CSE 564 - Spring ’21 - Lab 2

Open Asteroid Dataset Visualization

Nikhil Rao Sira

SBU ID : 113273807

1. Data

This dataset is an extended version of the dataset used in lab 1. It has more data points. For the plots in lab 2, I have filtered data to consider numerical variables. The data comprises of **17 numerical** and **6000 data points** taken from Kaggle. (Although, a more extensive dataset is available on the JPU website).

2. Data preparation for Tasks 1-4

The data was fused, cleaned and preprocessed using a Python Library - **Pandas**

A. Performed PCA on data to generate Eigen Vectors, Eigen Values and ratio of variance

```
[54]: normalize_columns = ['sm_axis','eccentricity','mag_slope','incline','longitude','argument_of_perihelion','perihelion_distance','aphelion_distance','orbital_period','data_arc','no_obs_used','absolute_magnitude_parameter','albed
for column in normalize_columns:
    df[column] = (df[column] - df[column].min()) / (df[column].max() - df[column].min())
```

```
[56]: df
```

[56]:

	full_name	sm_axis	eccentricity	mag_slope	incline	longitude	argument_of_perihelion	perihelion_distance	aphelion_distance	orbital_period	...	absolute_magnitude_parameter	albedo	rotation_period	bv_mag_difference	ub_mag_difference	taxonomic_type
0	1 Ceres	0.349487	0.189621	0.315789	0.296097	0.223342	0.203663	0.430278	0.285615	0.281099	...	0.014862	0.075203	0.005427	0.680993	0.650382	C
1	2 Pallas	0.350367	0.588293	0.298246	1.000000	0.481557	0.861848	0.302036	0.388576	0.281913	...	0.098726	0.086382	0.004347	0.606495	0.433588	B
2	3 Juno	0.322826	0.657022	0.666667	0.365635	0.472574	0.689515	0.256593	0.375187	0.256677	...	0.226115	0.201220	0.003830	0.787011	0.661069	Sk
3	4 Vesta	0.240794	0.222461	0.666667	0.195855	0.288763	0.418365	0.307483	0.188071	0.184417	...	0.000000	0.413415	0.002230	0.746896	0.751145	V
4	5 Astraea	0.297528	0.486919	0.105263	0.144322	0.393875	0.997239	0.286477	0.306265	0.233923	...	0.387473	0.262195	0.012051	0.788921	0.627481	S
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
594	595 Polyxena (1906 TZ)	0.466738	0.156397	0.105263	0.506066	0.066343	0.776246	0.565426	0.388717	0.393684	...	0.498938	0.131098	0.007763	0.000000	0.000000	0
595	596 Scheila (1906 UA)	0.392027	0.415174	0.105263	0.414198	0.196339	0.486805	0.397597	0.387624	0.320986	...	0.605096	0.024390	0.011231	0.681948	0.270229	T
596	597 Bandusia (1906 UB)	0.323181	0.368465	0.105263	0.360281	0.101517	0.854576	0.346942	0.304396	0.257000	...	0.636943	0.223679	0.004219	0.000000	0.000000	S
597	598 Octavia (1906 UC)	0.347407	0.639025	0.105263	0.343546	0.254572	0.812039	0.283085	0.398258	0.279178	...	0.671975	0.028455	0.006983	0.711557	0.575573	X
598	599 Luisa (1906 UJ)	0.350738	0.747151	0.105263	0.473612	0.122798	0.817926	0.250866	0.429694	0.282256	...	0.584926	0.100610	0.005849	0.839542	0.680916	K

599 rows × 23 columns

B. Function to get top 4 attributes on selecting intrinsic dimensionality:

```
def get_top_4_loading(n,pca_components):
    ss_list = []
    for j in range(len(pca_components)):
        sum = 0.0
        for i in range(n):
            sum = sum + (pca_components[i][j])**2
        ss_list.append(math.sqrt(sum))

    ss_list = np.asarray(ss_list, dtype=np.float64)
    indices = ss_list.argsort()[-4:][::-1]
    return ss_list, indices
```

