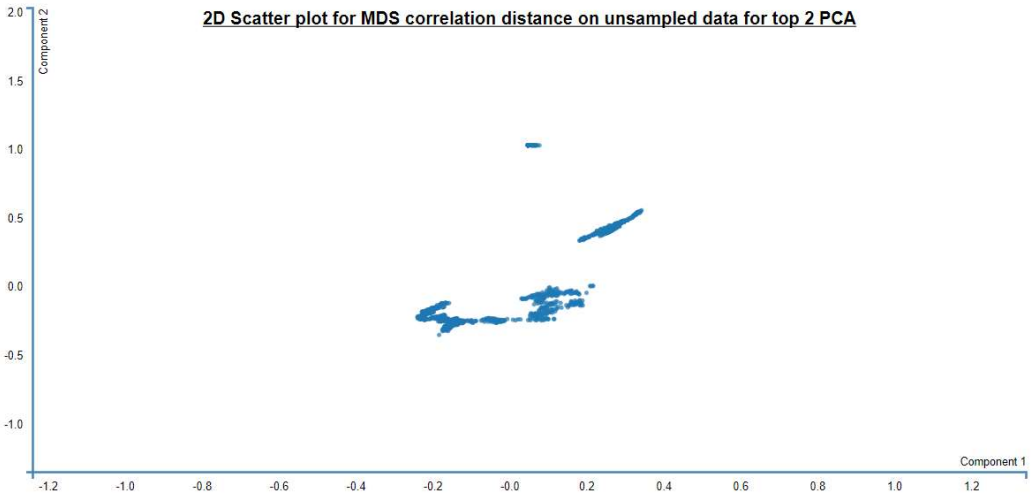



```
@app.route('/mds_euclidean_random')
def mds_euclidean_random():
    global df_main_random
    df_random_subset = df_main_random[0:int(0.1 * len(df_main))]
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    similarity = pairwise_distances(df_random_subset, metric='euclidean')
    X = mds_data.fit_transform(similarity)
    data_final = pd.DataFrame(X)
```

```
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D Scatter plot for MDS Euclidian distance on random data"}))
```

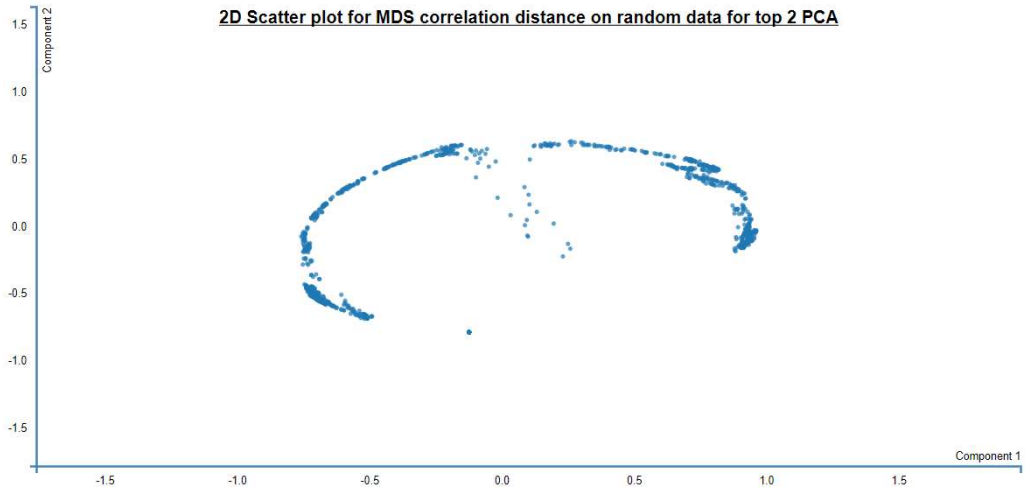



```
@app.route('/mds_euclidean_kmeans')
def mds_euclidean_kmeans():
    global df_main_kmeans
    df_kmeans_subset = df_main_kmeans.sample(int(0.2 * len(df_main)))
    print("dfdf {}".format(df_kmeans_subset.shape))
