

**DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING (DATA SCIENCE)**

**DATA VISUALIZATION LABORATORY MANUAL- 21ADL76**

**VII SEMESTER –BE**

**Prepared By,**

**Mrs. Shwetha S Shetty**

**Assistant Professor**

**Department of Information Science & Engineering**

**Department of Information Science &Engineering**

**SAHYADRI College of Engineering &Management Adyar, Mangaluru**

**575007**

**1. Load the specified dataset**

**a) Using numpy and compute mean, median, variance and Standard deviation and illustrate**

**Indexing, Slicing, Splitting, Iterating, Filtering, Sorting, Combining, and Reshaping**

**b) Using pandas and compute mean, median, variance and Standard deviation and**

**Illustrate Indexing, Slicing, Iterating, Filtering, Sorting and Reshaping.**

Import numpy as mp

Dataset=np.genfromtxt(‘c:/users/Sahyadri/downloads/normal distribution.csv:delimiter=’,’)

#Mean

#np.mean of third now

np.mean(dataset[2])

#mean of the last column

Np.mean(dataset[:-1])

#mean of the insersection of first 3 rows & first 3 columns

Np.mean(dataset[0:3,0:3])

# median

#Median of last row

Np.median(dataset[-1])

#median of last 3 columns

Np.median(dataset[:,-3])

# median of each row

Np.median(dataset ,axis=1)

#varience

#varience of each column

Np.var(dataset[-26,:2]

#standard deviation

#standard deviation for the dataset

np.std(dataset)

**#Indexing**

Import numpy as np

Dataxt=1 nnp.genfromtxt(‘c:user/admin/normal\_distribution\_spittable-csv’,\delimeter=’,’)

#indexing the 1st row of the dataset (1st row)

Dataset[-1]

#indexing the last element if the dataset (2nd row)

Dataset[-1]

#indexing the first value of the firs row(1st row,1st value)

Dataset[0][0]

#indexing the last value of the second to last now (we want to usethe combined acess syntax here)

**#Slicing**

#slicing an intersection of the elements (2\*2 of the first 2 rows and 1st 2 columns

Dataset[1:3,1:2]

#slicing every second element of 5th row

Dataset[4:2]

#reversing the entry order, selecting the first 2 rows in reversed order dataset[-1,::-1]

**#Splitting**

#splitting our dataset horizontally on inorder 2

Ver\_splits=np.vsplit(hur-splits[0],(2,))

#requested sub selection of our dataset which has only half the amount of rows and column

Print(“Dataset”,dataset-sp shape)

Print(“subset”,ver\_splits[0],shape)

**#Iterating**

#Iterating over dataset c each value in each now)

Ever\_index=0

For x in np.mditer(dataset)

Print (x,ever-index)

curv-index+=1

# iterating over whole dataset with index matching the position in the dataset.

For index,value in mp.ndenumerats(dataset):

Print(index,value)

**#Filtering**

Dataset [dataset>105]

np.extract((dataset>90)&(dataset<95),dataset)

rows.cols=np.where labs(datset-100)<1)

[[rows[index],cols[index]]for(index]]for(index,-)in np.ndenemeterate(rows)]

**#sorting**

np.sort(dataset)

np.sort(dataset,axis=0)

index\_sorted=np.argsort(dataset{0])

dataset[0][index\_sorted]

**#Combining**

Thirds=np.hspirit(dataset,(3))

Halfed\_first=np.vsplit(thirds[0],(2))

halfed\_first[0]

first\_col=np.vstack([halfed\_first[0],halfed\_first[1]])

first\_col

first\_seond\_col=np.hstack([first\_col,thirds[1]])

first\_second\_col

np.hstack([first\_second\_col,thirds[0])

#Reshaping

np.reshape(dataset,(1,-1)

dataset.reshape(-1,2)

Output:-

**2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the**

**(Note: Import Matplotlib)**

1. **Visualizing through a Line Plot**

# Import statements

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

%matplotlib inline

# load datasets

google = pd.read\_csv('../../Datasets/GOOGL\_data.csv')

facebook = pd.read\_csv('../../Datasets/FB\_data.csv')

apple = pd.read\_csv('../../Datasets/AAPL\_data.csv')

amazon = pd.read\_csv('../../Datasets/AMZN\_data.csv')

microsoft = pd.read\_csv('../../Datasets/MSFT\_data.csv')

# Create figure

plt.figure(figsize=(16, 8), dpi=300)

# Plot data

plt.plot('date', 'close', data=google, label='Google')

plt.plot('date', 'close', data=facebook, label='Facebook')

plt.plot('date', 'close', data=apple, label='Apple')

plt.plot('date', 'close', data=amazon, label='Amazon')

plt.plot('date', 'close', data=microsoft, label='Microsoft')

# Specify ticks for x- and y-axis

plt.xticks(np.arange(0, 1260, 40), rotation=70)

plt.yticks(np.arange(0, 1450, 100))

# Add title and label for y-axis

plt.title('Stock trend', fontsize=16)

plt.ylabel('Closing price in $', fontsize=14)

# Add grid

plt.grid()

# Add legend

plt.legend()

# Show plot

plt.show()

1. **Creating a Bar Plot**

# Import statements

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

# Load dataset

movie\_scores = pd.read\_csv('../../Datasets/movie\_scores.csv')

# Create figure

plt.figure(figsize=(10, 5), dpi=300)

# Create bar plot

pos = np.arange(len(movie\_scores['MovieTitle']))

width = 0.3

plt.bar(pos - width / 2, movie\_scores['Tomatometer'], width, label='Tomatometer')

plt.bar(pos + width / 2, movie\_scores['AudienceScore'], width, label='Audience Score')

# Specify ticks

plt.xticks(pos, rotation=10)

plt.yticks(np.arange(0, 101, 20))

# Get current Axes for setting tick labels and horizontal grid

ax = plt.gca()

# Set tick labels

ax.set\_xticklabels(movie\_scores['MovieTitle'])

ax.set\_yticklabels(['0%', '20%', '40%', '60%', '80%', '100%'])

# Add minor ticks for y-axis in the interval of 5

ax.set\_yticks(np.arange(0, 100, 5), minor=True)

# Add major horizontal grid with solid lines

ax.yaxis.grid(which='major')

# Add minor horizontal grid with dashed lines

ax.yaxis.grid(which='minor', linestyle='--')

# Add title

plt.title('Movie comparison')

# Add legend

plt.legend()

# Show plot

plt.show()