C. Function to get Bi-plot data - Data Points and Dimensions

```python
def getbiplotdata(df_lg, pca_components):
    df_lg_np = df_lg.to_numpy()
    x_updated = []
    y_updated = []
    for act_val in df_lg_np:
        sum = 0
        i=0
        for pc_val in pca_components[0]:
            sum = sum + act_val[i]*pc_val
            i = i+1
        x_updated.append(sum)

    for act_val in df_lg_np:
        sum = 0
        i=0
        for pc_val in pca_components[1]:
            sum = sum + act_val[i]*pc_val
            i = i+1
        y_updated.append(sum)
    pca_biplot_points = []
    for i in range(len(x_updated)):
        pca_biplot_points.append({'PC1':x_updated[i],'PC2':y_updated[i]})
    coordinate_list = []
    for x in range(2):
        l1=[]
        for y in range(len(pca_components[x])):
            l1.append(pca_components[x][y])
        coordinate_list.append(l1)
    pca_line_data = []
    print(coordinate_list)
    for i in range(len(coordinate_list[0])):
        pca_line_data.append({'X':coordinate_list[0][i],'Y':coordinate_list[1][i]})
    print(pca_line_data)
    return pca_biplot_points, pca_line_data
```

D. Ran MDS on data to generate datapoint on 2D and ran K-means clustering on it.

[57]:
```python
from sklearn.cluster import KMeans
from sklearn.manifold import MDS
mds = MDS(2,random_state=0)
df_lg_small = df_lg.sample(n=1500, random_state=1)
df_2d = mds.fit_transform(df_lg_small)
```

[59]:
```python
df_2d = pd.DataFrame(df_2d)
```

[61]:
```python
kmeans = KMeans(n_clusters=3).fit(df_2d)
predict=kmeans.predict(df_2d)
df_2d['cluster_label'] = pd.Series(predict, index=df_2d.index)
```

[62]:
```python
df_2d
```

[62]:

	0	1	cluster_label
0	-0.129837	0.097429	0
1	0.131158	-0.409495	2
2	0.419945	-0.228187	0
3	0.090694	-0.275184	2
4	0.389395	0.098709	0
...	...	...	...
1495	-0.090258	0.202631	0
1496	0.203604	0.470467	0
1497	0.332420	0.171170	0
1498	0.442757	-0.175515	0
1499	-0.131259	-0.261949	2

1500 rows × 3 columns

E. Computed dissimilarity matrix and ran MDS to generate 2D space on features.

[83]:
```python
df_dist = 1 - df_lg.corr()
```

[68]:
```python
mds = MDS(2,random_state=0,dissimilarity="precomputed")
df_2d_dist = mds.fit_transform(df_dist)
```

[72]:
```python
df_2d_dist = pd.DataFrame(df_2d_dist)
df_2d.to_csv('asteroid_mds.csv')
```

[75]:
```python
df_2d_dist.to_csv('asteroid_feat_mds.csv')
```

[76]:
```python
df_dist.columns
```

[76]:
```
Index(['sm_axis', 'eccentricity', 'mag_slope', 'incline', 'longitude',
       'argument_of_perihelion', 'perihelion_distance', 'aphelion_distance',
       'orbital_period', 'data_arc', 'no_obs_used',
       'absolute_magnitude_parameter', 'albedo', 'rotation_period',
       'bv_mag_difference', 'ub_mag_difference', 'moid'],
      dtype='object')
```