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    similarity = pairwise_distances(df_kmeans_subset, metric='euclidean')
    X = mds_data.fit_transform(similarity)
    data_final = pd.DataFrame(X)
    data_final['clusterid'] = np.nan
    x = 0
    for index, row in df_kmeans_subset.iterrows():
        data_final['clusterid'][x] = row['cluster_num']
        x = x + 1
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D Scatter plot for MDS Euclidian distance on Kmeans clustered
data"}))
```

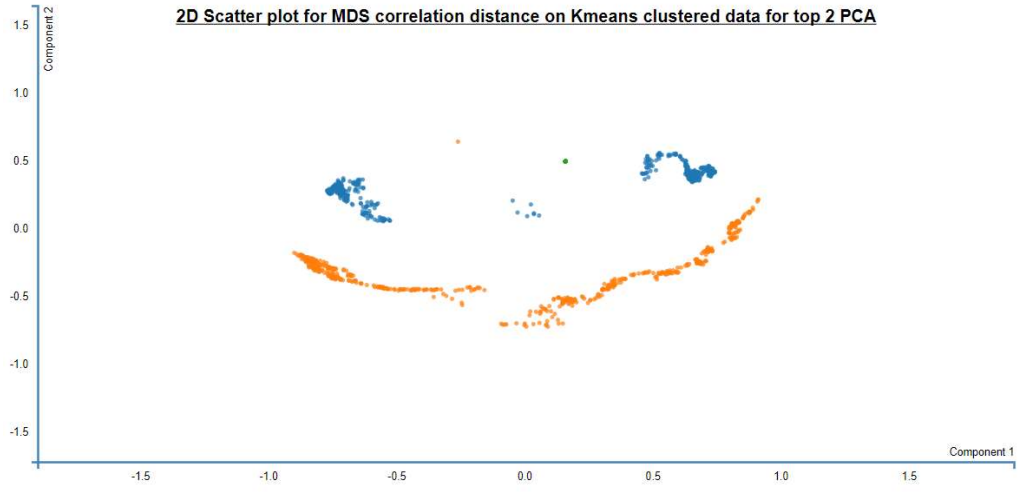



```
@app.route('/mds_correlation_full')
def mds_correlation_full():
    global df_main
```

```
    df_full_subset = df_main[0:int(0.1 * len(df_main))]
    print("dfdf {}".format(df_full_subset.shape))