1. **Creating a Stacked Bar Plot to visualize a specified parameter**

# Import statements

import pandas as sb

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

# Load dataset

bills = sns.load\_dataset('tips')

days = ['Thur', 'Fri', 'Sat', 'Sun']

days\_range = np.arange(len(days))

smoker = ['Yes', 'No']

bills\_by\_days = [bills[bills['day'] == day] for day in days]

bills\_by\_days\_smoker = [[bills\_by\_days[day][bills\_by\_days[day]['smoker'] == s] for s in smoker] for day in days\_range]

total\_by\_days\_smoker = [[bills\_by\_days\_smoker[day][s]['total\_bill'].sum() for s in range(len(smoker))] for day in days\_range]

totals = np.asarray(total\_by\_days\_smoker)

# Create figure

plt.figure(figsize=(10, 5), dpi=300)

# Create stacked bar plot

plt.bar(days\_range, totals[:, 0], label='Smoker')

plt.bar(days\_range, totals[:, 1], bottom=totals[:, 0], label='Non-smoker')

# Add legend

plt.legend()

# Add labels and title

plt.xticks(days\_range)

ax = plt.gca()

ax.set\_xticklabels(days)

ax.yaxis.grid()

plt.ylabel('Daily total sales in $')

plt.title('Restaurant performance')

# Show plot

plt.show()

1. **Comparing specific parameters using Stacked Area Chart.**

## # Import statements

## import pandas as pd

## import numpy as np

## import matplotlib.pyplot as plt

## %matplotlib inline

## # Load dataset

## sales = pd.read\_csv('../../Datasets/smartphone\_sales.csv')

## # Create figure

## plt.figure(figsize=(10, 6), dpi=300)

## # Create stacked area chart

## labels = sales.columns[2:]

## plt.stackplot('Quarter', 'Apple', 'Samsung', 'Huawei', 'Xiaomi', 'OPPO', data=sales, labels=labels)

## # Add legend

## plt.legend()

## # Add labels and title

## plt.xlabel('Quarters')

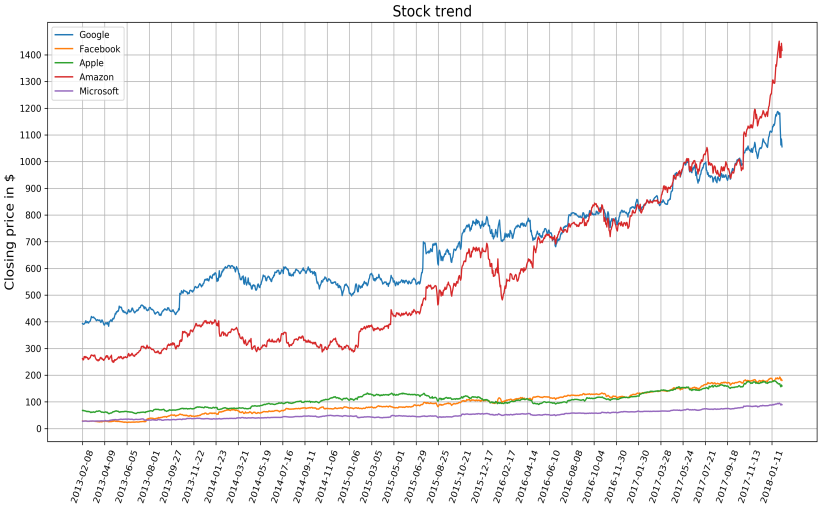
## plt.ylabel('Sales units in thousands')

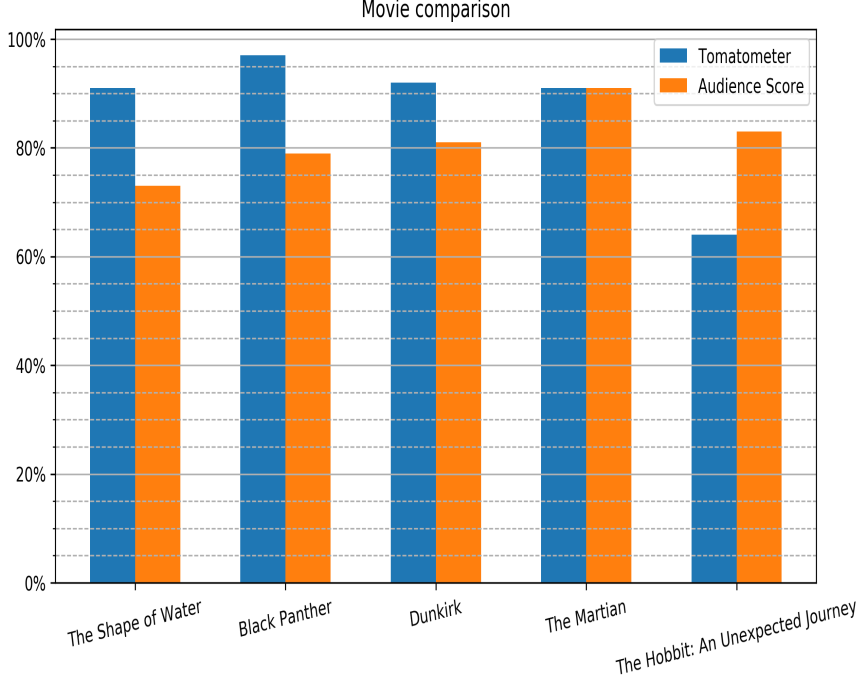
## plt.title('Smartphone sales units')