[77]:
```python
df_2d_dist['column_name'] = df_dist.columns
```

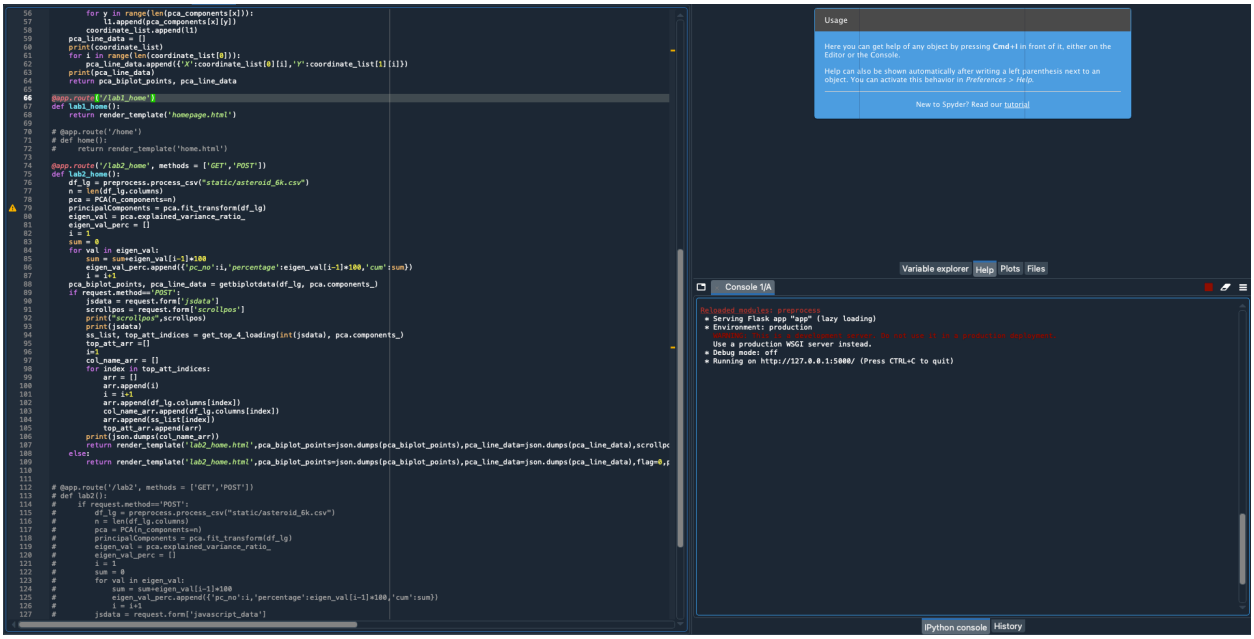
3. Plotting using Flask and D3.js

A. Run Flask server using app.py file

From terminal, command : python app.py

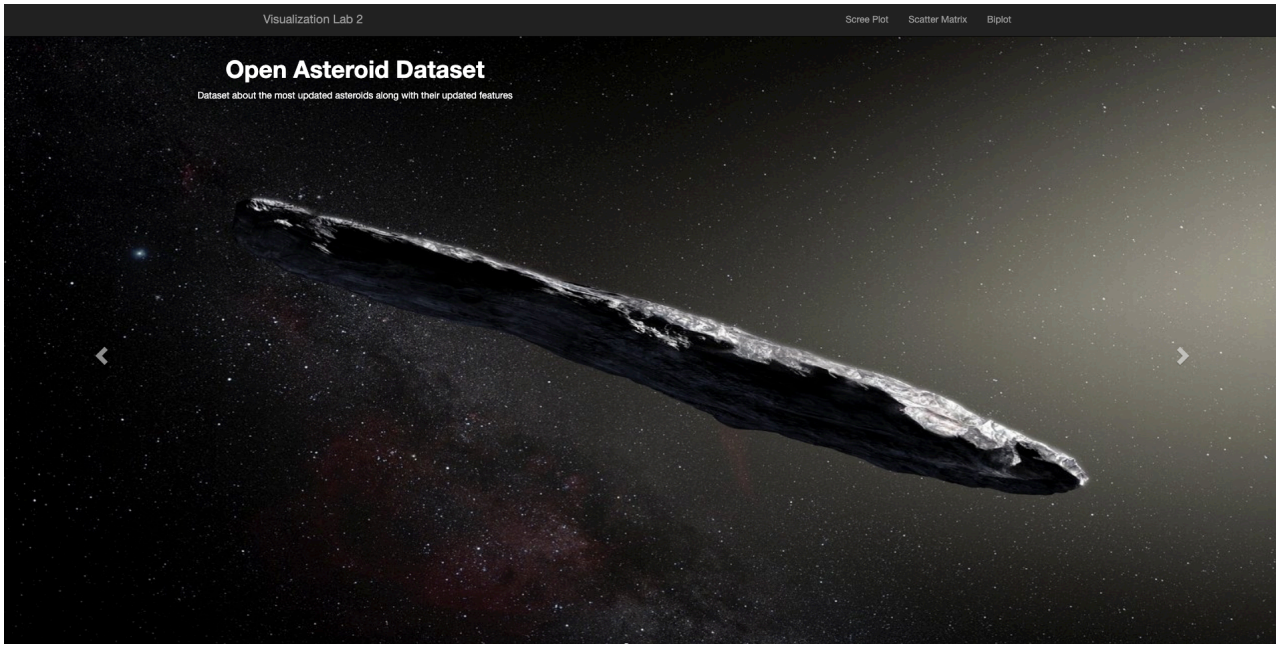