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    similarity = pairwise_distances(df_full_subset, metric='correlation')
    X = mds_data.fit_transform(similarity)
    data_final = pd.DataFrame(X)
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D Scatter plot for MDS correlation distance on unsampled data"}))
```



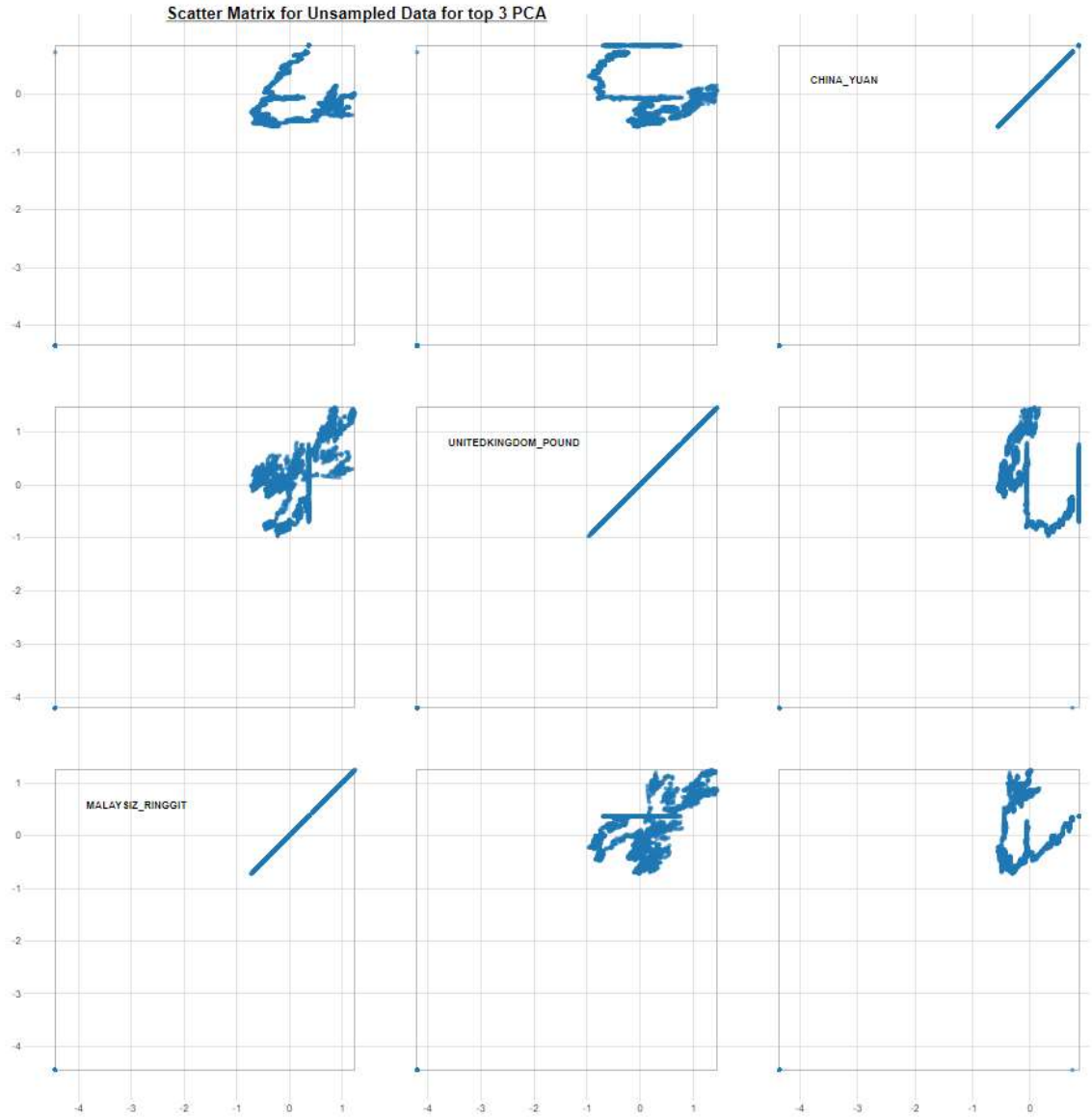

```
@app.route('/mds_correlation_random')
def mds_correlation_random():
    global df_main_random
    df_random_subset = df_main_random[0:int(0.1 * len(df_main))]
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    similarity = pairwise_distances(df_random_subset, metric='correlation')
    X = mds_data.fit_transform(similarity)
    data_final = pd.DataFrame(X)
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D Scatter plot for MDS correlation distance on random data"}))