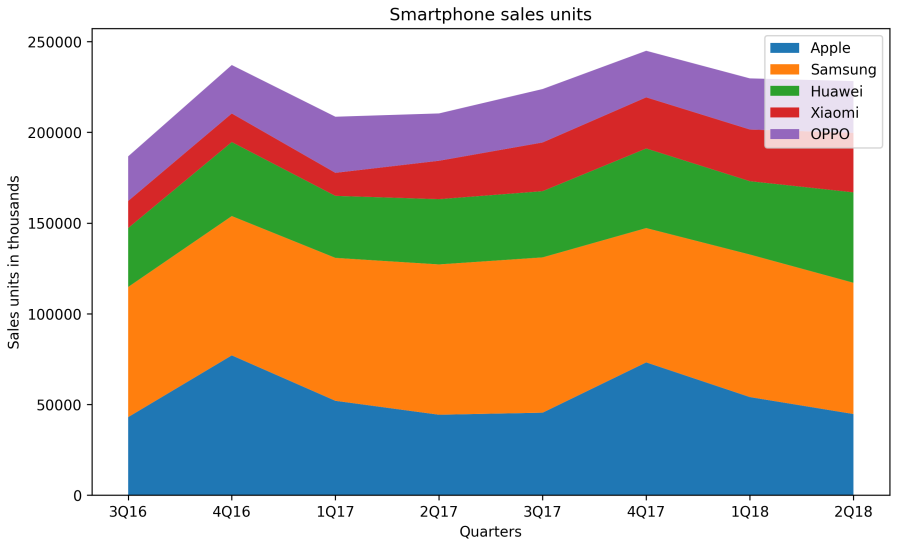
## # Show plot

## plt.show()

Output:-







**3. For a given set of training data examples stored in a .CSV file, implement and demonstrate**

**the (Note: Import Matplotlib)**

1. **Histogram and a Box Plot to visualize the given parameter**

# Import statements

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

# IQ samples

iq\_scores = [126, 89, 90, 101, 102, 74, 93, 101, 66, 120, 108, 97, 98,

105, 119, 92, 113, 81, 104, 108, 83, 102, 105, 111, 102, 107,

103, 89, 89, 110, 71, 110, 120, 85, 111, 83, 122, 120, 102,

84, 118, 100, 100, 114, 81, 109, 69, 97, 95, 106, 116, 109,

114, 98, 90, 92, 98, 91, 81, 85, 86, 102, 93, 112, 76,

89, 110, 75, 100, 90, 96, 94, 107, 108, 95, 96, 96, 114,

93, 95, 117, 141, 115, 95, 86, 100, 121, 103, 66, 99, 96,

111, 110, 105, 110, 91, 112, 102, 112, 75]

# Create figure

plt.figure(figsize=(6, 4), dpi=150)

# Create histogram

plt.hist(iq\_scores, bins=10)

plt.axvline(x=100, color='r')

plt.axvline(x=115, color='r', linestyle= '--')

plt.axvline(x=85, color='r', linestyle= '--')

# Add labels and title

plt.xlabel('IQ score')

plt.ylabel('Frequency')

plt.title('IQ scores for a test group of a hundred adults')

# Show plot

plt.show()

Create a box plot to visualize the same IQ scores. Add labels and a title.

# Create figure

plt.figure(figsize=(6, 4), dpi=150)

# Create histogram

plt.boxplot(iq\_scores)

# Add labels and title

ax = plt.gca()

ax.set\_xticklabels(['Test group'])

plt.ylabel('IQ score')

plt.title('IQ scores for a test group of a hundred adults')

# Show plot

plt.show()

group\_a = [118, 103, 125, 107, 111, 96, 104, 97, 96, 114, 96, 75, 114,

107, 87, 117, 117, 114, 117, 112, 107, 133, 94, 91, 118, 110,

117, 86, 143, 83, 106, 86, 98, 126, 109, 91, 112, 120, 108,

111, 107, 98, 89, 113, 117, 81, 113, 112, 84, 115, 96, 93,

128, 115, 138, 121, 87, 112, 110, 79, 100, 84, 115, 93, 108,

130, 107, 106, 106, 101, 117, 93, 94, 103, 112, 98, 103, 70,

139, 94, 110, 105, 122, 94, 94, 105, 129, 110, 112, 97, 109,

121, 106, 118, 131, 88, 122, 125, 93, 78]

group\_b = [126, 89, 90, 101, 102, 74, 93, 101, 66, 120, 108, 97, 98,

105, 119, 92, 113, 81, 104, 108, 83, 102, 105, 111, 102, 107,

103, 89, 89, 110, 71, 110, 120, 85, 111, 83, 122, 120, 102,

84, 118, 100, 100, 114, 81, 109, 69, 97, 95, 106, 116, 109,

114, 98, 90, 92, 98, 91, 81, 85, 86, 102, 93, 112, 76,

89, 110, 75, 100, 90, 96, 94, 107, 108, 95, 96, 96, 114,

93, 95, 117, 141, 115, 95, 86, 100, 121, 103, 66, 99, 96,

111, 110, 105, 110, 91, 112, 102, 112, 75]

group\_c = [108, 89, 114, 116, 126, 104, 113, 96, 69, 121, 109, 102, 107,

122, 104, 107, 108, 137, 107, 116, 98, 132, 108, 114, 82, 93,

89, 90, 86, 91, 99, 98, 83, 93, 114, 96, 95, 113, 103,

81, 107, 85, 116, 85, 107, 125, 126, 123, 122, 124, 115, 114,

93, 93, 114, 107, 107, 84, 131, 91, 108, 127, 112, 106, 115,

82, 90, 117, 108, 115, 113, 108, 104, 103, 90, 110, 114, 92,

101, 72, 109, 94, 122, 90, 102, 86, 119, 103, 110, 96, 90,

110, 96, 69, 85, 102, 69, 96, 101, 90]

group\_d = [ 93, 99, 91, 110, 80, 113, 111, 115, 98, 74, 96, 80, 83,

102, 60, 91, 82, 90, 97, 101, 89, 89, 117, 91, 104, 104,

102, 128, 106, 111, 79, 92, 97, 101, 106, 110, 93, 93, 106,

108, 85, 83, 108, 94, 79, 87, 113, 112, 111, 111, 79, 116,

104, 84, 116, 111, 103, 103, 112, 68, 54, 80, 86, 119, 81,

84, 91, 96, 116, 125, 99, 58, 102, 77, 98, 100, 90, 106,

109, 114, 102, 102, 112, 103, 98, 96, 85, 97, 110, 131, 92,

79, 115, 122, 95, 105, 74, 85, 85, 95]

Create a box plot for each of the IQ scores of different test groups. Add labels and a title.

