CSE 564 - Visualization

Mini Project 2 Report

This report contains the brief implementation details of all the tasks of Mini Project 2 of CSE 564.

Practice the three basic tasks of visual data analytics
(a) use data from mini project #1 (or other), begin with |N|≥500, |D|≥10)
(b) client-server system: python for processing (server), D3 for VIS (client)

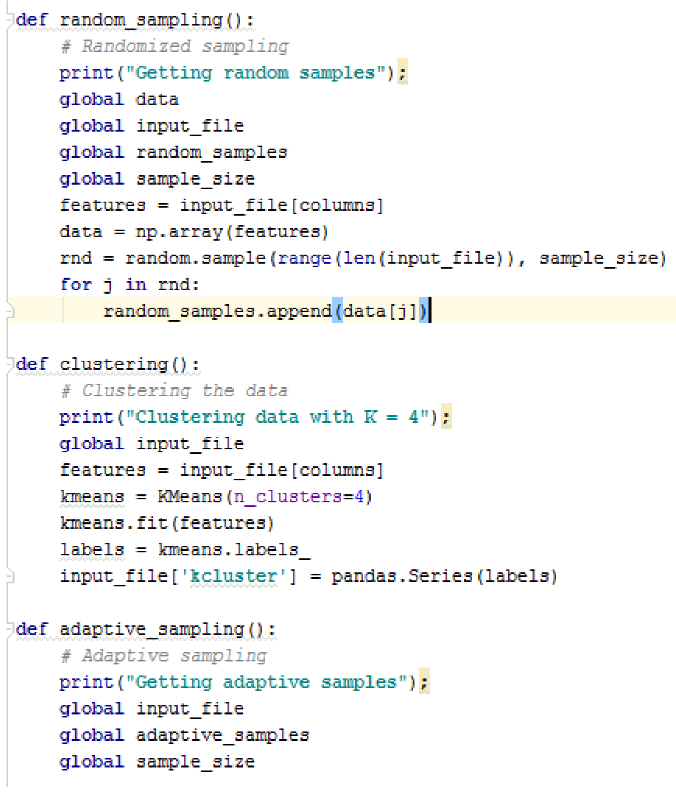
Dataset
I have used *FlightData2008.csv* for this lab. This data has 30 dimensions and has 10000 rows. Half of the columns have non numerical or redundant data. So I have deleted those columns in the start so that the input file is has 15 dimensions. Also the data is standardized to scale the units.

Client-Server System
I have used Python 3.6 for server side program and employed Python Flask Framework. For the client side visualization, I have used D3 library and JavaScript. AJAX calls are made to fetch data from the server.

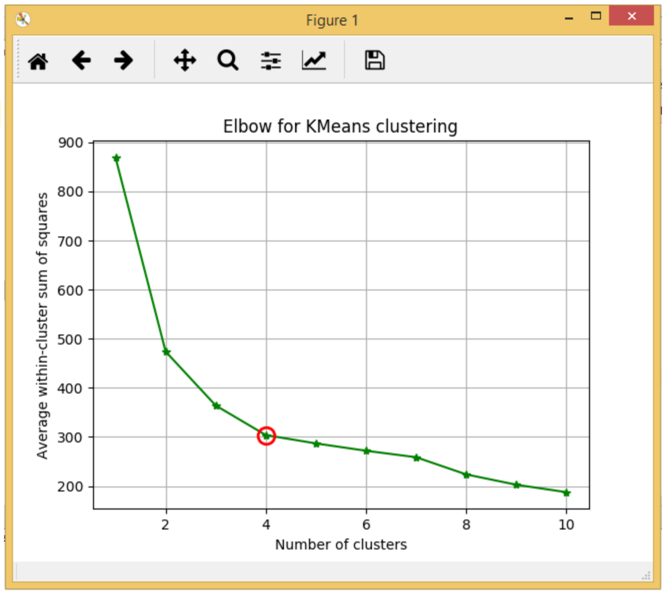
Task 1 - Data clustering and decimation

(a) Implement random sampling and stratified sampling.
(b) The latter includes the need for k-means clustering (optimize k using elbow)

Random sampling and Stratified sampling is implemented with a sample size of 1000.
To implement stratified sampling, we need to cluster the data in k groups. The optimum value of k is calculated using the elbow method in k clustering which comes as 4. The plot and code snippet is shown below -



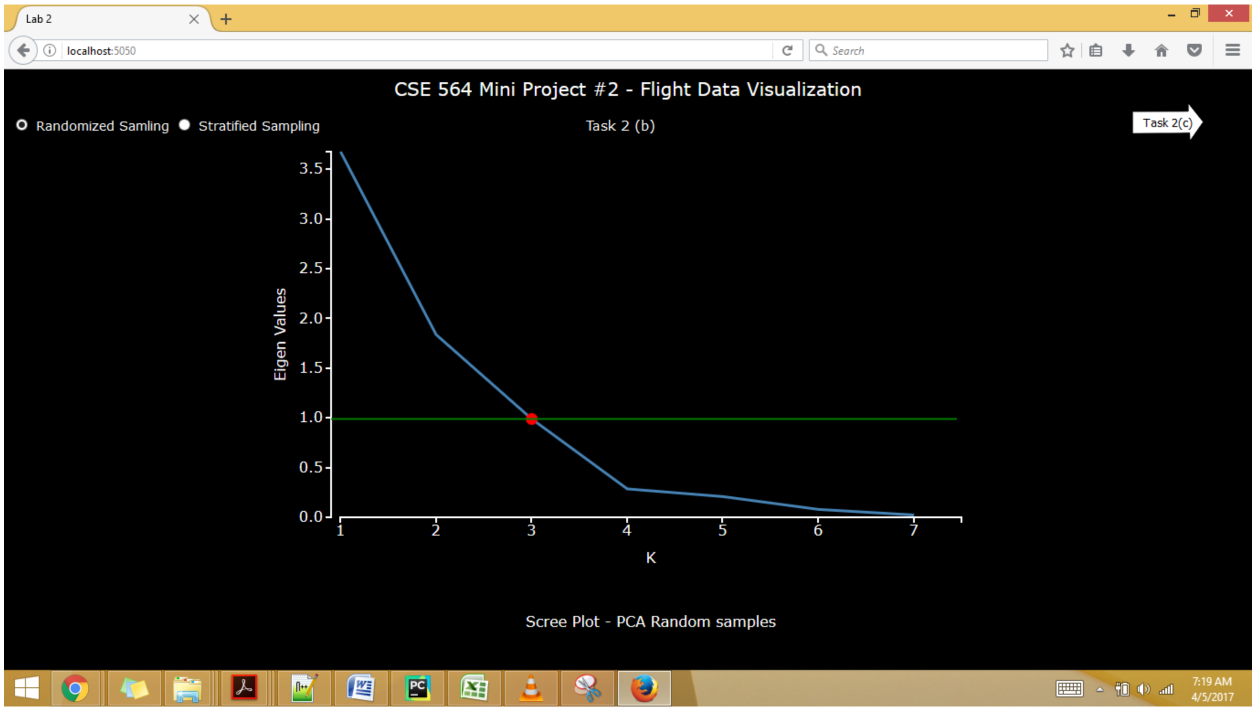
```
def random_sampling():
    # Randomized sampling
    print("Getting random samples");
    global data
    global input_file
    global random_samples
    global sample_size
    features = input_file[columns]
    data = np.array(features)
    rnd = random.sample(range(len(input_file)), sample_size)
    for j in rnd:
        random_samples.append(data[j])

def clustering():
    # Clustering the data
    print("Clustering data with K = 4");
    global input_file
    features = input_file[columns]
    kmeans = KMeans(n_clusters=4)
    kmeans.fit(features)
    labels = kmeans.labels_
    input_file['kcluster'] = pandas.Series(labels)

def adaptive_sampling():
    # Adaptive sampling
    print("Getting adaptive samples");
    global input_file
    global adaptive_samples
    global sample_size
```

