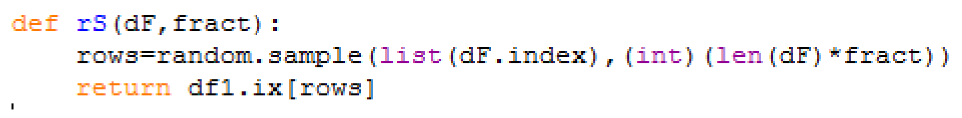
CSE 564 – Mini Project 2 **Manisha Mishra**

Data Set Chosen – Orange Juice Data Set
The data contains 1070 purchases where the customer either purchased Citrus Hill or Minute Maid Orange Juice. It has 1070 observations on 16 variables. The data set can be found at https://vincentarelbundock.github.io/Rdatasets/doc/ISLR/OJ.html .

Python was used to analyze the data and convert to .csv to be plotted in JavaScript.

Task 1 - Data clustering and Decimation

Random Sampling: Implemented using the random.sample function. The sampled list has 20% of the original data set and is stored as a.



```
def rS(dF,fract):
    rows=random.sample(list(dF.index),(int)(len(dF)*fract))
    return df1.ix[rows]
```

Fig 1. Random Sampling implementation

K-Means Clustering: In order to proceed with Adaptive Sampling, we first find the optimal number of clusters for the data set by plotting the elbow plot (number of clusters vs variance of each cluster). It can be seen in Fig 2 that the optimal number of clusters is 4 in this case. The sklearn module in Python was used for clustering.

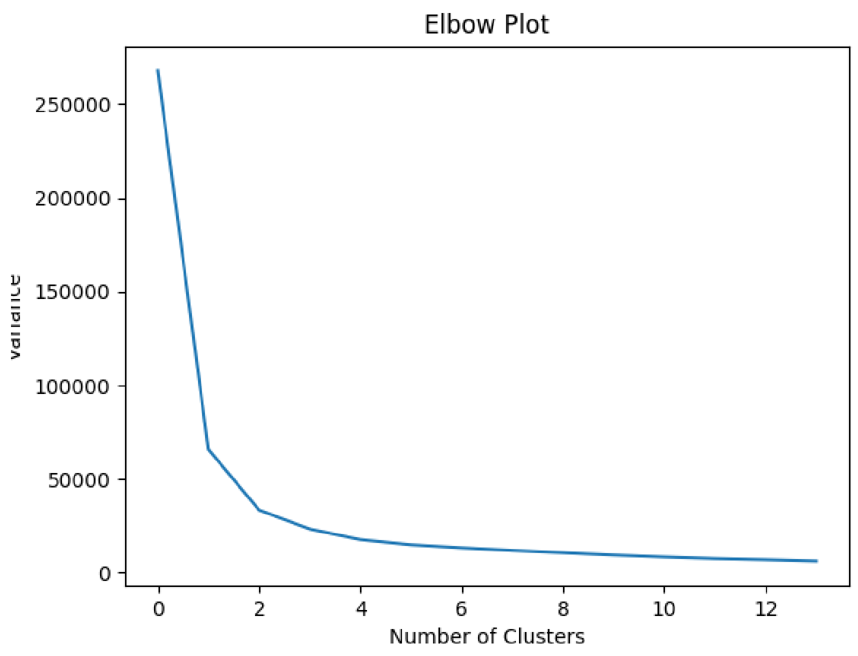
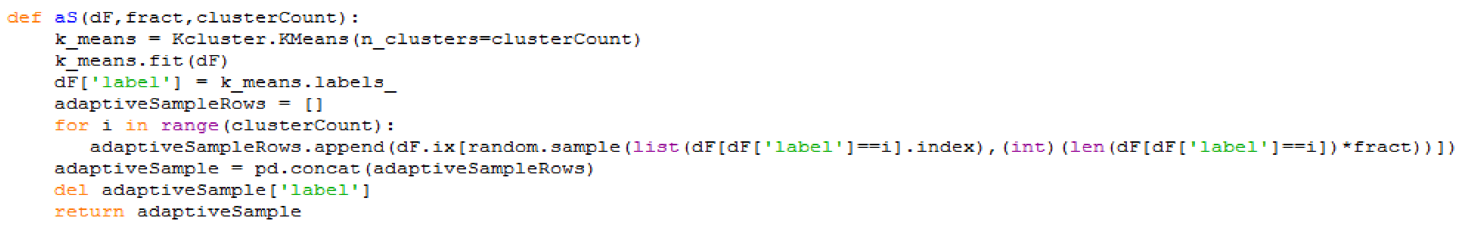


Fig 2. Elbow Plot

Stratified Sampling: Done with 4 clusters and the final sample had 20% of the original data



```python
def aS(dF,fract,clusterCount):
    k_means = Kcluster.KMeans(n_clusters=clusterCount)
    k_means.fit(dF)
    dF['label'] = k_means.labels_
    adaptiveSampleRows = []
    for i in range(clusterCount):
       adaptiveSampleRows.append(dF.ix[random.sample(list(dF[dF['label']==i].index),(int)(len(dF[dF['label']==i])*fract))])
    adaptiveSample = pd.concat(adaptiveSampleRows)
    del adaptiveSample['label']
    return adaptiveSample
```

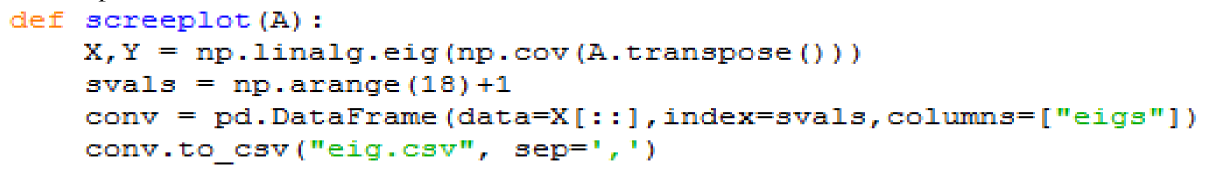
set.

Fig 3. Adaptive Sampling Implementation

Task 2 – Dimension Reduction

Intrinsic dimensionality of the data using PCA: The intrinsic dimensionality was found through the elbow of the scree plot. The data was stored in a CSV file and visualized through d3. Fig 4 shows the code for the same.

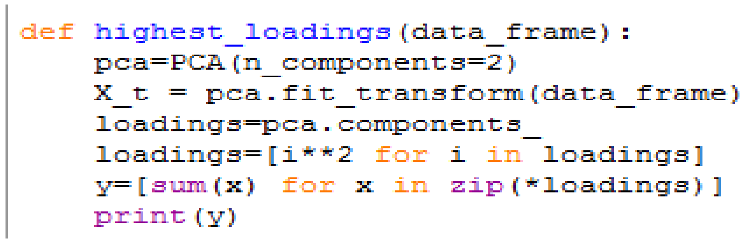
Produce Scree Plot Visualization and Mark the Intrinsic Dimensionality: Fig 4 shows the Python implementation.



```python
def screeplot(A):
    X,Y = np.linalg.eig(np.cov(A.transpose()))
    svals = np.arange(18)+1
    conv = pd.DataFrame(data=X[::],index=svals,columns=["eigs"])
    conv.to_csv("eig.csv", sep=',')
```

Fig 4. Scree plot CSV file Implementation

Three Attributes with Highest PCA Loadings: In the code shown below, the square of the variance for each attribute is computed and plotted in d3 and stored in a CSV file for visualization in d3.

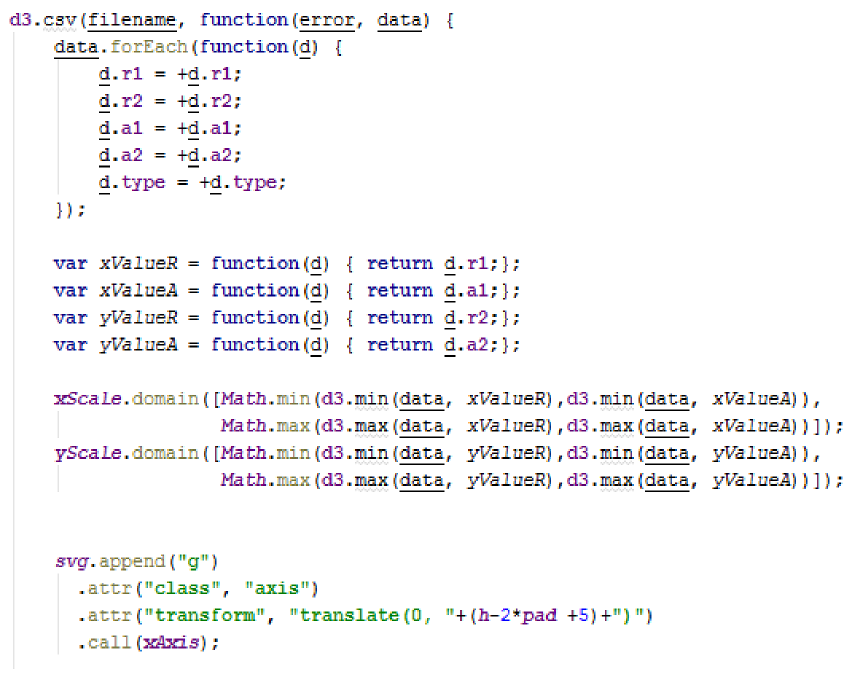


```python
def highest_loadings(data_frame):
    pca=PCA(n_components=2)
    X_t = pca.fit_transform(data_frame)
    loadings=pca.components_
    loadings=[i**2 for i in loadings]
    y=[sum(x) for x in zip(*loadings)]
    print(y)
```

Fig 5. Attributes with highest PCA loadings

Task 3 - Visualization

Visualize Data Projected into the top two PCA vectors via 2D Scatterplot: The pca2.csv file has the PCA values of the sampled data (random and adaptive sampling). The d3 implementation is shown in Fig 6.

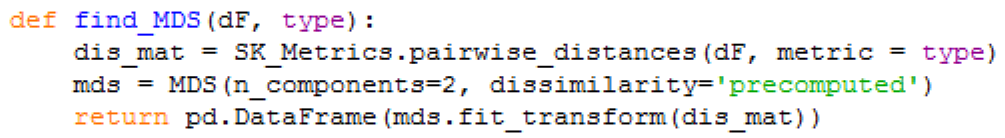


```
d3.csv(filename, function(error, data) {
    data.forEach(function(d) {
        d.r1 = +d.r1;
        d.r2 = +d.r2;
        d.a1 = +d.a1;
        d.a2 = +d.a2;
        d.type = +d.type;
    });

    var xValueR = function(d) { return d.r1;};
    var xValueA = function(d) { return d.a1;};
    var yValueR = function(d) { return d.r2;};
    var yValueA = function(d) { return d.a2;};

    xScale.domain([Math.min(d3.min(data, xValueR),d3.min(data, xValueA)),
                   Math.max(d3.max(data, xValueR),d3.max(data, xValueA))]);
    yScale.domain([Math.min(d3.min(data, yValueR),d3.min(data, yValueA)),
                   Math.max(d3.max(data, yValueR),d3.max(data, yValueA))]);


    svg.append("g")
      .attr("class", "axis")
      .attr("transform", "translate(0, "+(h-2*pad +5)+")")
      .call(xAxis);
```

Fig 6. PCA Scatterplot Implementation

Visualize data via MDS (Euclidian & Correlation Distance) in 2D Scatterplots: Computed in Python and stored in .csv format. Visualization is done in d3.

Fig 7.



```
def find_MDS(dF, type):
    dis_mat = SK_Metrics.pairwise_distances(dF, metric = type)
    mds = MDS(n_components=2, dissimilarity='precomputed')
    return pd.DataFrame(mds.fit_transform(dis_mat))
```

MDS Computation in Python

Visualize Scatterplot Matrix of the Three Highest PCA Loaded Attributes: The top three PCA loaded attributes are visualized using a scattermatrix to represent how they are related to each other.

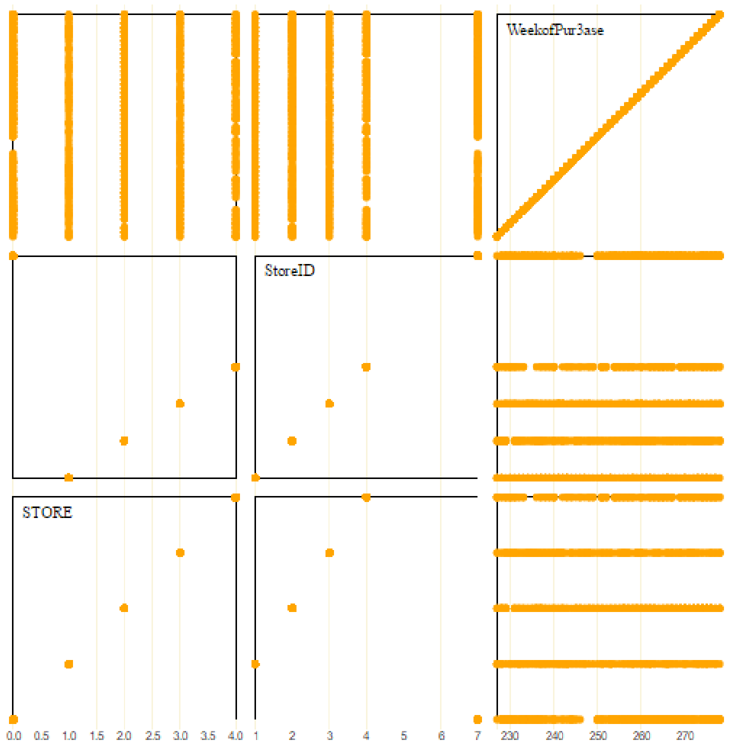


Fig 8. Scatterplot Matrix