# Create figure

plt.figure(figsize=(6, 4), dpi=150)

# Create histogram

plt.boxplot([group\_a, group\_b, group\_c, group\_d])

# Add labels and title

ax = plt.gca()

ax.set\_xticklabels(['Group A', 'Group B', 'Group C', 'Group D'])

plt.ylabel('IQ score')

plt.title('IQ scores for different test groups')

# Show plot

plt.show()

**b )Scatter Plot with Marginal Histograms**

# Import statements

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

# Load dataset

data = pd.read\_csv('/…./anage\_data.csv')

# Preprocessing

longevity = 'Maximum longevity (yrs)'

mass = 'Body mass (g)'

data = data[np.isfinite(data[longevity]) & np.isfinite(data[mass])]

# Sort according to class

aves = data[data['Class'] == 'Aves']

aves = aves[aves[mass] < 20000]

# Create figure

fig = plt.figure(figsize=(8, 8), dpi=150, constrained\_layout=True)

# Create gridspec

gs = fig.add\_gridspec(4, 4)

# Specify subplots

histx\_ax = fig.add\_subplot(gs[0, :-1])

histy\_ax = fig.add\_subplot(gs[1:, -1])

scatter\_ax = fig.add\_subplot(gs[1:, :-1])

# Create plots

scatter\_ax.scatter(aves[mass], aves[longevity])

histx\_ax.hist(aves[mass], bins=20, density=True)

histx\_ax.set\_xticks([])

histy\_ax.hist(aves[longevity], bins=20, density=True, orientation='horizontal')

histy\_ax.set\_yticks([])

# Add labels and title

plt.xlabel('Body mass in grams')

plt.ylabel('Maximum longevity in years')

fig.suptitle('Scatter plot with marginal histograms')

# Show plot

plt.show()

**C) Plotting Multiple Images in a Grid.**

# Import statements

import os

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

# Enable inline plotting for Jupyter Notebooks

%matplotlib inline

# Load images

image\_folder = 'C:/Users/……../Desktop/images'

img\_filenames = sorted(os.listdir(image\_folder))

# Ensure only image files are selected

# (you can adjust this to check for specific extensions if needed)

img\_filenames = [f for f in img\_filenames if f.endswith(('.png', '.jpg', '.jpeg'))]

# Load the images

imgs = [mpimg.imread(os.path.join(image\_folder, img\_filename)) for img\_filename in img\_filenames]

# Create a subplot grid of 2x2

fig, axes = plt.subplots(2, 2)

# Set figure size and DPI

fig.set\_size\_inches(6, 6)

fig.dpi = 150

axes = axes.ravel()

# Specify labels for each image

labels = ['coast', 'beach', 'building', 'city at night']

# Plot images in the subplots

for i in range(len(imgs)):

axes[i].imshow(imgs[i])

axes[i].set\_xticks([]) # Remove x-axis ticks

axes[i].set\_yticks([]) # Remove y-axis ticks

axes[i].set\_xlabel(labels[i]) # Add labels to the subplots

# Display the plot

plt.tight\_layout()

plt.show()

**d) Give Scatter Plot to Visualize Correlation**

import numpy as np

import matplotlib.pyplot as plt

# IQ scores for different test groups

group\_a = [118, 103, 125, 107, 111, 96, 104, 97, 96, 114, 96, 75, 114,

107, 87, 117, 117, 114, 117, 112, 107, 133, 94, 91, 118, 110,

117, 86, 143, 83, 106, 86, 98, 126, 109, 91, 112, 120, 108,

111, 107, 98, 89, 113, 117, 81, 113, 112, 84, 115, 96, 93,

128, 115, 138, 121, 87, 112, 110, 79, 100, 84, 115, 93, 108,

130, 107, 106, 106, 101, 117, 93, 94, 103, 112, 98, 103, 70,

139, 94, 110, 105, 122, 94, 94, 105, 129, 110, 112, 97, 109,

121, 106, 118, 131, 88, 122, 125, 93, 78]

group\_b = [126, 89, 90, 101, 102, 74, 93, 101, 66, 120, 108, 97, 98,

105, 119, 92, 113, 81, 104, 108, 83, 102, 105, 111, 102, 107,

103, 89, 89, 110, 71, 110, 120, 85, 111, 83, 122, 120, 102,

84, 118, 100, 100, 114, 81, 109, 69, 97, 95, 106, 116, 109,

114, 98, 90, 92, 98, 91, 81, 85, 86, 102, 93, 112, 76,

89, 110, 75, 100, 90, 96, 94, 107, 108, 95, 96, 96, 114,

93, 95, 117, 141, 115, 95, 86, 100, 121, 103, 66, 99, 96,

111, 110, 105, 110, 91, 112, 102, 112, 75]

group\_c = [108, 89, 114, 116, 126, 104, 113, 96, 69, 121, 109, 102, 107,

122, 104, 107, 108, 137, 107, 116, 98, 132, 108, 114, 82, 93,

89, 90, 86, 91, 99, 98, 83, 93, 114, 96, 95, 113, 103,

81, 107, 85, 116, 85, 107, 125, 126, 123, 122, 124, 115, 114,

93, 93, 114, 107, 107, 84, 131, 91, 108, 127, 112, 106, 115,

82, 90, 117, 108, 115, 113, 108, 104, 103, 90, 110, 114, 92,

101, 72, 109, 94, 122, 90, 102, 86, 119, 103, 110, 96, 90,

110, 96, 69, 85, 102, 69, 96, 101, 90]

group\_d = [ 93, 99, 91, 110, 80, 113, 111, 115, 98, 74, 96, 80, 83,

102, 60, 91, 82, 90, 97, 101, 89, 89, 117, 91, 104, 104,

102, 128, 106, 111, 79, 92, 97, 101, 106, 110, 93, 93, 106,

108, 85, 83, 108, 94, 79, 87, 113, 112, 111, 111, 79, 116,

104, 84, 116, 111, 103, 103, 112, 68, 54, 80, 86, 119, 81,

84, 91, 96, 116, 125, 99, 58, 102, 77, 98, 100, 90, 106,

109, 114, 102, 102, 112, 103, 98, 96, 85, 97, 110, 131, 92,

79, 115, 122, 95, 105, 74, 85, 85, 95]

# Create scatter plots to visualize correlations between different groups

plt.figure(figsize=(12, 8), dpi=150)

# Scatter plot for Group A vs Group B

plt.subplot(2, 2, 1)

plt.scatter(group\_a, group\_b, color='blue')

plt.xlabel('Group A')

plt.ylabel('Group B')

plt.title('Group A vs Group B')

# Scatter plot for Group A vs Group C

plt.subplot(2, 2, 2)

plt.scatter(group\_a, group\_c, color='green')

plt.xlabel('Group A')

plt.ylabel('Group C')

plt.title('Group A vs Group C')

# Scatter plot for Group A vs Group D

plt.subplot(2, 2, 3)

plt.scatter(group\_a, group\_d, color='red')

plt.xlabel('Group A')

plt.ylabel('Group D')

plt.title('Group A vs Group D')

# Scatter plot for Group B vs Group C

plt.subplot(2, 2, 4)

plt.scatter(group\_b, group\_c, color='purple')

plt.xlabel('Group B')

plt.ylabel('Group C')

plt.title('Group B vs Group C')

# Adjust layout and show plot

plt.tight\_layout()

plt.show()