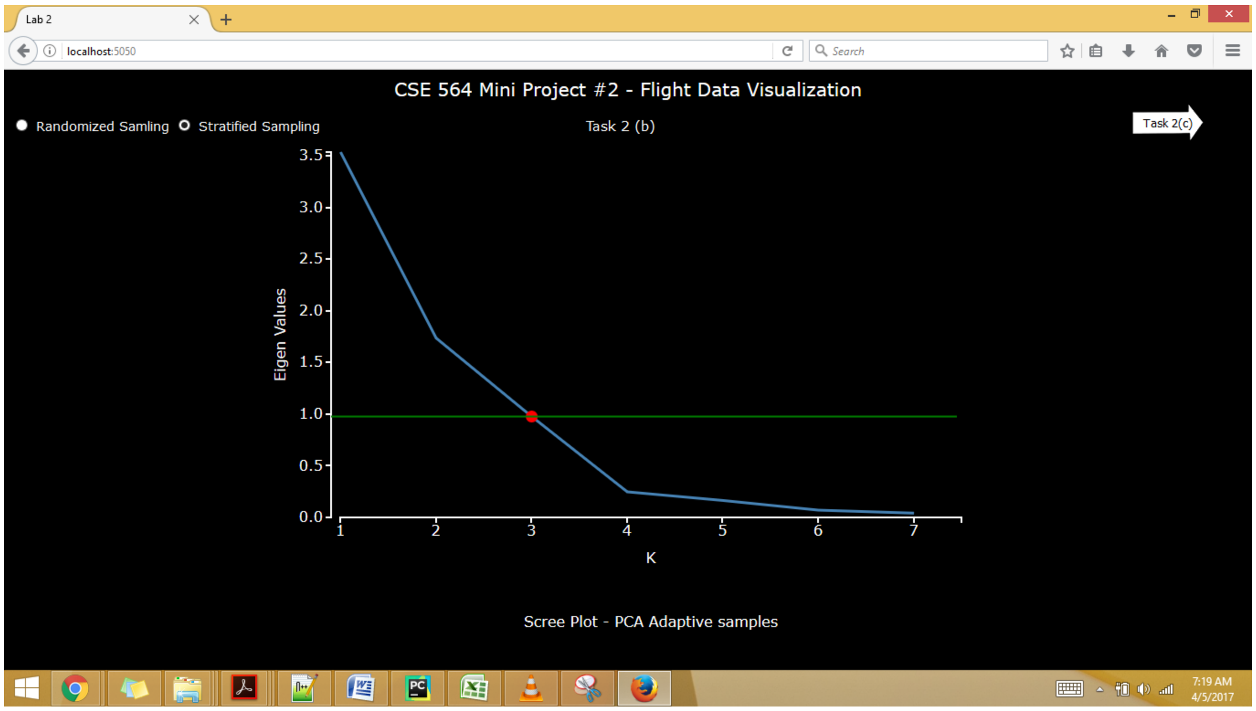


Task 2 - Dimension reduction (use decimated data)

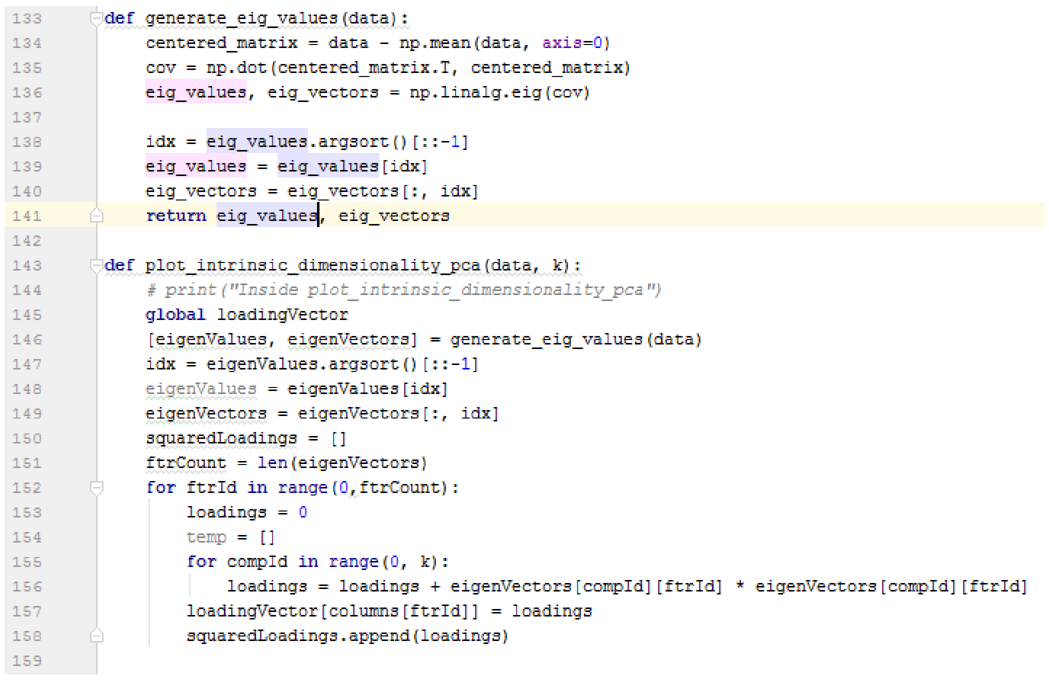
(a) find the intrinsic dimensionality of the data using PCA
(b) produce scree plot visualization and mark the intrinsic dimensionality
(c) obtain the three attributes with highest PCA loadings