```




```
@app.route('/mds_correlation_kmeans')
def mds_correlation_kmeans():
    global df_main_kmeans
    df_kmeans_subset = df_main_kmeans.sample(int(0.2 * len(df_main)))
    mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
    similarity = pairwise_distances(df_kmeans_subset, metric='correlation')
    X = mds_data.fit_transform(similarity)
    data_final = pd.DataFrame(X)
```

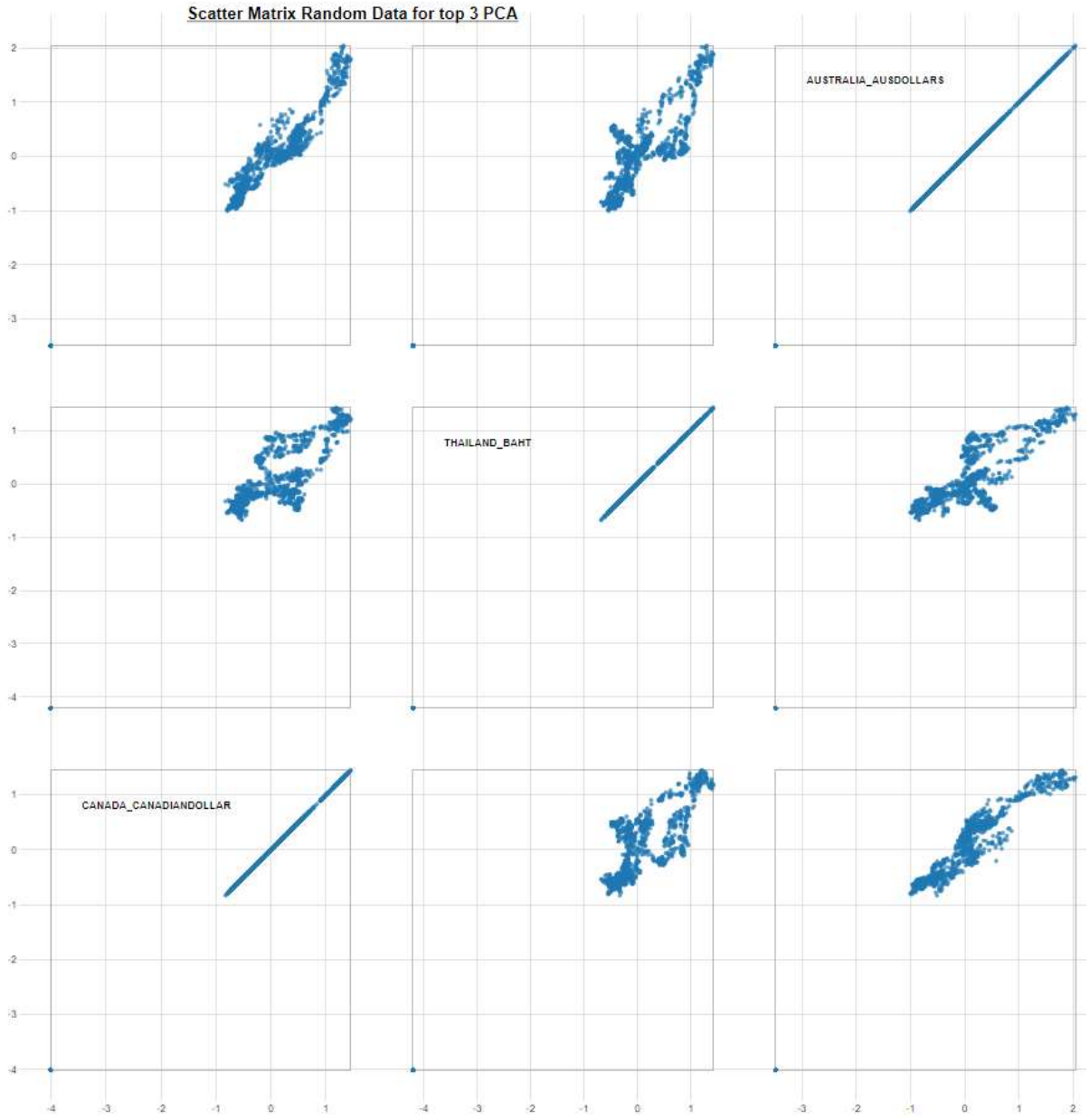
```
    data_final['clusterid'] = np.nan
    x = 0
    for index, row in df_kmeans_subset.iterrows():
        data_final['clusterid'][x] = row['cluster_num']
        x = x + 1
    return render_template("scatter_2d.html", data=json.dumps(data_final.to_dict()),
title=json.dumps({"title":"2D Scatter plot for MDS correlation distance on Kmeans clustered
data"}))
```