```
(base) nikhil@Nikhils-MacBook-Pro ~ % cd Downloads/Visualization-Lab2
(base) nikhil@Nikhils-MacBook-Pro Visualization-Lab2 % python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

From Spyder IDE:



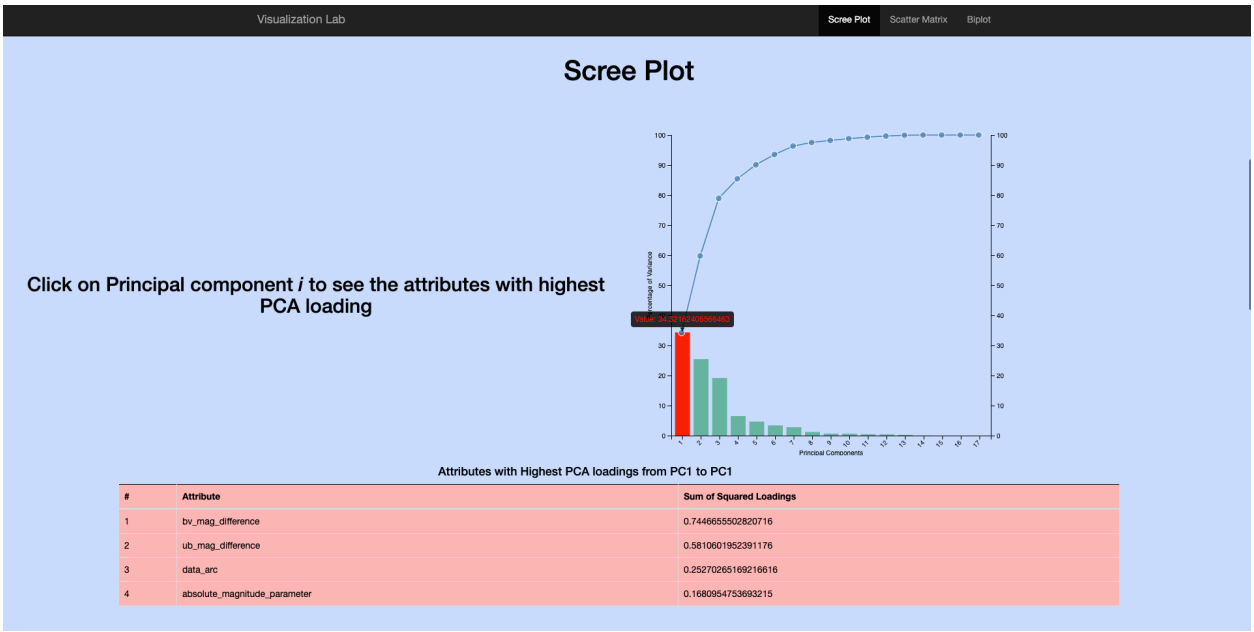