Using the Principle component analysis technique, eigen values for each feature is calculated and it is plotted on a scree plot. The components with eigen values more than 1 are considered as the intrinsic dimensions. After drawing the scree plot for both random samples and adaptive samples, I came to conclusion that *there are 3 intrinsic dimensions* in this data. This can be seen in the scree plot as marked with the red dot in figures below -





Once I get the intrinsic dimensionality of the data as 3, I calculated the eigen vector for all the features and their squared loadings. The code snippet to calculate the square loadings is shown below -

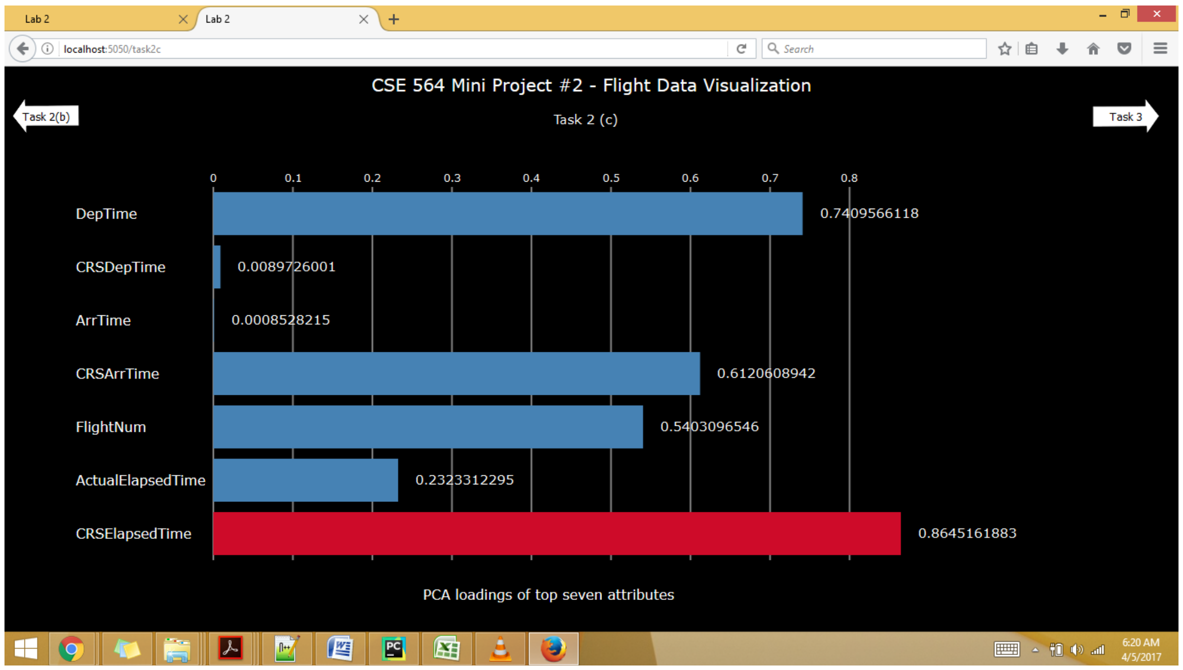


```python
def generate_eig_values(data):
    centered_matrix = data - np.mean(data, axis=0)
    cov = np.dot(centered_matrix.T, centered_matrix)
    eig_values, eig_vectors = np.linalg.eig(cov)

    idx = eig_values.argsort()[::-1]
    eig_values = eig_values[idx]
    eig_vectors = eig_vectors[:, idx]
    return eig_values, eig_vectors

def plot_intrinsic_dimensionality_pca(data, k):
    # print("Inside plot_intrinsic_dimensionality_pca")
    global loadingVector
    [eigenValues, eigenVectors] = generate_eig_values(data)
    idx = eigenValues.argsort()[::-1]
    eigenValues = eigenValues[idx]
    eigenVectors = eigenVectors[:, idx]
    squaredLoadings = []
    ftrCount = len(eigenVectors)
    for ftrId in range(0,ftrCount):
        loadings = 0
        temp = []
        for compId in range(0, k):
            loadings = loadings + eigenVectors[compId][ftrId] * eigenVectors[compId][ftrId]
        loadingVector[columns[ftrId]] = loadings
        squaredLoadings.append(loadings)

```

Once the square loading are calculated, we can determine the top three attributes with highest loadings. These attributes are -

1. CRSElapsedTime
2. DepTime
3. CRSArrTime

This is visualized with the help of a horizontal bar chart as shown in the figure below. The value with highest squared loading is highlighted -



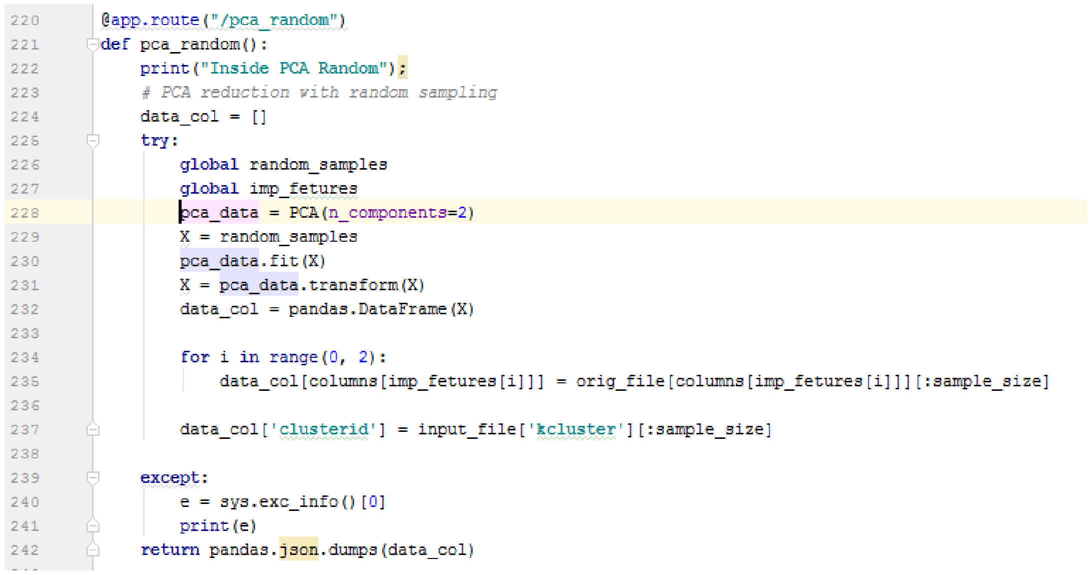
Task 3 - Visualization (use dimension reduced data)