**4. For a given set of training data examples stored in a CSV file, implement to generate surface temperature analysis and demonstrate using heat map to find patterns.**

%matplotlib inline

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns**.**set()

data **=** pd**.**read\_csv("../../Datasets/northern\_surface\_temperature.csv", index\_col**=**['Year'])

data **=** data**.**transpose()

heat\_colormap **=** sns**.**diverging\_palette(240, 15, s**=**99, as\_cmap**=**True)

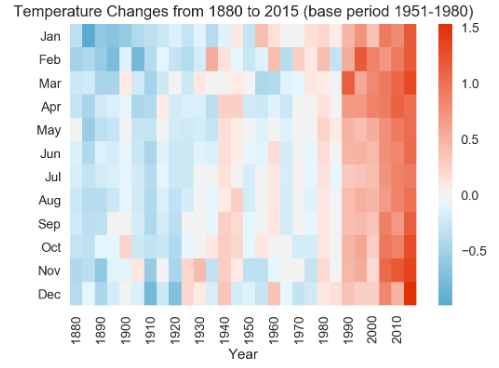
plt**.**figure(dpi**=**200)

sns**.**heatmap(data**.**iloc[:, ::5], cmap**=**heat\_colormap, center**=**0)

plt**.**title("Temperature Changes from 1880 to 2015 (base period 1951-1980)")

plt**.**show()

OUTPUT:



**5. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

def kernel(point, xmat, k):

m,n = np.shape(xmat)

weights = np.mat(np.eye((m)))

for j in range(m):

diff = point - X[j]

weights[j,j] = np.exp(diff\*diff.T/(-2.0\*k\*\*2))

return weights

def localWeight(point, xmat, ymat, k):

wei = kernel(point,xmat,k)

W = (X.T\*(wei\*X)).I\*(X.T\*(wei\*ymat.T))

return W

def localWeightRegression(xmat, ymat, k):

m,n = np.shape(xmat)

ypred = np.zeros(m)

for i in range(m):

ypred[i] = xmat[i]\*localWeight(xmat[i],xmat,ymat,k)

return ypred

# load data points

data = pd.read\_csv('10-dataset.csv')

bill = np.array(data.total\_bill)

tip = np.array(data.tip)

#preparing and add 1 in bill

mbill = np.mat(bill)

mtip = np.mat(tip)

m= np.shape(mbill)[1]

one = np.mat(np.ones(m))

X = np.hstack((one.T,mbill.T))

#set k here

ypred = localWeightRegression(X,mtip,0.5)

SortIndex = X[:,1].argsort(0)

xsort = X[SortIndex][:,0]

fig = plt.figure()

ax = fig.add\_subplot(1,1,1)

ax.scatter(bill,tip, color='green')

ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)

plt.xlabel('Total bill')

plt.ylabel('Tip')

plt.show();

Import numpy as mp

Dataset=np.genfromtxt(‘c:/users/Sahyadri/downloads/normal distribution.csv:delimiter=’,’)

#Mean

#np.mean of third now

np.mean(dataset[2])

#mean of the last column

Np.mean(dataset[:-1])

#mean of the insersection of first 3 rows & first 3 columns

Np.mean(dataset[0:3,0:3])

# median

#Median of last row

Np.median(dataset[-1])

#median of last 3 columns

Np.median(dataset[:,-3])

# median of each row

Np.median(dataset ,axis=1)

#varience

#varience of each column

Np.var(dataset[-26,:2]

#standard deviation

#standard deviation for the dataset

np.std(dataset)

#Indexing

Import numpy as np

Dataxt=1 nnp.genfromtxt(‘c:user/admin/normal\_distribution\_spittable-csv’,\delimeter=’,’)

#indexing the 1st row of the dataset (1st row)

Dataset[-1]

#indexing the last element if the dataset (2nd row)

Dataset[-1]

#indexing the first value of the firs row(1st row,1st value)

Dataset[0][0]

#indexing the last value of the second to last now (we want to usethe combined acess syntax here)

#slicing

#slicing an intersection of the elements (2\*2 of the first 2 rows and 1st 2 columns

Dataset[1:3,1:2]

#slicing every second element of 5th row

Dataset[4:2]

#reversing the entry order, selecting the first 2 rows in reversed order dataset[-1,::-1]

#splitting

#splitting our dataset horizontally on inorder 2

Ver\_splits=np.vsplit(hur-splits[0],(2,))

#requested sub selection of our dataset which has only half the amount of rows and column

Print(“Dataset”,dataset-sp shape)

Print(“subset”,ver\_splits[0],shape)

#Iterating

#Iterating over dataset c each value in each now)

Ever\_index=0

For x in np.mditer(dataset)

Print (x,ever-index)

curv-index+=1

# iterating over whole dataset with index matching the position in the dataset.

For index,value in mp.ndenumerats(dataset):

Print(index,value)

#Filtering

Dataset [dataset>105]

np.extract((dataset>90)&(dataset<95),dataset)

rows.cols=np.where labs(datset-100)<1)

[[rows[index],cols[index]]for(index]]for(index,-)in np.ndenemeterate(rows)]

#sorting

np.sort(dataset)

np.sort(dataset,axis=0)

index\_sorted=np.argsort(dataset{0])

dataset[0][index\_sorted]

#Combining

Thirds=np.hspirit(dataset,(3))

Halfed\_first=np.vsplit(thirds[0],(2))

halfed\_first[0]

first\_col=np.vstack([halfed\_first[0],halfed\_first[1]])

first\_col

first\_seond\_col=np.hstack([first\_col,thirds[1]])

first\_second\_col

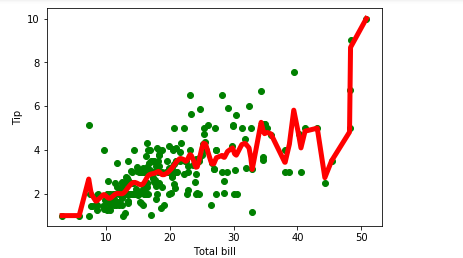
np.hstack([first\_second\_col,thirds[0])

#Reshaping

np.reshape(dataset,(1,-1)

dataset.reshape(-1,2)

Output:-



100.20466135250001

100.4404927375

97.87197312333333

99.18748092

99.08416696500001

[ 98.77910163 97.17512034 98.58782879 100.68449836 101.00170737

97.76908825 101.85002253 100.04756697 102.24292555 99.59514997

100.4955753 99.8860714 99.00647994 98.67276177 102.44376222

96.61933565 104.0968893 100.72023043 98.70877396 99.75008654

104.89344428 101.00634942 98.30543801 99.18748092]

26.72423527090201

4.838197554269257

[ 99.14931546 104.03852715 107.43534677 97.85230675 98.74986914

98.80833412 96.81964892 98.56783189]

[ 94.11176915 99.62387832 104.51786419 97.62787811 93.97853495

98.75108352 106.05042487 100.07721494]

99.14931546

103.83852459

[[97.10439252 99.32066924]

[95.17750125 90.93318132]]

[101.20862522 100.28690912 93.37126331 100.79478953]

[[103.83852459 97.99762409 95.62359311 98.61325194 105.02389857

100.80409326 94.57421727 96.10020311]

[100.07721494 106.05042487 98.75108352 93.97853495 97.62787811

104.51786419 99.62387832 94.11176915]]

[[ 99.14931546 104.03852715 107.43534677 97.85230675 98.74986914

98.80833412 96.81964892 98.56783189]

[ 92.02628776 97.10439252 99.32066924 97.24584816 92.9267508

92.65657752 105.7197853 101.23162942]

[ 95.66253664 95.17750125 90.93318132 110.18889465 98.80084371

105.95297652 98.37481387 106.54654286]

[ 91.37294597 100.96781394 100.40118279 113.42090475 105.48508838

91.6604946 106.1472841 95.08715803]

[101.20862522 103.5730309 100.28690912 105.85269352 93.37126331

108.57980357 100.79478953 94.20019732]

[102.80387079 98.29687616 93.24376389 97.24130034 89.03452725

96.2832753 104.60344836 101.13442416]

[106.71751618 102.97585605 98.45723272 100.72418901 106.39798503

95.46493436 94.35373179 106.83273763]

[ 96.02548256 102.82360856 106.47551845 101.34745901 102.45651798

98.74767493 97.57544275 92.5748759 ]

[105.30350449 92.87730812 103.19258339 104.40518318 101.29326772

100.85447132 101.2226037 106.03868807]

[110.44484313 93.87155456 101.5363647 97.65393524 92.75048583

101.72074646 96.96851209 103.29147111]

[101.3514185 100.37372248 106.6471081 100.61742813 105.0320535

99.35999981 98.87007532 95.85284217]

[ 97.21315663 107.02874163 102.17642112 96.74630281 95.93799169

102.62384733 105.07475277 97.59572169]]

(0, 0) 99.14931546

(0, 1) 104.03852715

(0, 2) 107.43534677

(0, 3) 97.85230675

(0, 4) 98.74986914

(0, 5) 98.80833412

(0, 6) 96.81964892

(0, 7) 98.56783189

(1, 0) 92.02628776

(1, 1) 97.10439252

(1, 2) 99.32066924

(1, 3) 97.24584816

(1, 4) 92.9267508

(1, 5) 92.65657752

(1, 6) 105.7197853

(1, 7) 101.23162942

(2, 0) 95.66253664

(2, 1) 95.17750125

(2, 2) 90.93318132

(2, 3) 110.18889465

(2, 4) 98.80084371

(2, 5) 105.95297652

(2, 6) 98.37481387

(2, 7) 106.54654286

(3, 0) 91.37294597

(3, 1) 100.96781394

(3, 2) 100.40118279

(3, 3) 113.42090475

(3, 4) 105.48508838

(3, 5) 91.6604946

(3, 6) 106.1472841

(3, 7) 95.08715803

(4, 0) 101.20862522

(4, 1) 103.5730309

(4, 2) 100.28690912

(4, 3) 105.85269352

(4, 4) 93.37126331

(4, 5) 108.57980357

(4, 6) 100.79478953

(4, 7) 94.20019732

(5, 0) 102.80387079

(5, 1) 98.29687616

(5, 2) 93.24376389

(5, 3) 97.24130034

(5, 4) 89.03452725

(5, 5) 96.2832753

(5, 6) 104.60344836

(5, 7) 101.13442416

(6, 0) 106.71751618

(6, 1) 102.97585605

(6, 2) 98.45723272

(6, 3) 100.72418901

(6, 4) 106.39798503

(6, 5) 95.46493436

(6, 6) 94.35373179

(6, 7) 106.83273763

(7, 0) 96.02548256

(7, 1) 102.82360856

(7, 2) 106.47551845

(7, 3) 101.34745901

(7, 4) 102.45651798

(7, 5) 98.74767493

(7, 6) 97.57544275

(7, 7) 92.5748759

(8, 0) 105.30350449

(8, 1) 92.87730812

(8, 2) 103.19258339

(8, 3) 104.40518318

(8, 4) 101.29326772

(8, 5) 100.85447132

(8, 6) 101.2226037

(8, 7) 106.03868807

(9, 0) 110.44484313

(9, 1) 93.87155456

(9, 2) 101.5363647

(9, 3) 97.65393524

(9, 4) 92.75048583

(9, 5) 101.72074646

(9, 6) 96.96851209

(9, 7) 103.29147111

(10, 0) 101.3514185

(10, 1) 100.37372248

(10, 2) 106.6471081

(10, 3) 100.61742813

(10, 4) 105.0320535

(10, 5) 99.35999981

(10, 6) 98.87007532

(10, 7) 95.85284217

(11, 0) 97.21315663

(11, 1) 107.02874163

(11, 2) 102.17642112

(11, 3) 96.74630281

(11, 4) 95.93799169

(11, 5) 102.62384733

(11, 6) 105.07475277

(11, 7) 97.59572169

(12, 0) 95.65982034

(12, 1) 107.22482426

(12, 2) 107.19119932

(12, 3) 102.93039474

(12, 4) 85.98839623

(12, 5) 95.19184343

(12, 6) 91.32093303

(12, 7) 102.35313953

(13, 0) 100.39303522

(13, 1) 92.0108226

(13, 2) 97.75887636

(13, 3) 93.18884302

(13, 4) 100.44940274

(13, 5) 108.09423367

(13, 6) 96.50342927

(13, 7) 99.58664719

(14, 0) 103.1521596

(14, 1) 109.40523174

(14, 2) 93.83969256

(14, 3) 99.95827854

(14, 4) 101.83462816

(14, 5) 99.69982772

(14, 6) 103.05289628

(14, 7) 103.93383957

(15, 0) 106.11454989

(15, 1) 88.80221141

(15, 2) 94.5081787

(15, 3) 94.59300658

(15, 4) 101.08830521

(15, 5) 96.34622848

(15, 6) 96.89244283

(15, 7) 98.07122664

(16, 0) 96.78266211

(16, 1) 99.84251605

(16, 2) 104.03478031

(16, 3) 106.57052697

(16, 4) 105.13668343

(16, 5) 105.37011896

(16, 6) 99.07551254

(16, 7) 104.15899829

(17, 0) 101.86186193

(17, 1) 103.61720152

(17, 2) 99.57859892

(17, 3) 99.4889538

(17, 4) 103.05541444

(17, 5) 98.65912661

(17, 6) 98.72774132

(17, 7) 104.70526438

(18, 0) 97.49594839

(18, 1) 96.59385486

(18, 2) 104.63817694

(18, 3) 102.55198606

(18, 4) 105.86078488

(18, 5) 96.5937781

(18, 6) 93.04610867

(18, 7) 99.92159953

(19, 0) 96.76814836

(19, 1) 91.6779221

(19, 2) 101.79132774

(19, 3) 101.20773355

(19, 4) 98.29243952

(19, 5) 101.83845792

(19, 6) 97.94046856

(19, 7) 102.20618501

(20, 0) 106.89005002

(20, 1) 106.57364584

(20, 2) 102.26648279

(20, 3) 107.40064604

(20, 4) 99.94318168

(20, 5) 103.40412146

(20, 6) 106.38276709

(20, 7) 98.00253006

(21, 0) 99.80873105

(21, 1) 101.63973121

(21, 2) 106.46476468

(21, 3) 110.43976681

(21, 4) 100.69156231

(21, 5) 99.99579473

(21, 6) 101.32113654

(21, 7) 94.76253572

(22, 0) 96.10020311

(22, 1) 94.57421727

(22, 2) 100.80409326

(22, 3) 105.02389857

(22, 4) 98.61325194

(22, 5) 95.62359311

(22, 6) 97.99762409

(22, 7) 103.83852459

(23, 0) 94.11176915

(23, 1) 99.62387832

(23, 2) 104.51786419

(23, 3) 97.62787811

(23, 4) 93.97853495

(23, 5) 98.75108352

(23, 6) 106.05042487

(23, 7) 100.07721494

[92.02628776 92.9267508 92.65657752 90.93318132 91.37294597 91.6604946

93.37126331 94.20019732 93.24376389 94.35373179 92.5748759 92.87730812

93.87155456 92.75048583 91.32093303 92.0108226 93.18884302 93.83969256

94.5081787 94.59300658 93.04610867 91.6779221 94.76253572 94.57421727

94.11176915 93.97853495]

[[ 91.37294597 88.80221141 90.93318132 93.18884302 85.98839623

91.6604946 91.32093303 92.5748759 ]

[ 92.02628776 91.6779221 93.24376389 94.59300658 89.03452725

92.65657752 93.04610867 94.20019732]

[ 94.11176915 92.0108226 93.83969256 96.74630281 92.75048583

95.19184343 94.35373179 94.76253572]

[ 95.65982034 92.87730812 94.5081787 97.24130034 92.9267508

95.46493436 96.50342927 95.08715803]

[ 95.66253664 93.87155456 97.75887636 97.24584816 93.37126331

95.62359311 96.81964892 95.85284217]

[ 96.02548256 94.57421727 98.45723272 97.62787811 93.97853495

96.2832753 96.89244283 97.59572169]

[ 96.10020311 95.17750125 99.32066924 97.65393524 95.93799169

96.34622848 96.96851209 98.00253006]

[ 96.76814836 96.59385486 99.57859892 97.85230675 98.29243952

96.5937781 97.57544275 98.07122664]

[ 96.78266211 97.10439252 100.28690912 99.4889538 98.61325194

98.65912661 97.94046856 98.56783189]

[ 97.21315663 98.29687616 100.40118279 99.95827854 98.74986914

98.74767493 97.99762409 99.58664719]

[ 97.49594839 99.62387832 100.80409326 100.61742813 98.80084371

98.75108352 98.37481387 99.92159953]

[ 99.14931546 99.84251605 101.5363647 100.72418901 99.94318168

98.80833412 98.72774132 100.07721494]

[ 99.80873105 100.37372248 101.79132774 101.20773355 100.44940274

99.35999981 98.87007532 101.13442416]

[100.39303522 100.96781394 102.17642112 101.34745901 100.69156231

99.69982772 99.07551254 101.23162942]

[101.20862522 101.63973121 102.26648279 102.55198606 101.08830521

99.99579473 100.79478953 102.20618501]

[101.3514185 102.82360856 103.19258339 102.93039474 101.29326772

100.85447132 101.2226037 102.35313953]

[101.86186193 102.97585605 104.03478031 104.40518318 101.83462816

101.72074646 101.32113654 103.29147111]

[102.80387079 103.5730309 104.51786419 105.02389857 102.45651798

101.83845792 103.05289628 103.83852459]

[103.1521596 103.61720152 104.63817694 105.85269352 103.05541444

102.62384733 104.60344836 103.93383957]

[105.30350449 104.03852715 106.46476468 106.57052697 105.0320535

103.40412146 105.07475277 104.15899829]

[106.11454989 106.57364584 106.47551845 107.40064604 105.13668343

105.37011896 105.7197853 104.70526438]

[106.71751618 107.02874163 106.6471081 110.18889465 105.48508838

105.95297652 106.05042487 106.03868807]

[106.89005002 107.22482426 107.19119932 110.43976681 105.86078488

108.09423367 106.1472841 106.54654286]

[110.44484313 109.40523174 107.43534677 113.42090475 106.39798503

108.57980357 106.38276709 106.83273763]]

[[ 99.14931546 104.03852715 107.43534677 97.85230675 98.74986914

98.80833412 96.81964892 98.56783189 92.02628776 97.10439252

99.32066924 97.24584816 92.9267508 92.65657752 105.7197853

101.23162942 95.66253664 95.17750125 90.93318132 110.18889465

98.80084371 105.95297652 98.37481387 106.54654286 91.37294597

100.96781394 100.40118279 113.42090475 105.48508838 91.6604946

106.1472841 95.08715803 101.20862522 103.5730309 100.28690912

105.85269352 93.37126331 108.57980357 100.79478953 94.20019732

102.80387079 98.29687616 93.24376389 97.24130034 89.03452725

96.2832753 104.60344836 101.13442416 106.71751618 102.97585605

98.45723272 100.72418901 106.39798503 95.46493436 94.35373179

106.83273763 96.02548256 102.82360856 106.47551845 101.34745901

102.45651798 98.74767493 97.57544275 92.5748759 105.30350449

92.87730812 103.19258339 104.40518318 101.29326772 100.85447132

101.2226037 106.03868807 110.44484313 93.87155456 101.5363647

97.65393524 92.75048583 101.72074646 96.96851209 103.29147111

101.3514185 100.37372248 106.6471081 100.61742813 105.0320535

99.35999981 98.87007532 95.85284217 97.21315663 107.02874163

102.17642112 96.74630281 95.93799169 102.62384733 105.07475277

97.59572169 95.65982034 107.22482426 107.19119932 102.93039474

85.98839623 95.19184343 91.32093303 102.35313953 100.39303522

92.0108226 97.75887636 93.18884302 100.44940274 108.09423367

96.50342927 99.58664719 103.1521596 109.40523174 93.83969256

99.95827854 101.83462816 99.69982772 103.05289628 103.93383957

106.11454989 88.80221141 94.5081787 94.59300658 101.08830521

96.34622848 96.89244283 98.07122664 96.78266211 99.84251605

104.03478031 106.57052697 105.13668343 105.37011896 99.07551254

104.15899829 101.86186193 103.61720152 99.57859892 99.4889538

103.05541444 98.65912661 98.72774132 104.70526438 97.49594839

96.59385486 104.63817694 102.55198606 105.86078488 96.5937781

93.04610867 99.92159953 96.76814836 91.6779221 101.79132774

101.20773355 98.29243952 101.83845792 97.94046856 102.20618501

106.89005002 106.57364584 102.26648279 107.40064604 99.94318168

103.40412146 106.38276709 98.00253006 99.80873105 101.63973121

106.46476468 110.43976681 100.69156231 99.99579473 101.32113654

94.76253572 96.10020311 94.57421727 100.80409326 105.02389857

98.61325194 95.62359311 97.99762409 103.83852459 94.11176915

99.62387832 104.51786419 97.62787811 93.97853495 98.75108352

106.05042487 100.07721494]]

[[ 99.14931546 104.03852715]

[107.43534677 97.85230675]

[ 98.74986914 98.80833412]

[ 96.81964892 98.56783189]

[ 92.02628776 97.10439252]

[ 99.32066924 97.24584816]

[ 92.9267508 92.65657752]

[105.7197853 101.23162942]

[ 95.66253664 95.17750125]

[ 90.93318132 110.18889465]

[ 98.80084371 105.95297652]

[ 98.37481387 106.54654286]

[ 91.37294597 100.96781394]

[100.40118279 113.42090475]

[105.48508838 91.6604946 ]

[106.1472841 95.08715803]

[101.20862522 103.5730309 ]

[100.28690912 105.85269352]

[ 93.37126331 108.57980357]

[100.79478953 94.20019732]

[102.80387079 98.29687616]

[ 93.24376389 97.24130034]

[ 89.03452725 96.2832753 ]

[104.60344836 101.13442416]

[106.71751618 102.97585605]

[ 98.45723272 100.72418901]

[106.39798503 95.46493436]

[ 94.35373179 106.83273763]

[ 96.02548256 102.82360856]

[106.47551845 101.34745901]

[102.45651798 98.74767493]

[ 97.57544275 92.5748759 ]

[105.30350449 92.87730812]

[103.19258339 104.40518318]

[101.29326772 100.85447132]

[101.2226037 106.03868807]

[110.44484313 93.87155456]

[101.5363647 97.65393524]

[ 92.75048583 101.72074646]

[ 96.96851209 103.29147111]

[101.3514185 100.37372248]

[106.6471081 100.61742813]

[105.0320535 99.35999981]

[ 98.87007532 95.85284217]

[ 97.21315663 107.02874163]

[102.17642112 96.74630281]

[ 95.93799169 102.62384733]

[105.07475277 97.59572169]

[ 95.65982034 107.22482426]

[107.19119932 102.93039474]

[ 85.98839623 95.19184343]

[ 91.32093303 102.35313953]

[100.39303522 92.0108226 ]

[ 97.75887636 93.18884302]

[100.44940274 108.09423367]

[ 96.50342927 99.58664719]

[103.1521596 109.40523174]

[ 93.83969256 99.95827854]

[101.83462816 99.69982772]

[103.05289628 103.93383957]

[106.11454989 88.80221141]

[ 94.5081787 94.59300658]

[101.08830521 96.34622848]

[ 96.89244283 98.07122664]

[ 96.78266211 99.84251605]

[104.03478031 106.57052697]

[105.13668343 105.37011896]

[ 99.07551254 104.15899829]

[101.86186193 103.61720152]

[ 99.57859892 99.4889538 ]

[103.05541444 98.65912661]

[ 98.72774132 104.70526438]

[ 97.49594839 96.59385486]

[104.63817694 102.55198606]

[105.86078488 96.5937781 ]

[ 93.04610867 99.92159953]

[ 96.76814836 91.6779221 ]

[101.79132774 101.20773355]

[ 98.29243952 101.83845792]

[ 97.94046856 102.20618501]

[106.89005002 106.57364584]

[102.26648279 107.40064604]

[ 99.94318168 103.40412146]

[106.38276709 98.00253006]

[ 99.80873105 101.63973121]

[106.46476468 110.43976681]

[100.69156231 99.99579473]

[101.32113654 94.76253572]

[ 96.10020311 94.57421727]

[100.80409326 105.02389857]

[ 98.61325194 95.62359311]

[ 97.99762409 103.83852459]

[ 94.11176915 99.62387832]

[104.51786419 97.62787811]

[ 93.97853495 98.75108352]

[106.05042487 100.07721494]]

**9. Write a program to parse HTML using Beutifulsoup/equivalent Library support.**

import urllib.request    
from bs4 import BeautifulSoup  
  
url = "<https://www.geeksforgeeks.org/how-to-automate-an-excel-sheet-in-python/?ref=feed>"  
   
html = urllib.request.urlopen(url)  
   
  
htmlParse = BeautifulSoup(html, 'html.parser')  
   
  
p=htmlParse.find("p").get\_text()  
print(p)  
  
  
import matplotlib.pyplot as plt  
def Cvc(text):  
    vowels='AEIOUaeiou'  
    vowelcount=0  
    consonentcount=0  
    for char in text:  
        if char in vowels:  
            vowelcount+=1  
        else:  
            consonentcount+=1  
    return vowelcount,consonentcount  
  
vowelcount,consonentcount=Cvc(p)  
categories=['Vowels','Consonents']  
values=[vowelcount,consonentcount]  
print(values)  
plt.pie(values,labels=categories)  
plt.show()