B. **For Tasks 1 and 2:** In browser, go to http://localhost:5000/lab2a_home
It will open the below home page:

C. On, top Navbar, click on Scree Plot/Scatter Matrix/Biplot to move down to respective plots.

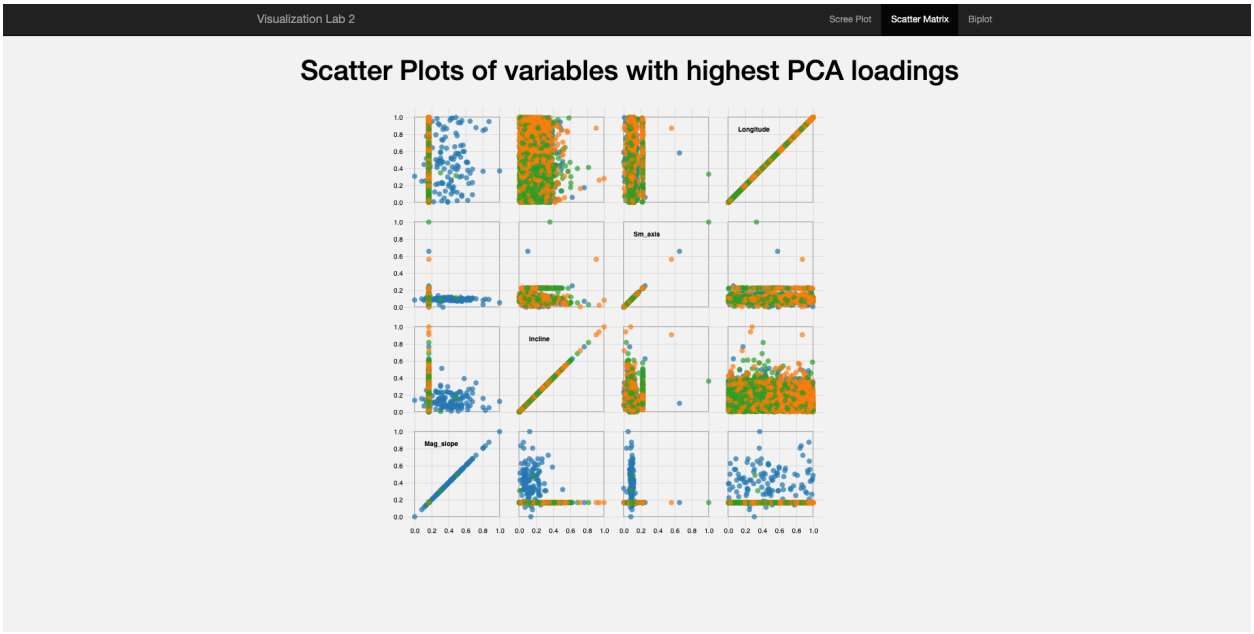


Scree Plot:

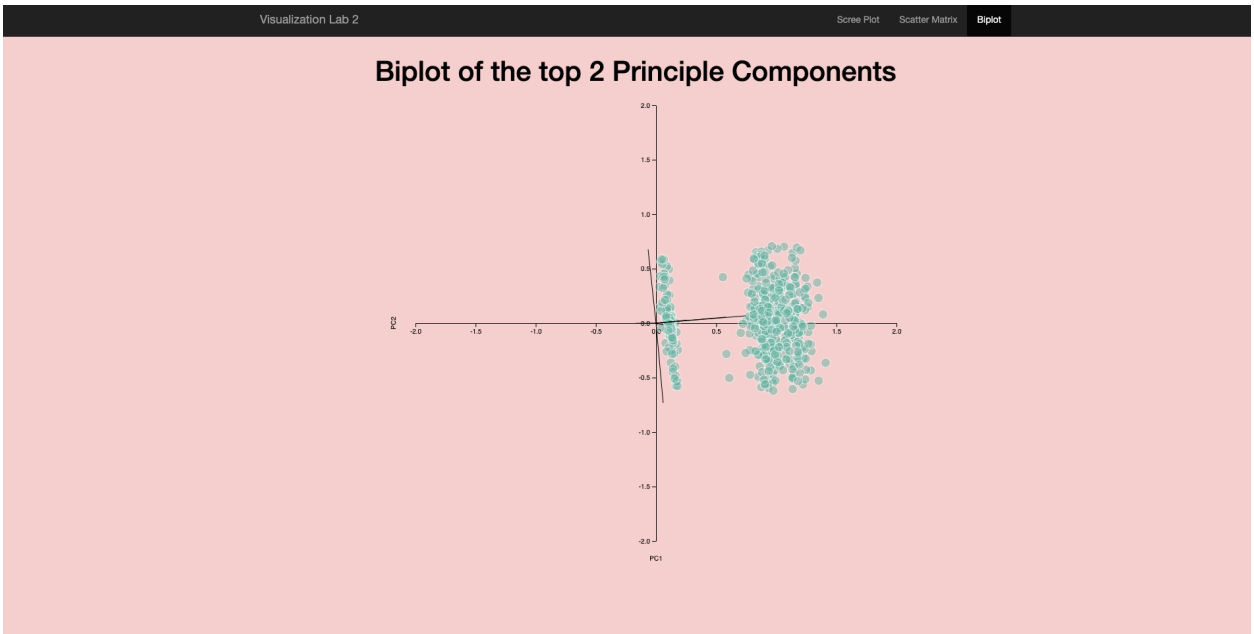
Click on any one of the Principle Component to generate the attributes with highest PCA loading.



Scatter Plot Matrix:

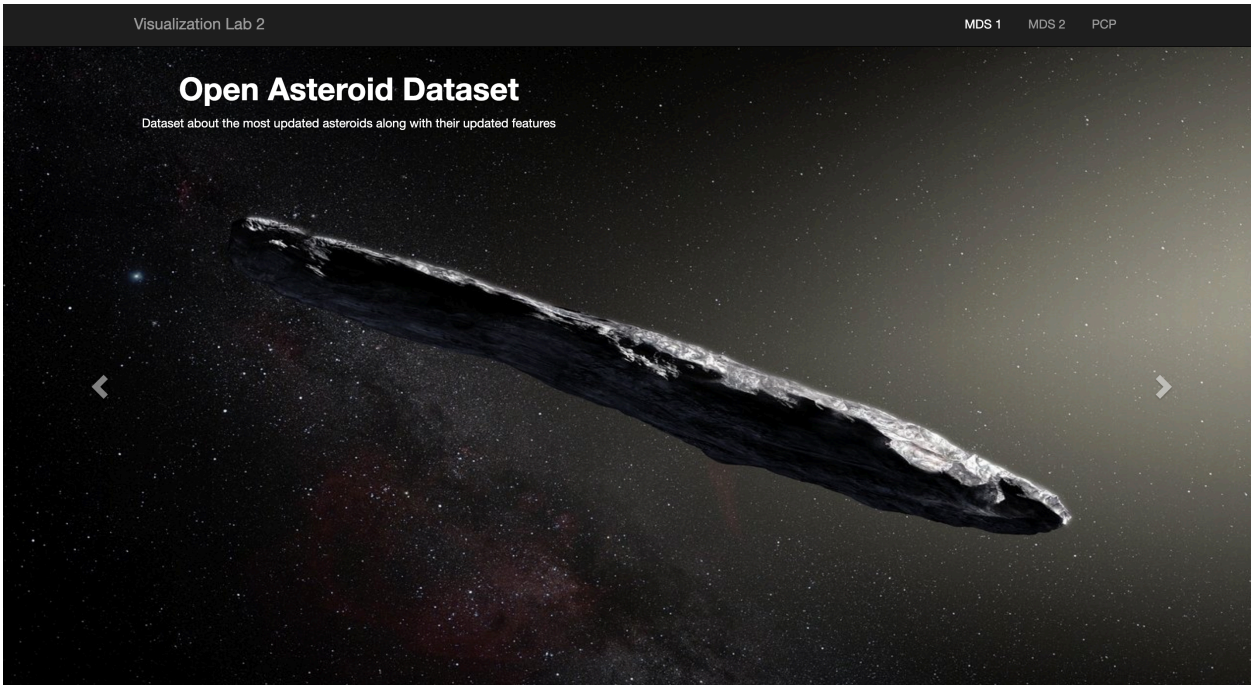


Biplot:

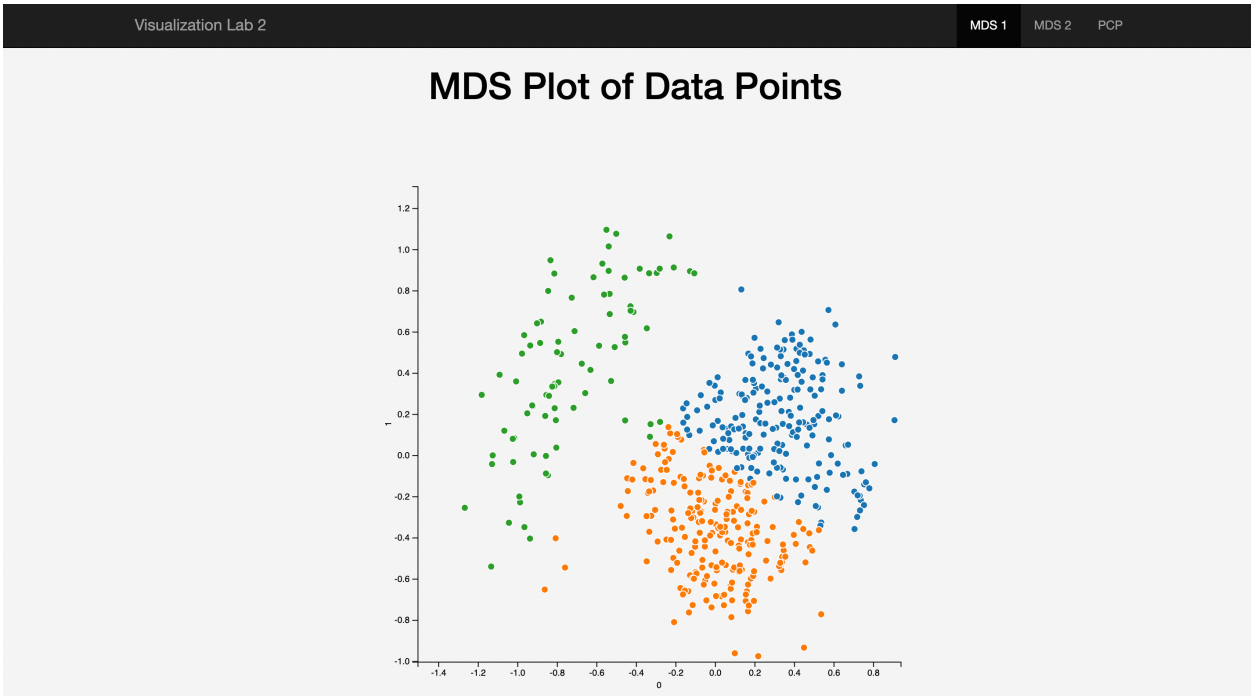