(a) visualize data projected into the top two PCA vectors via 2D scatter plot
(b) visualize data via MDS (Euclidean & correlation distance) in 2D scatter plots
(c) visualize scatter plot matrix of the three highest PCA loaded attributes

Once I know the squared loadings of all the features, I reduced the dimension of the data taking the top seven attributes which contribute almost 98% in the principal components. These components are -

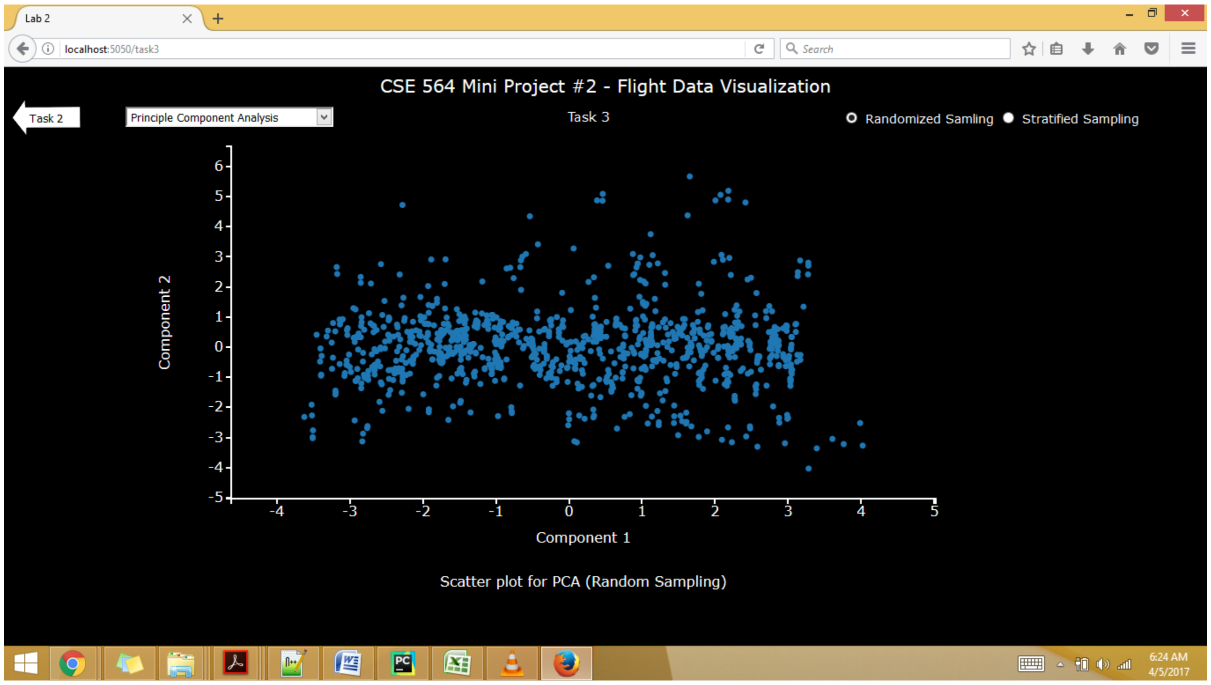
'DepTime', 'CRSDepTime', 'ArrTime', 'CRSArrTime', 'FlightNum', 'ActualElapsedTime', 'CRSElapsedTime'

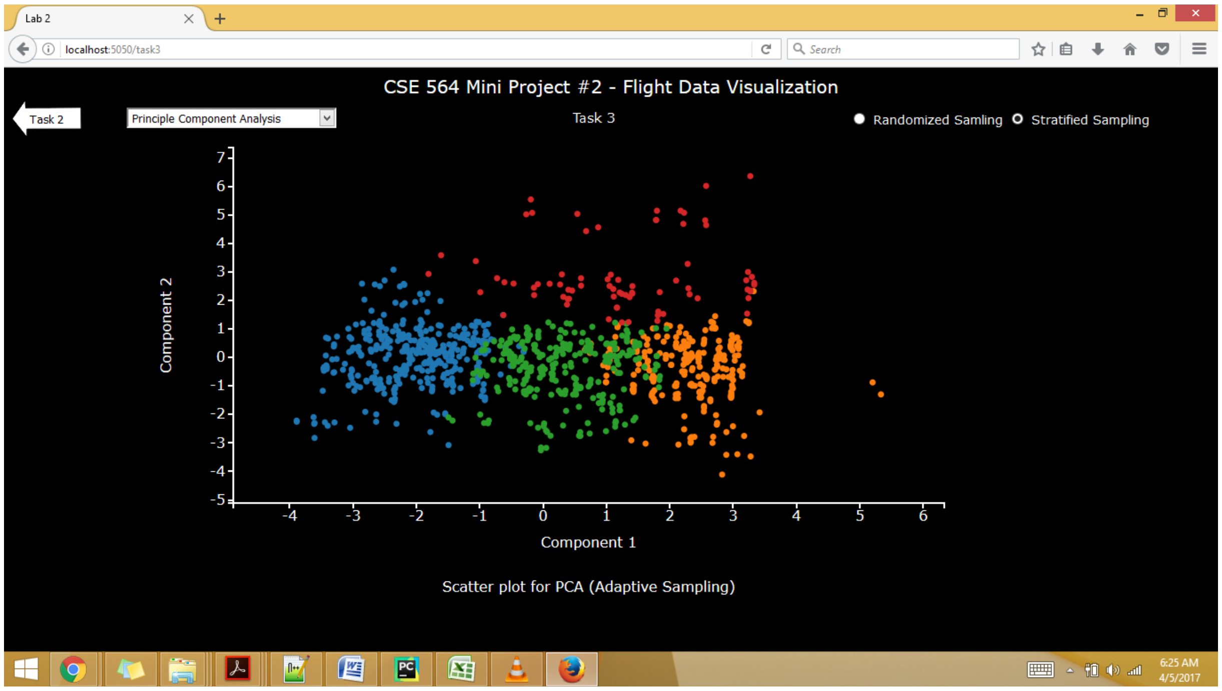
Now I pass these columns of the random and adaptive samples from task 1 to the function written to calculate the principle components. PCA library of sklearn.decomposition in Python is used to calculate the output for 2 PCA. The code snippet is shown below -



```
@app.route("/pca_random")
def pca_random():
    print("Inside PCA Random");
    # PCA reduction with random sampling
    data_col = []
    try:
        global random_samples
        global imp_fetures
        pca_data = PCA(n_components=2)
        X = random_samples
        pca_data.fit(X)
        X = pca_data.transform(X)
        data_col = pandas.DataFrame(X)

        for i in range(0, 2):
            data_col[columns[imp_fetures[i]]] = orig_file[columns[imp_fetures[i]]][:sample_size]

        data_col['clusterid'] = input_file['kcluster'][:sample_size]

    except:
        e = sys.exc_info()[0]
        print(e)
    return pandas.json.dumps(data_col)
```

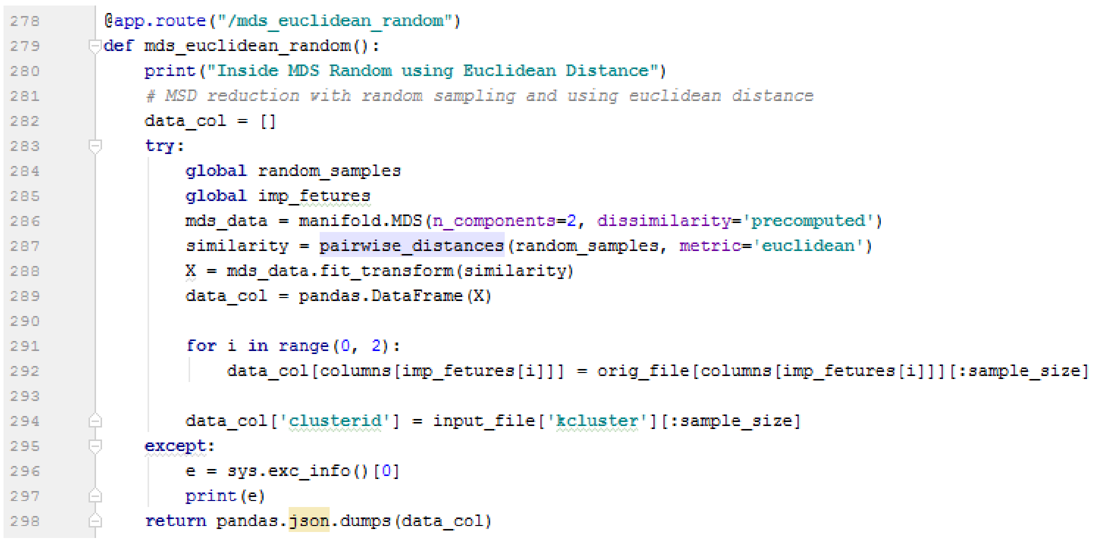
The scatter plots of PCA for random and adaptive samples are shown below -



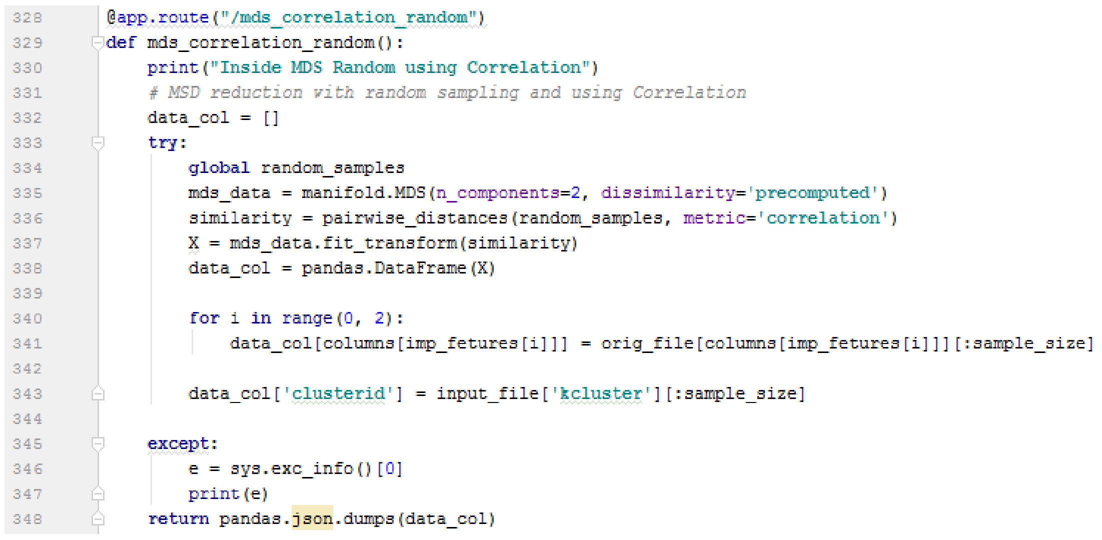


These plots show that the PCA technique can be used to reduce the dimensions of the data without losing much information. The data is transformed such that component with the maximum variation is plotted on the X-axis and component with the second best variation is plotted on the Y-axis. The clusters in the scatter plot with adaptive sampling clearly shows that the data with similar features are clustered together.

Similarly, the MDS technique is employed to show its dimension reduction capabilities. Both Euclidean distance and correlation distance methods are used on both random and adaptive samples from task 1. MDS and pairwise_distance form sklearn.metrices are used to calculate the pair-wise distance and the MDS. The code snippets are shown below -

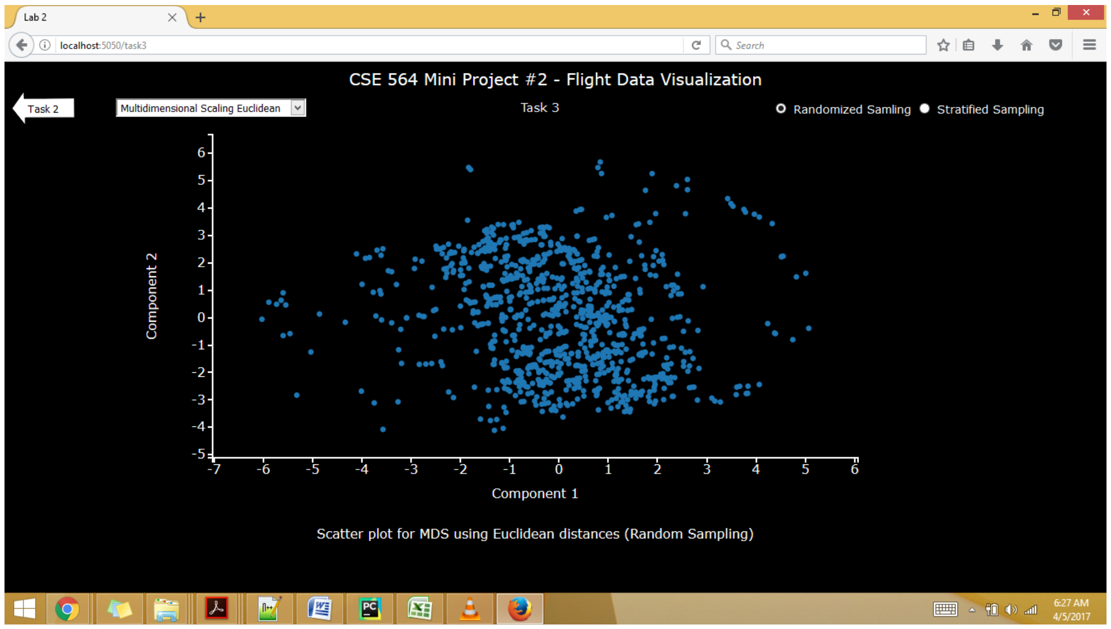


```python
@app.route("/mds_euclidean_random")
def mds_euclidean_random():
    print("Inside MDS Random using Euclidean Distance")
    # MSD reduction with random sampling and using euclidean distance
    data_col = []
    try:
        global random_samples
        global imp_fetures
        mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
        similarity = pairwise_distances(random_samples, metric='euclidean')
        X = mds_data.fit_transform(similarity)
        data_col = pandas.DataFrame(X)

        for i in range(0, 2):
            data_col[columns[imp_fetures[i]]] = orig_file[columns[imp_fetures[i]]][:sample_size]

        data_col['clusterid'] = input_file['kcluster'][:sample_size]
    except:
        e = sys.exc_info()[0]
        print(e)
    return pandas.json.dumps(data_col)
```

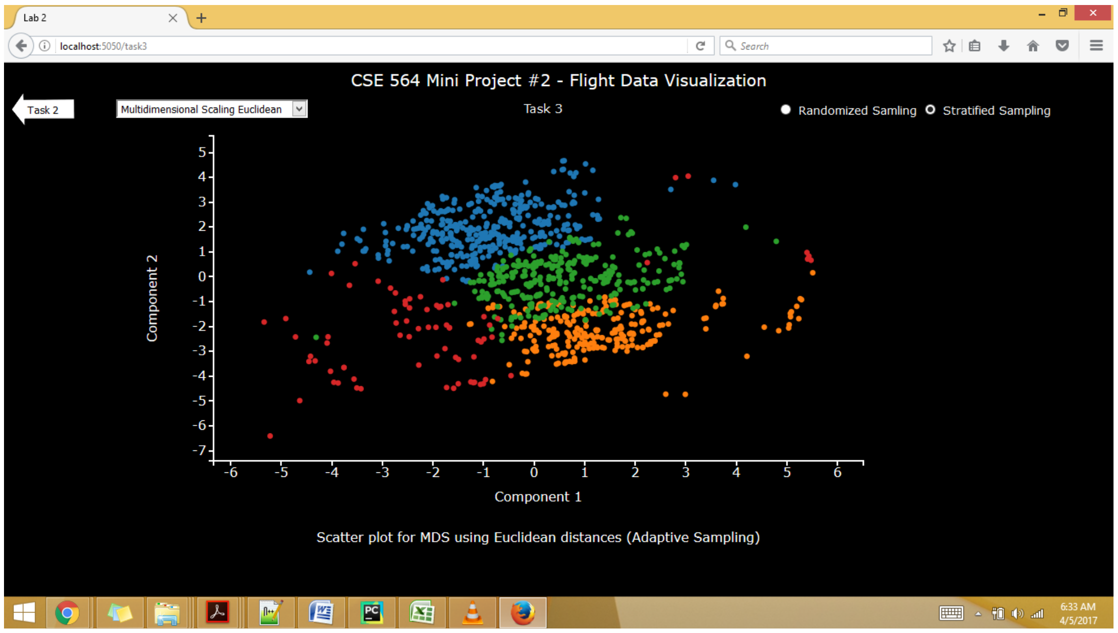


```python
@app.route("/mds_correlation_random")
def mds_correlation_random():
    print("Inside MDS Random using Correlation")
    # MSD reduction with random sampling and using Correlation
    data_col = []
    try:
        global random_samples
        mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')
        similarity = pairwise_distances(random_samples, metric='correlation')
        X = mds_data.fit_transform(similarity)
        data_col = pandas.DataFrame(X)

        for i in range(0, 2):
            data_col[columns[imp_fetures[i]]] = orig_file[columns[imp_fetures[i]]][:sample_size]

        data_col['clusterid'] = input_file['kcluster'][:sample_size]

    except:
        e = sys.exc_info()[0]
        print(e)
    return pandas.json.dumps(data_col)
```

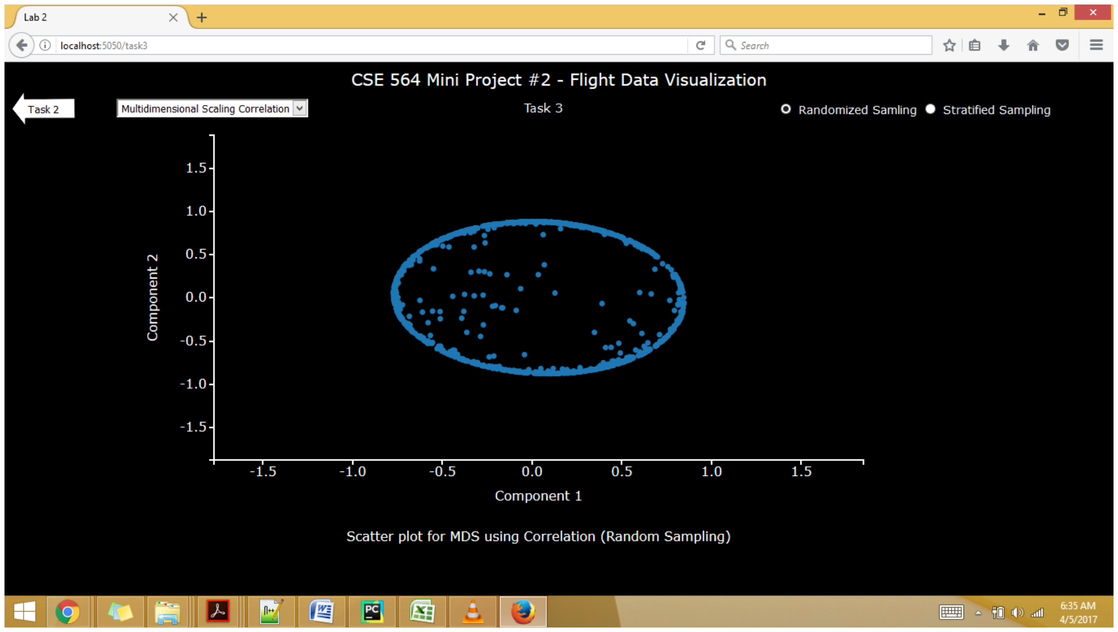
We get the following four scatter plots for the MDS techniques -

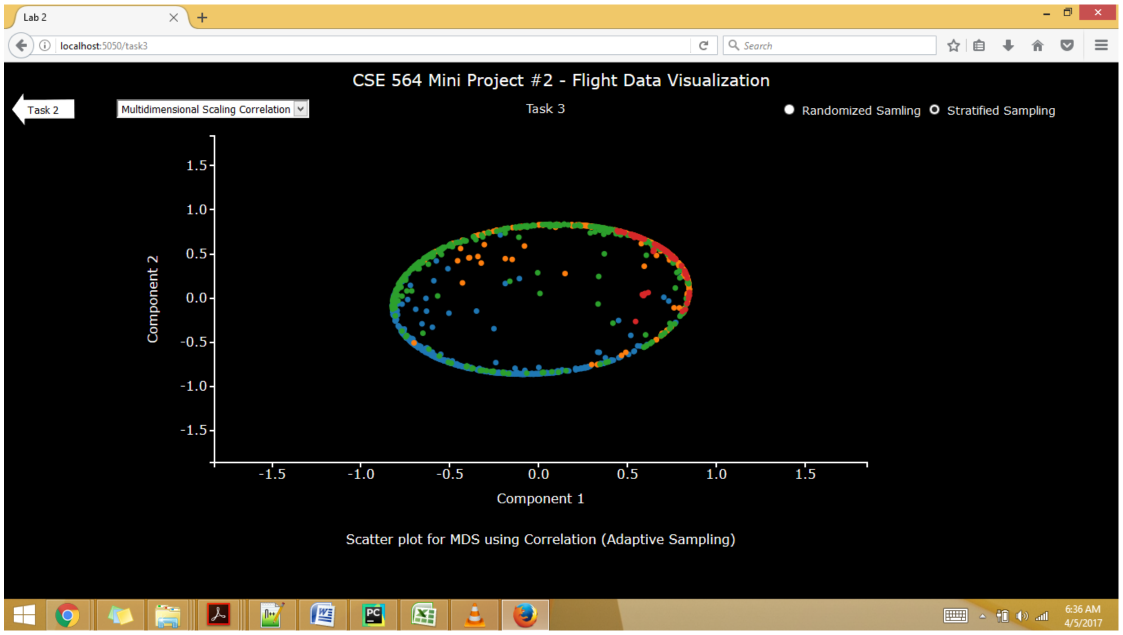


Scatter plot for MDS using Euclidean distances (Random Sampling)



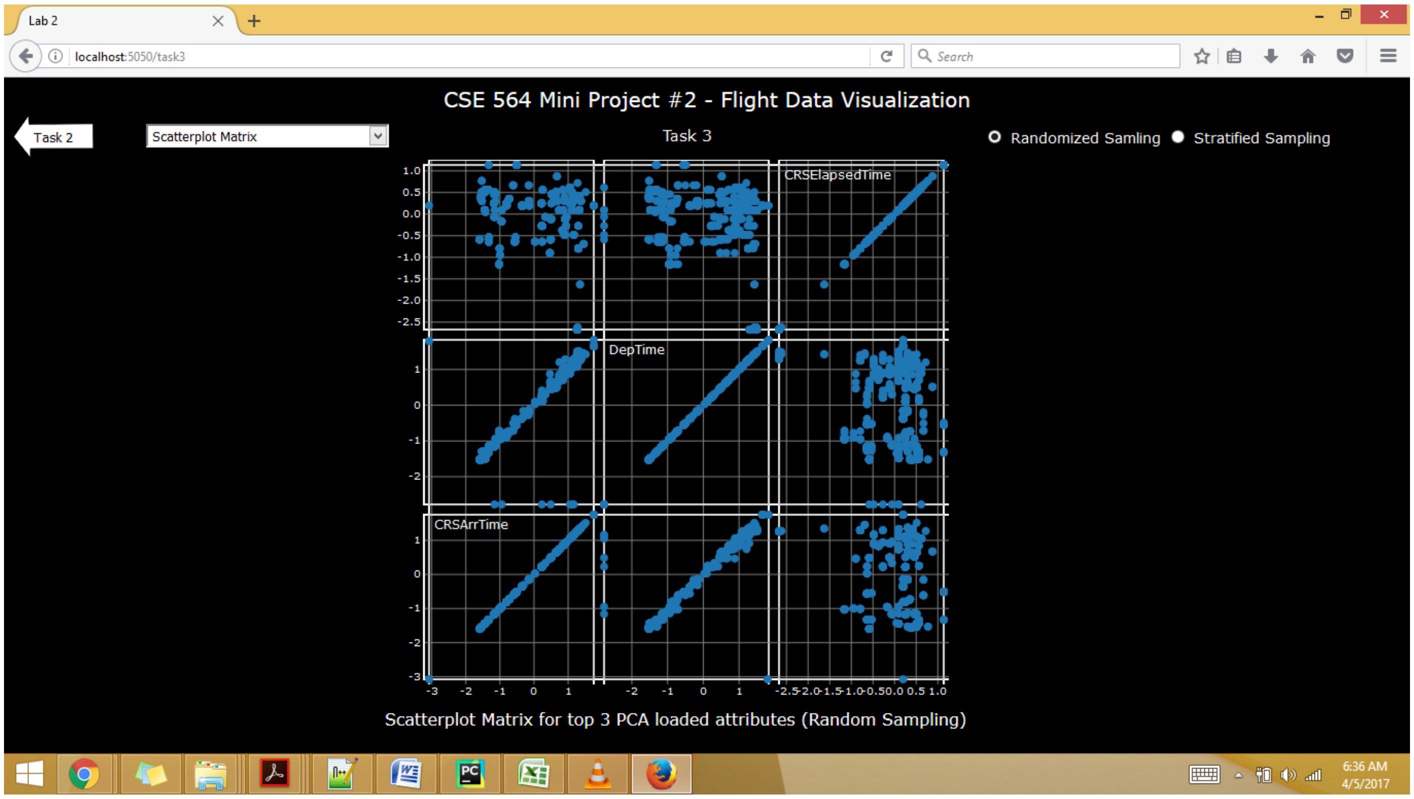
Scatter plot for MDS using Euclidean distances (Adaptive Sampling)





MDS finds a set of vectors in p-dimensional space such that the matrix of Euclidean distances among them corresponds as closely as possible to some function of the input matrix according to a criterion function called stress. The plot depicts the same behavior as all the points are closely packed together. The Euclidean distance groups the points according to the Euclidean distances while the correlation method tries to put variables with high positive correlations near each other, and variables with strong negative correlations far apart. The plots with the adaptive samples depict that the points in the same clusters are placed closed to each other as they will have very less Euclidean distance (in the Euclidean plots) and high Correlation (in the correlation plot).

Finally, I have visualized the scatterplot matrix of the three highest loaded attributes calculated in the task 2(c). The scatterplot shows the pair-wise relation with one another. There are 9 sub plots showing the variance and covariance between the parameters placed on x-axis and y-axis. The adaptive plot again shows that points in the same clusters are closely placed as they are similar.

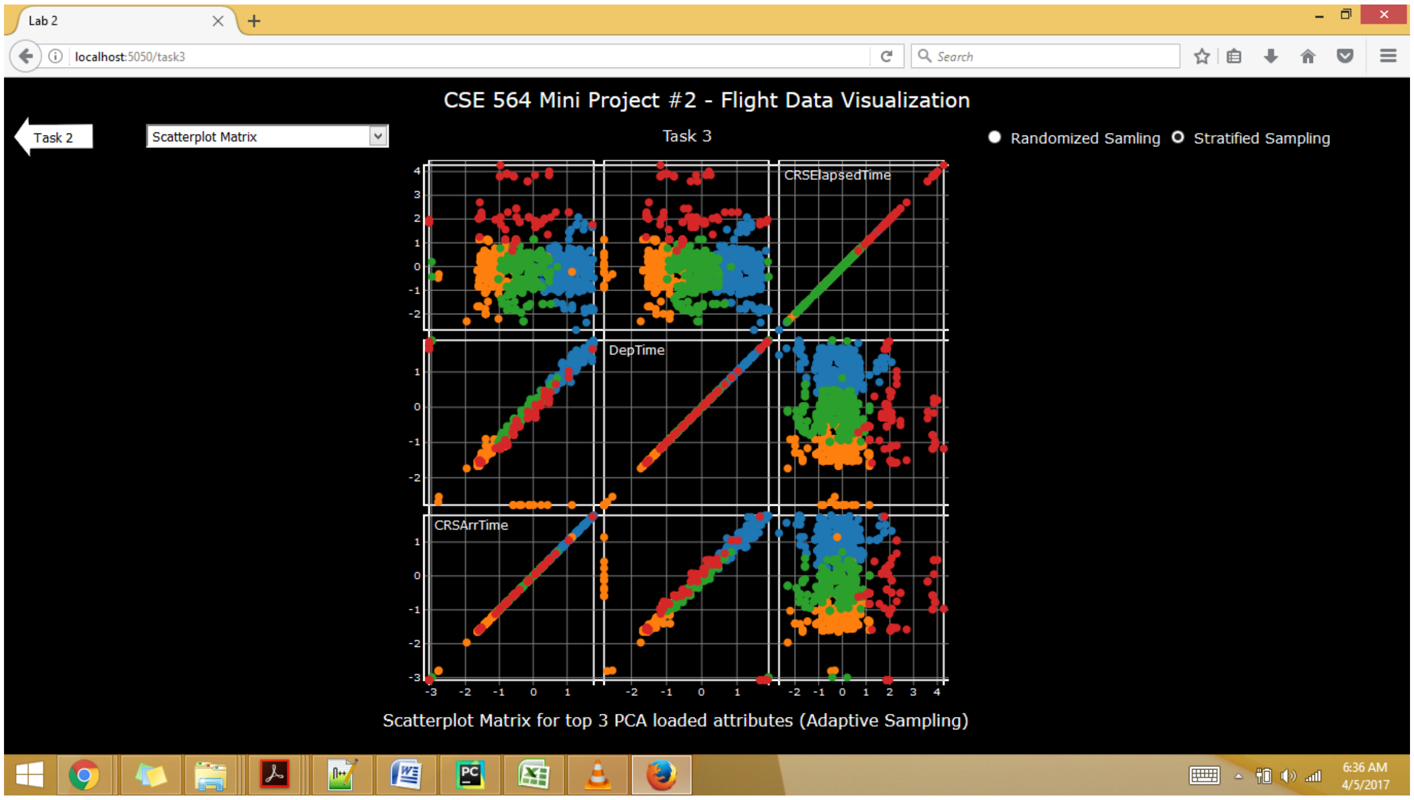


CSE 564 Mini Project #2 - Flight Data Visualization

Task 3

Randomized Samling [✓] Stratified Sampling

Scatterplot Matrix for top 3 PCA loaded attributes (Random Sampling)



CSE 564 Mini Project #2 - Flight Data Visualization

Task 3

Randomized Samling [✓ Stratified Sampling]

Scatterplot Matrix for top 3 PCA loaded attributes (Adaptive Sampling)