**VISUALIZATION**

**ASSGINMENT 2**

SUBMITTED BY

JAYAKRISHNA ANVESH PAIDY

(112675567)

**DESCRIPTION:**

The dataset used for the project is Hospital Inpatient Discharge record for the year 2017 in New York. The data consists of 15 attributes of both numerical and categorical type. As part of this assignment we have done Dimensionality reduction task using the PCA and MDS. Data processing is done in the back end using python and visualization with d3 js

As Part of **Task1**,

I have the random sampling and removed 75% of the data. For this I have used binomial distribution to pick data.

**CODE:**

This python code picks the data sample of required size from the given data sample.

def pick\_random\_sample(data,reqd\_length):

    # picking 25 % of data by random sampling

    p=reqd\_length/len(data)

    n=len(data)

    random\_sample=[]

    rand=np.random.binomial(1,p,n)

    k=0

    for i in rand:

        if(i==1):

            random\_sample.append(data[k].tolist())

            k+=1

    return random\_sample

I have implemented the stratified sampling by using the KMeans clustering and picking the proportionate amount of data from each cluster at random. For the given data I have plotted the graph(Distortions vs Number of clusters) and obtained a cluster size 4.

**Code:**

for i in range(1, 10):

        km = KMeans(

            n\_clusters=i, init='random',

            n\_init=10, max\_iter=300,

            tol=1e-04, random\_state=0

        )

        km.fit(data)

        distortions.append(km.inertia\_)

for i in range(len(sampled\_data)):

        cluster\_length=len(sampled\_data[i])

        req\_cluster\_length=cluster\_length\*req\_sample\_length//sample\_length

        sample=pick\_random\_sample(sampled\_data[i],req\_cluster\_length)

        reqd\_sample.extend(sample)

For this **Task2:**

Performed PCA, computed the data correlation with respect to principal components and plotted the Scree plot and found the intrinsic dimensionality which explains the 75% variance. We computed this for the original dataset as well as the sampled datasets.

Code for computing PCA:

std\_df=StandardScaler().fit\_transform(df)

    pca = PCA(n\_components=len(std\_df[0]))

    principalComponents = pca.fit\_transform(std\_df)

    #print(principalComponents)

    columns=[str(i) for i in range(1,len(std\_df[0])+1)]

    principalDataframe = pd.DataFrame(data = principalComponents,columns=columns )

    i=0

    percent\_variance = np.round(pca.explained\_variance\_ratio\_\* 100, decimals =2)

    for csum in pca.explained\_variance\_ratio\_.cumsum():

        if(csum<0.75):

            i+=1

On sampling the data a bias is introduced and the intrinsic dimensionality for the give data has reduced from 11 to 10 for both the samples.

Also Obtained the top 3 attributes with highest PCA loadings by computing the squared sum of the values (weights) associated with the attributes for the original as well as sampled data this attributes also changed.

For **Task3 (Visualization):**

Visualized the data obtained from the top principle components using d3 js via 2d scatter plot. Code for the visualization.

 svg.selectAll(".dot")

      .data(gdata)

    .enter().append("circle")

      .attr("class", "point")

      .attr("cx", function(d, i) { return xscale(d.p1) })

      .attr("cy", function(d) {

        return yscale(d.p2) })

      .attr("r", 3)

      .style("fill", function(d) { return get\_color();});

Also visualized the data via computed MDS via Euclidian & correlation distance via Scatter plot for the top two components.

emb = MDS(n\_components=2,max\_iter=100,n\_jobs=-1,dissimilarity=func)

    if(func=='precomputed'):

        df=compute\_dissim(df)

    trans = emb.fit\_transform(df)

    trans=np.reshape(trans,[trans.shape[1],trans.shape[0]])

dissim = np.sqrt(1 - np.abs(np.corrcoef(data)))

    dissimilarity = np.triu(dissim.T, k=1) + np.tril(dissim)

    return dissimilarity

As part of this task, Computed the Scatter matrix plot for the top three highest PCA loaded attribues previously loaded.

For this assignment I have used flask for to handling the requests in python.

@app.route('/',methods=['GET'])

def init():

   return render\_template("index.html")

@app.route('/data',methods=['GET'])

def get\_data():

   data=s.handle\_operations()

   return jsonify(data)

**Note:** For this assignment, I have used the same scale for all the data samples inorder the clearly differentiate their properties.

youtube link: https://youtu.be/dtGGQh0IHOs

Run python file app.py

I have computed all the caluclations at load time so it might take time to start the server