4. Next, we visualize the scatter matrix plot for unsampled, random sampled and KMeans clustered data.

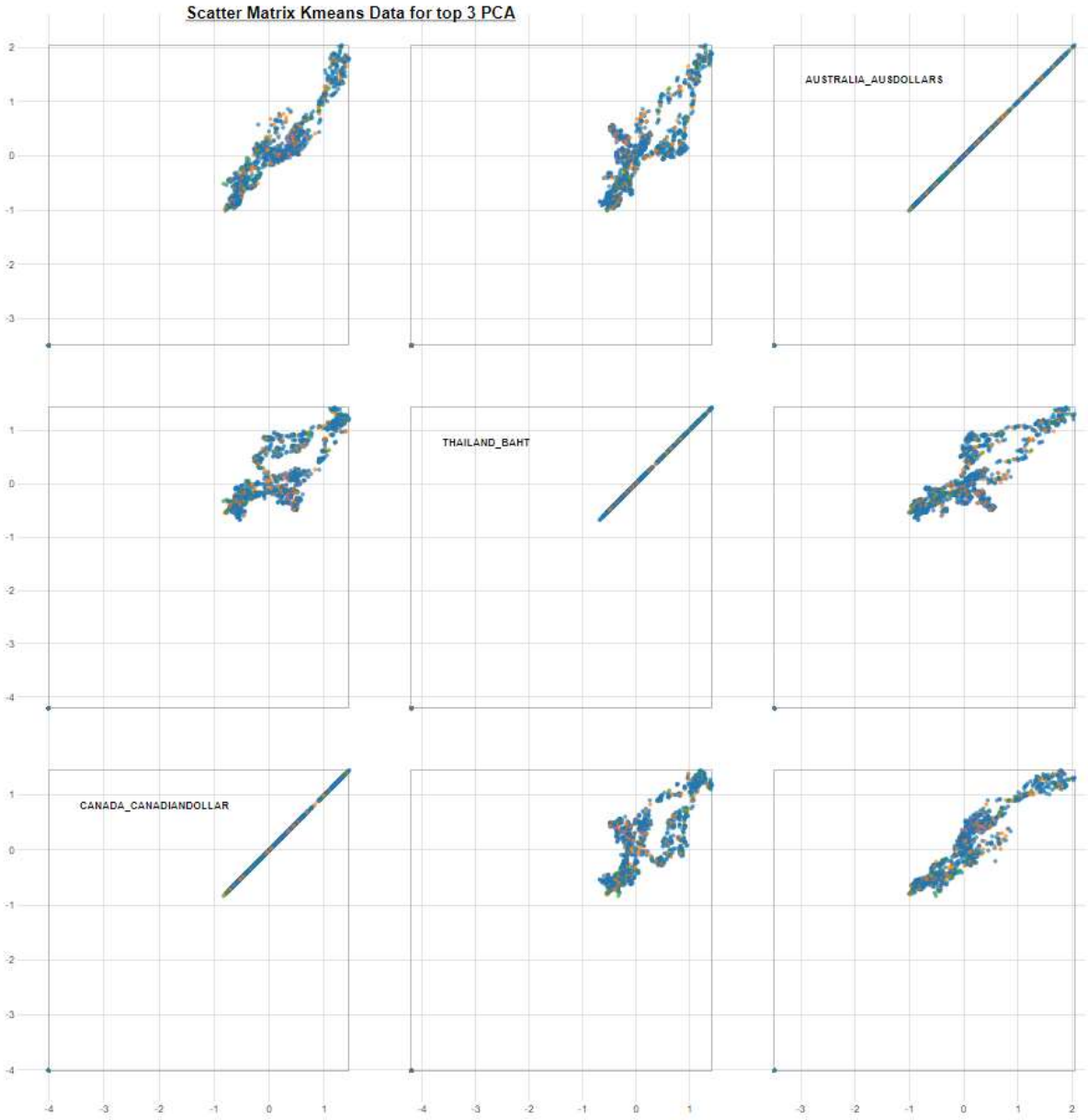



```
@app.route('/scatter_matrix_full')
def scatter_matrix_full():
    global df_main
    data_columns = pd.DataFrame()
    pca = PCA()
    pca.fit(df_main)
    loadings = np.sum(np.square(pca.components_), axis=0)
    indices = loadings.argsort()[-3:][::-1]
    for i in range(len(indices)):
        data_columns[df_main.columns[indices[i]]] = df_main[df_main.columns[indices[i]]]
    return render_template("scatter_matrix.html",
                           rs=json.dumps({'rs': True}),
```

```
                        data=json.dumps(data_columns.to_dict()),  title=json.dumps({"title":
"Scatter Matrix for Unsampled Data"}))
```



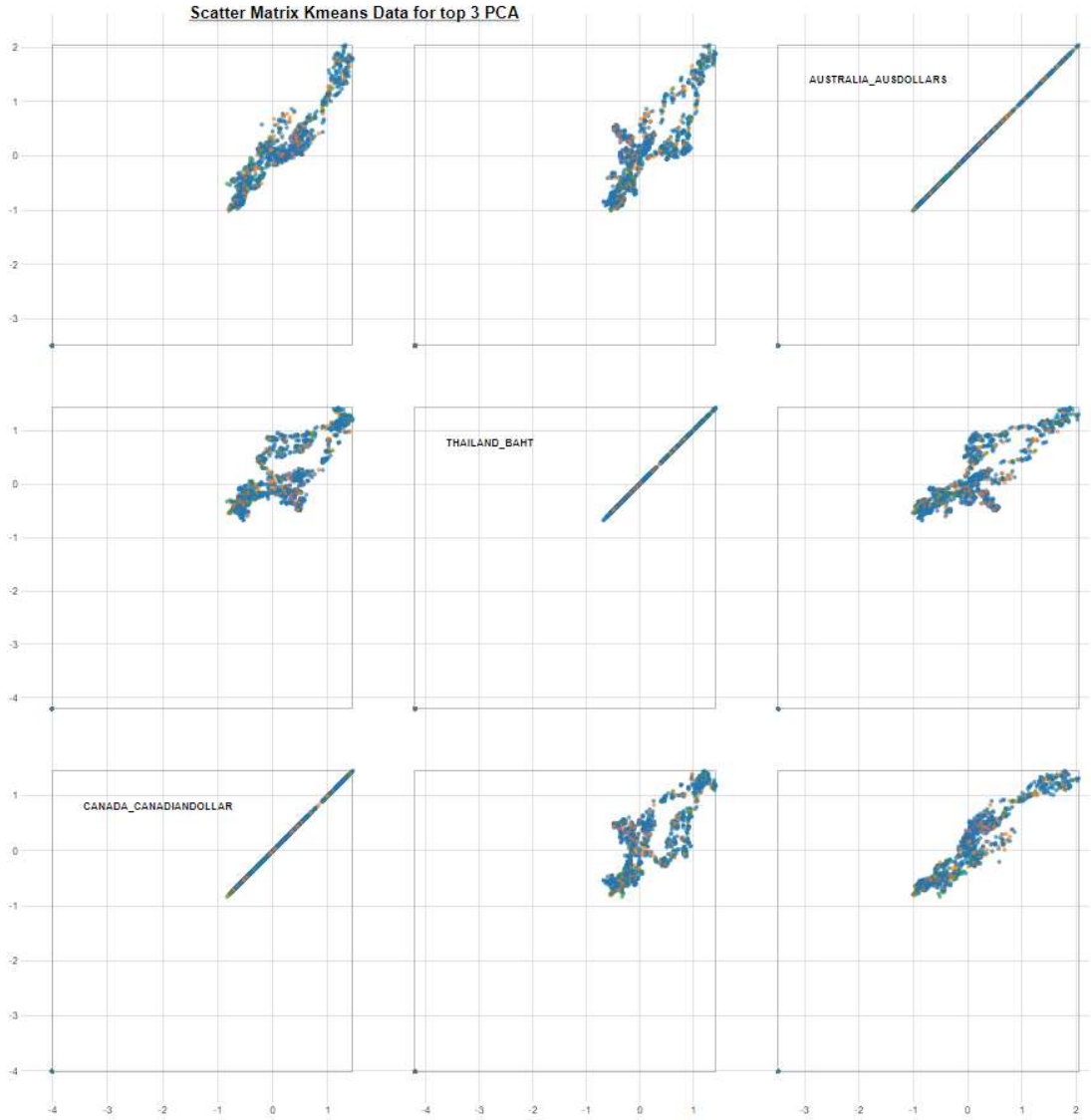

```
@app.route('/scatter_matrix_random')
def scatter_matrix_random():
    global df_main_random
    data_columns = pd.DataFrame()
    pca = PCA()
    pca.fit(df_main_random)
    loadings = np.sum(np.square(pca.components_), axis=0)
    indices = loadings.argsort()[-3:][::-1]
    for i in range(len(indices)):
        data_columns[df_main_random.columns[indices[i]]] =
df_main_random[df_main_random.columns[indices[i]]]
    return render_template("scatter_matrix.html",
                           rs=json.dumps({'rs': True}),
                           data=json.dumps(data_columns.to_dict()),  title=json.dumps({"title":
"Scatter Matrix Random Data"}))
```




```
@app.route('/scatter_matrix_kmeans')
def scatter_matrix_kmeans():
    global df_main_kmeans
    data_columns = pd.DataFrame()
    pca = PCA()
    pca.fit(df_main_random)
    loadings = np.sum(np.square(pca.components_), axis=0)
    indices = loadings.argsort()[-3:][::-1]
    for i in range(len(indices)):
        data_columns[df_main_random.columns[indices[i]]] =
df_main_random[df_main_random.columns[indices[i]]]
    data_columns['clusterid'] = df_main_kmeans['cluster_num']
    return render_template("scatter_matrix.html",
                           rs=json.dumps({'rs': False}),
                           data=json.dumps(data_columns.to_dict()),  title=json.dumps({"title":
"Scatter Matrix Kmeans Data"}))
```

Discussion:

1. **We found the intrinsic dimensionality to be 2 in the PCA scree plot shown by a green marker.**
2. **The elbow was obtained at K=3 using the elbow method for KMeans clustering.**
3. **In the scatter plot for KMeans clustered data, we see clear 3 clusters of blue, yellow and green color.**
4. **There is a difference between the values obtained for the MDS cluster using Euclidean and Correlation distances.**
5. **The data is heavily affected by the PC1 as can be seen by the scatter plots of the PCA and MDS, where in there is heavy**
6. **We also observe that the MDS algorithm takes longer to run than the PCA algorithm on the same data, PCA gives response in a few seconds while MDS can take up to 1 minute in certain cases also.**
7. **From the scatter plots of the PCA of top 2 features, we find that there is a heavy influence of PC1 as the data points in the plot are accumulated around the component 2.**
8. **Upon finding the top 3 PC components we found that the Australian Dollar, Thai Baht and Canadian Dollar are the 3 columns which heavily influence the scatter matrix plot shown below:**



9. **If we see further analyse, we find that the that these 3 currencies have shown highest variance with respect to the US dollar in past 20 years, hence this could be a reason for this heavy correlation between these three as can be seen from the above matrix.**