Name: Pulkit Kalia

SBU ID: 112504287

MINI PROJECT 2

Tasks done:

1. Implemented random and stratified sampling
2. Implemented K-means algorithm and chose the optimal value of k using elbow graph.
3. Found intrinsic dimensionality of the data (random and stratified) using PCA.
4. Produced scree plot visualization (for random and stratified data).
5. Plotted MDS (for random and stratified data).
6. Obtained the three attributes with highest PCA loadings.
7. Visualized the data (random and stratified) projected into the top two PCA vectors via 2D scatterplot.
8. Visualized the data via MDS (Euclidian & correlation distance) in 2D scatterplots for random and stratified data.
9. Visualized scatterplot matrix of the three highest PCA loaded attributes(for random and stratified data).

Implementation:

1. Random points -

```
random_points=500
random_row=np.asarray(random.sample(range(2000),random_points))
df_random=df.loc[random_row]
```

2. Optimal value of K in K-means

```
distortions = []
K = range(1,20)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(df_normalized)
    kmeanModel.fit(df_normalized)
    distortions.append(kmeanModel.inertia_)
    plt.plot(K, distortions, 'bx-')
    plt.xlabel('k')
```

```
plt.ylabel('Distortion')
plt.show()
```

3. Stratified points-

```
df_stratified=df
kmeanModel = KMeans(n_clusters=3).fit(df_normalized)
kmeanModel.fit(df_normalized)
groups=kmeanModel.labels_
df_stratified["groups"]=groups
stratified_points=500
percentage=stratified_points/len(df)
pd_result=pd.DataFrame()

for i in range(0,3):
x=df_stratified[df_stratified["groups"]==i]
no_random_points=math.ceil(percentage*len(x))
random_row=np.asarray(random.sample(range(len(x)),no_random_points))
points=x.iloc[random_row,:]
pd_result=pd.concat([pd_result,points])
```

4. Eigen values-

```
pca_stratified = PCA(n_components=10).fit(standard_data_stratified)
 eigenvalues=pca_stratified.explained_variance_
```

5. PCA values-

```
pca_random = PCA(n_components=2).fit_transform(df_random)
PCA1=pca_random[:,0]
PCA2=pca_random[:,1]
```

6. Euclidean distance-

mds_data = manifold.MDS(n_components=2, dissimilarity='precomputed')

similarity = pairwise_distances(df_random, metric='euclidean') or similarity = pairwise_distances(df_random, metric='correlation')

components = mds_data.fit_transform(similarity)

data={'X':components[:,0],'Y':components[:,1]}

7. MDS matrix-

   sq_loading = get_intrinsic_dimensionality(df_random, 3)

   top_columns = sorted(range(len(sq_loading)), key=lambda k: sq_loading[k], reverse=True)[:3]

   def get_intrinsic_dimensionality(data, k):

   std_data = StandardScaler().fit_transform(data);

   cov_mat = np.cov(std_data.T)

   eigenValues, eigenVectors = np.linalg.eig(cov_mat)

   squaredLoadings = []

   ftrCount = len(eigenVectors)

   for ftrId in range(0, ftrCount):

   loadings = 0

   for compId in range(0, k):

   loadings = loadings + eigenVectors[compId][ftrId] * eigenVectors[compId][ftrId]

   squaredLoadings.append(loadings)

   return squaredLoadings

Observations:

1. Random data has more bias then stratified sampling when compared to the original data.
2. Eigen values from random data has big variance when run multiple times, whereas the eigen values from stratified data fairly remains the same.
3. PCA2 in random data has more range than in stratified data.
4. MDS(Euclidean) data shows clearly segregated groups in stratified data.
5. MDS(Correlation) data is very compact and closely related.

Alternative solutions:

1. Task 1: To randomize the data, either random function from pandas can be used to random numbers can be generated from 0 to nrow(df) and then these values as index, can be used to extract data from the data.
2. Task 2 : To find Eigen values, either the correlation matrix can be used or variance_explained_ variable from PCA object.

Youtube Video link:

https://youtu.be/OnWOU4fVJeA

References:

1. TA's sample code(Code from Ayush Kumar's GitHub repo).
2. https://bl.ocks.org/mbostock/4063663
3. https://pythonprogramminglanguage.com/kmeans-elbow-method/
4. https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/multivariate/how-to/factor-analysis/methods-and-formulas/methods-and-formulas/