D. **For Tasks 3 and 4** In browser, go to http://localhost:5000/lab2b_home

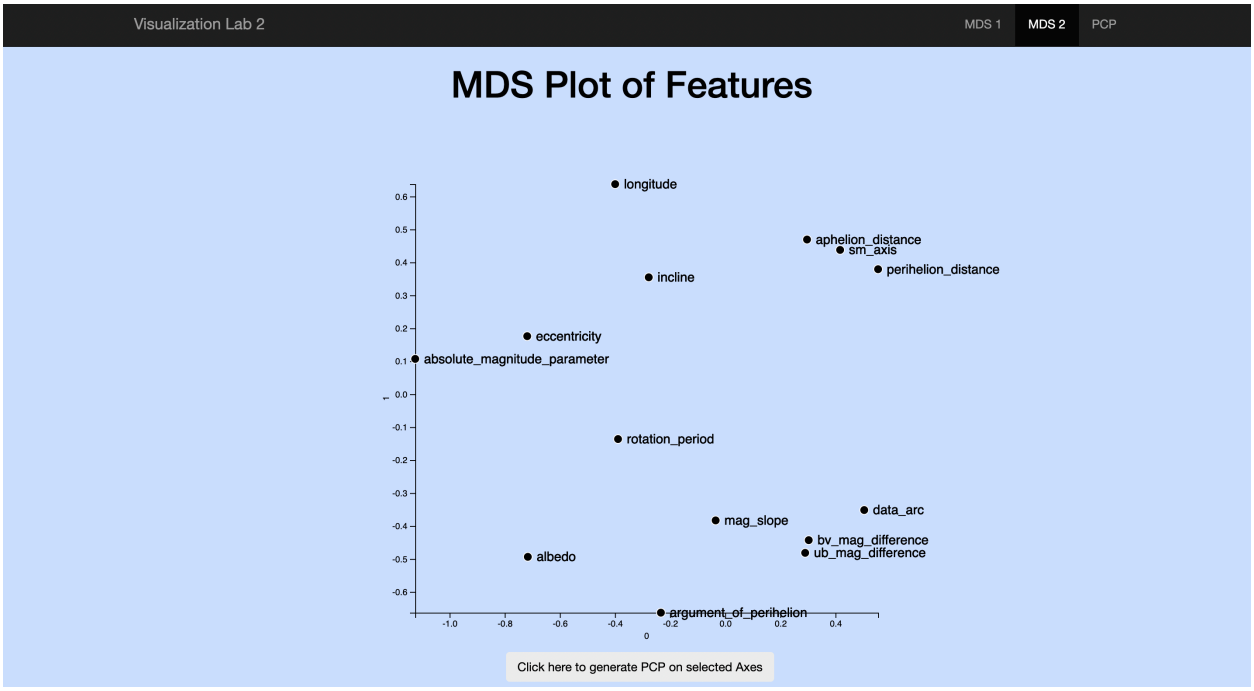
It will open the below home page:



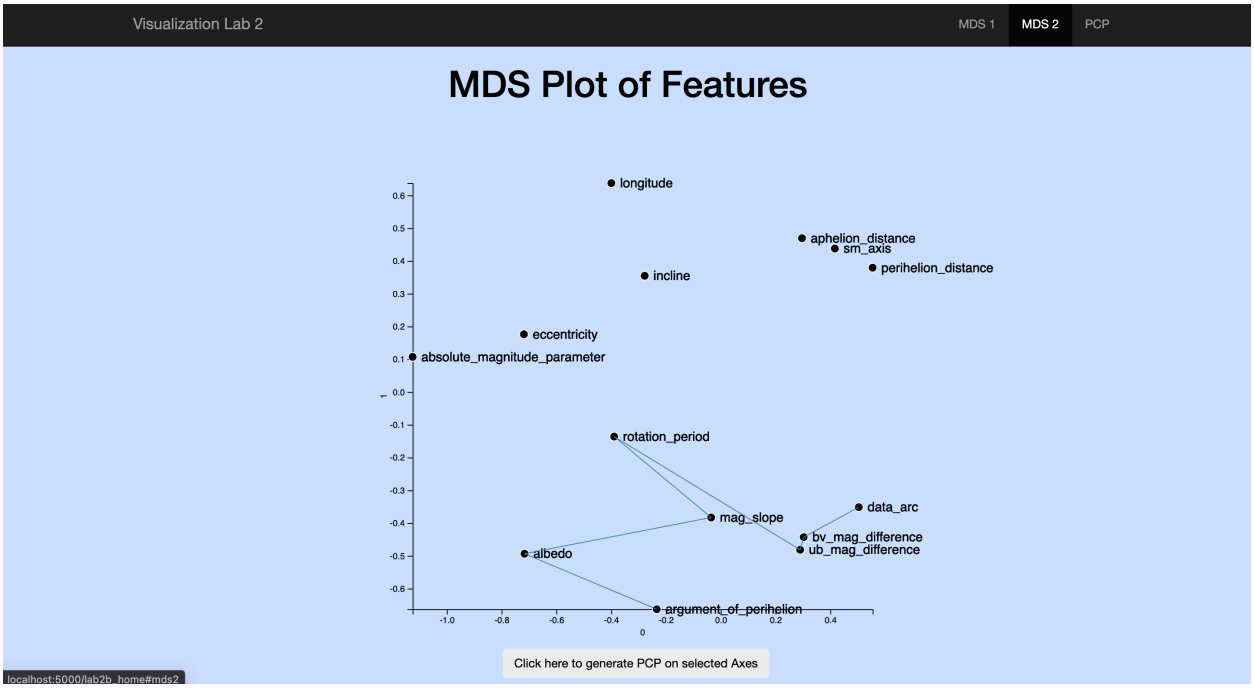
E. MDS Plot of Data Points



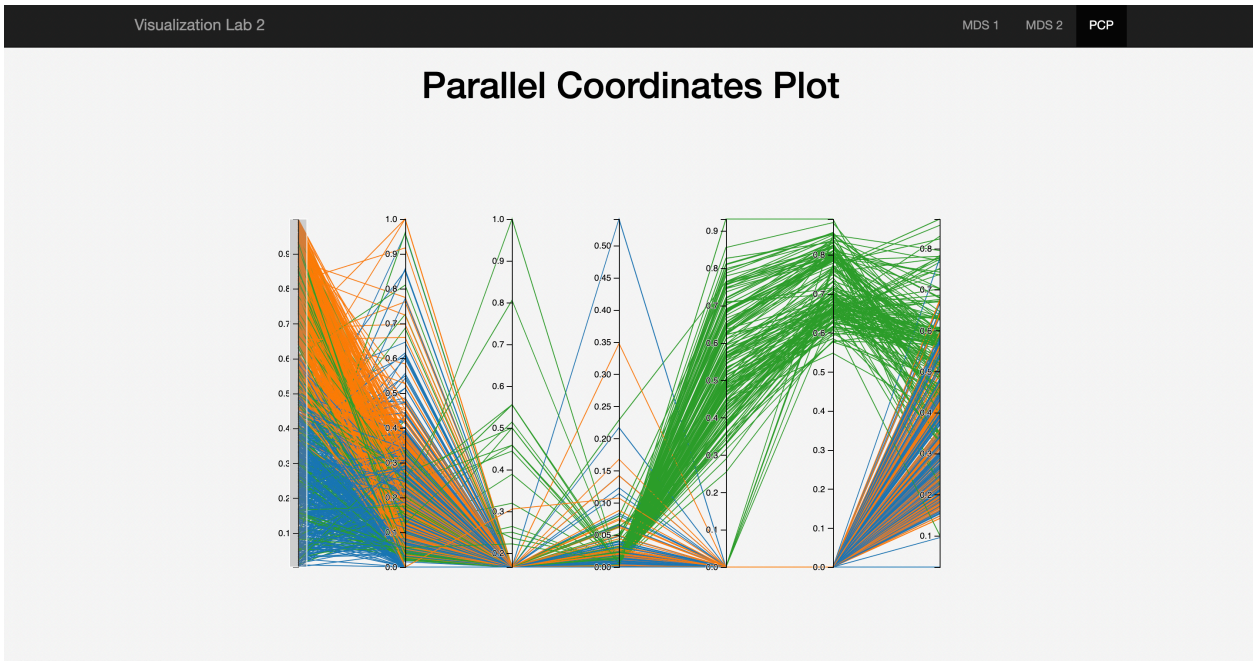
F. MDS Plot of Features. You can click on multiple attributes and generate PCP for the selected attributes.



You can select network of features for PCP plot by clicking on the circles:



G. PCP generated on selected features



H. Default PCP for all features: