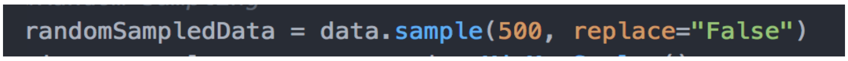
CSE 564 LAB 2 REPORT

Student Id: 112077242
Name: Kajal Dalvi

YouTube Video Link: https://youtu.be/FI3JhPxHxRU

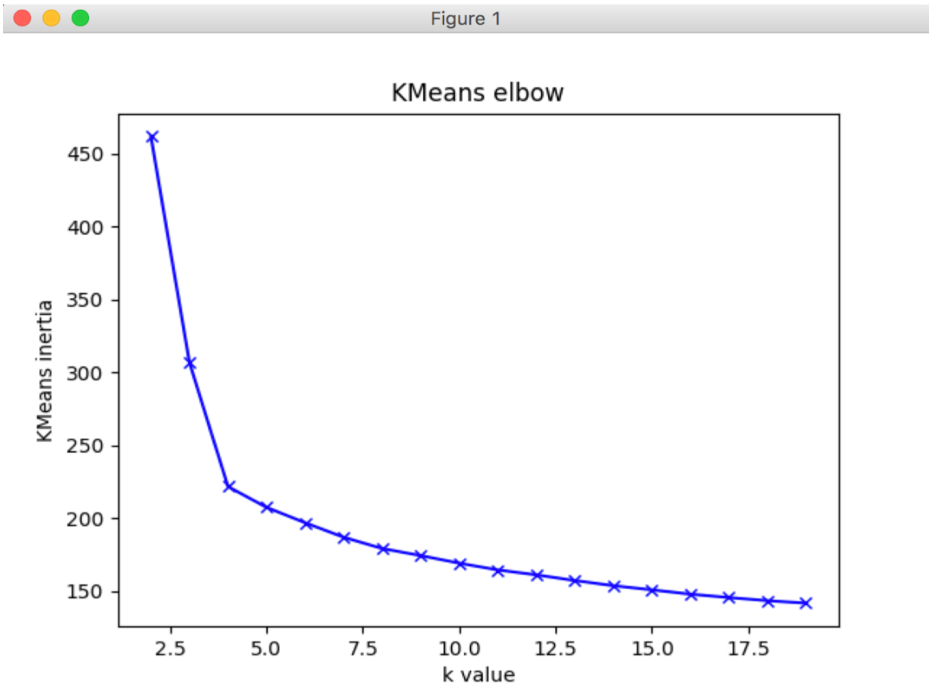
- For project 2, I have performed random and stratified sampling on NBA (National Basketball Association) statistics data. I also performed dimensionality reduction on the data and then visualized the data through various methods of data decimation and dimensionality reduction. For this, I have used various libraries provided by python like sklearn, numpy, pandas etc. For client server connection between python and d3.js I have used Flask.

- I used '**sample**' method of python which reduces the data to given number using random sampling.
  Example: Below code performed data decimation with random sampling to reduce size from 836 to 500.

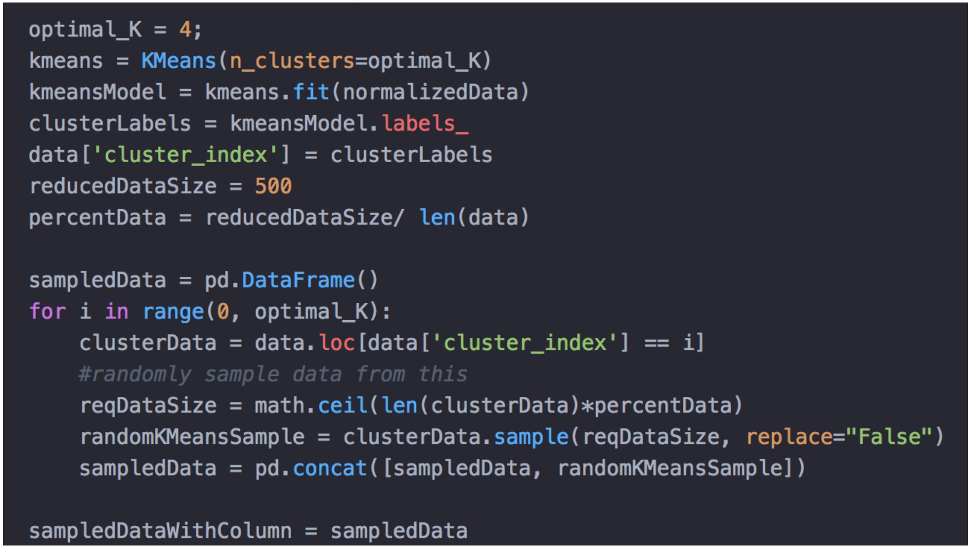


```
randomSampledData = data.sample(500, replace="False")
```

To perform stratified sampling, we need to first cluster the data in strata.
Using K-means clustering method, I plotted graph for various values of K with inertia for that particular value, which represents sum of squared difference for error. Using elbow method, I got the optimal value of K as 4. Using which, I performed K-means clustering on the data. Using the clustered data, I have performed stratified sampling to reduce the data size to required number.

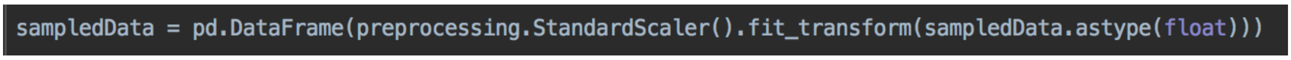


For example: To reduce the data by 50% I have taken 50% data from each of the 4 clusters. After performing K-means I have added the cluster index to identify the group to which data belongs which will help in visualizing the data in better manner.

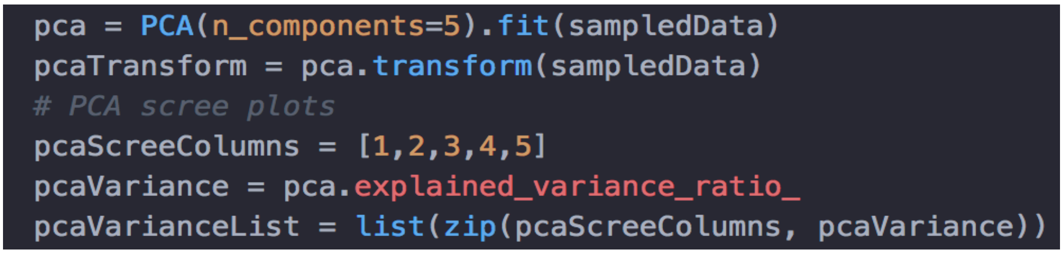


```
optimal_K = 4;
kmeans = KMeans(n_clusters=optimal_K)
kmeansModel = kmeans.fit(normalizedData)
clusterLabels = kmeansModel.labels_
data['cluster_index'] = clusterLabels
reducedDataSize = 500
percentData = reducedDataSize/ len(data)

sampledData = pd.DataFrame()
for i in range(0, optimal_K):
    clusterData = data.loc[data['cluster_index'] == i]
    #randomly sample data from this
    reqDataSize = math.ceil(len(clusterData)*percentData)
    randomKMeansSample = clusterData.sample(reqDataSize, replace="False")
    sampledData = pd.concat([sampledData, randomKMeansSample])

sampledDataWithColumn = sampledData
```

- For dimension reduction, I applied principal component analysis on the decimated standardized data. Using PCA I plotted scree plot for 5 PCA components with variance ratio and study the bias introduced in the data.
  I also standardized the data to compare attributes with different units and scales.



```
sampledData = pd.DataFrame(preprocessing.StandardScaler().fit_transform(sampledData.astype(float)))
```



```
pca = PCA(n_components=5).fit(sampledData)
pcaTransform = pca.transform(sampledData)
# PCA scree plots
pcaScreeColumns = [1,2,3,4,5]
pcaVariance = pca.explained_variance_ratio_
pcaVarianceList = list(zip(pcaScreeColumns, pcaVariance))
```

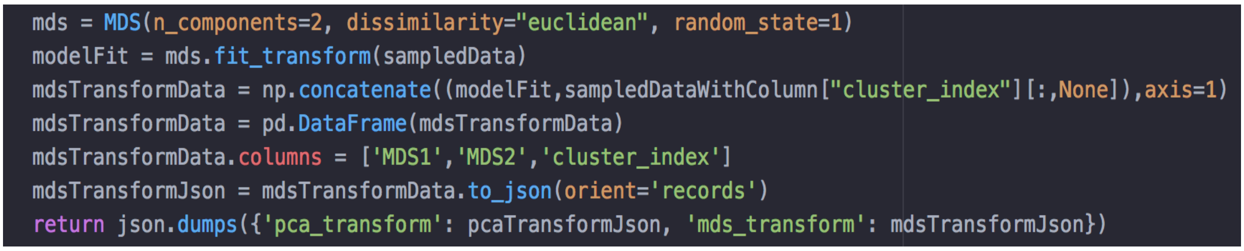
To calculate percent variance for the PCA we have check the variance values by taking pca.explained_variance_

Variance for top 3 is: 9.26090663, 2.81102671, 1.23237138

% variance for PC 1 = (9.26090663) / (9.26090663+ 2.81102671 + 1.23237138)

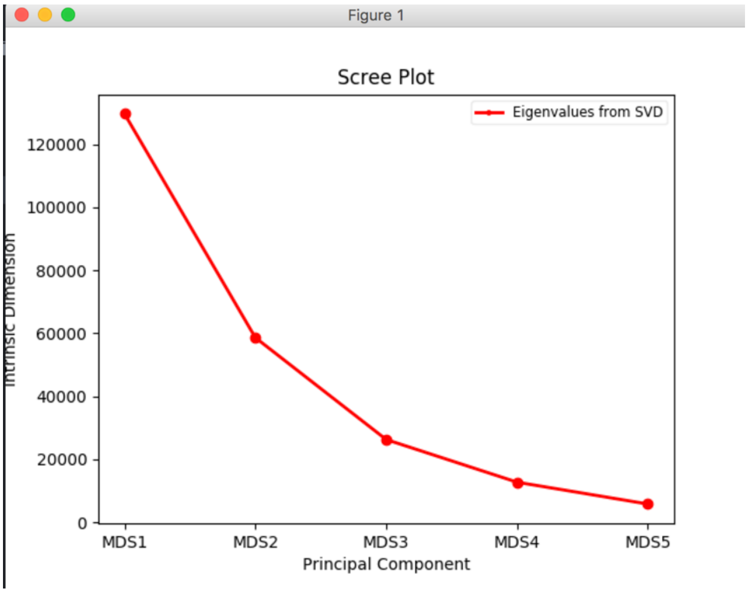
% variance for PC 1 = 70%

- I plotted PCA 2D scatterplot using two principal components using PCA fit and transform method supported by python.
  Similarly, I also plotted 2D scatterplot using MDS mechanism i.e. using multidimensional scaling by finding the Euclidean distances bet`en the data.

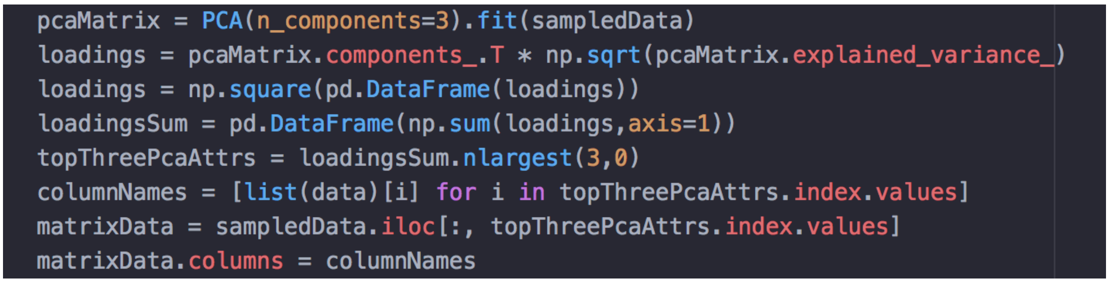


```
mds = MDS(n_components=2, dissimilarity="euclidean", random_state=1)
modelFit = mds.fit_transform(sampledData)
mdsTransformData = np.concatenate((modelFit,sampledDataWithColumn["cluster_index"][:,None]),axis=1)
mdsTransformData = pd.DataFrame(mdsTransformData)
mdsTransformData.columns = ['MDS1','MDS2','cluster_index']
mdsTransformJson = mdsTransformData.to_json(orient='records')
return json.dumps({'pca_transform': pcaTransformJson, 'mds_transform': mdsTransformJson})
```

I also plotted MDS stress chart to analyze the stress values after scaling for different number of components.



- To visualize scatterplot matrix, I identified 3 highest loaded PCA attributes and then plot scatterplot matrix for these three attributes.



```
pcaMatrix = PCA(n_components=3).fit(sampledData)
loadings = pcaMatrix.components_.T * np.sqrt(pcaMatrix.explained_variance_)
loadings = np.square(pd.DataFrame(loadings))
loadingsSum = pd.DataFrame(np.sum(loadings,axis=1))
topThreePcaAttrs = loadingsSum.nlargest(3,0)
columnNames = [list(data)[i] for i in topThreePcaAttrs.index.values]
matrixData = sampledData.iloc[:, topThreePcaAttrs.index.values]
matrixData.columns = columnNames
```

References:

- Code sample template given by TA
- http://bl.ocks.org/owendall/96f75a508c8521e407e798659ddfff51
- https://www.w3schools.com/css/css3_buttons.asp
- https://bl.ocks.org/mbostock/4063663
- Discussed with Pooja Agarwal-classmate