**Practical No.01**

**Aim : Installation and study of any one Data Analytics Tool Frame work.**

**Theory :**

Programming languages are used to solve a variety of data problems. now we will focus on general ones that use letters, numbers, and symbols to create programs and require formal syntax used by programmers. Often, they’re also called text-based programs because you need to write software that will ultimately solve a problem. Examples include C#, Java, PHP, Ruby, Julia, and Python, among many others on the market. Here we will present Python as one of the best tools for data analysts that have coding knowledge as well.

**PYTHON**

**KEY FEATURES:** An open-source solution that has simple coding processes and syntax so it’s fairly easy to learn Integration with other languages such as C/C++, Java, PHP, C#, etc. Advanced analysis processes through machine learning and text mining [Python](https://www.python.org/) is extremely accessible to code in comparison to other popular languages such as Java, and its syntax is relatively easy to learn making this tool popular among users that look for an open-source solution and simple coding processes. In data analysis, Python is used for data crawling, cleaning, modeling, and constructing analysis algorithms based on business scenarios. One of the best features is actually its user-friendliness: programmers don’t need to remember the architecture of the system nor handle the memory – Python is considered a high-level language that is not subject to the computer’s local processor. Another noticeable feature of Python is its portability. Users can simply run the code on several operating systems without making any changes to it so it’s not necessary to write completely new code. This makes Python a highly portable language since programmers can run it both on Windows and mac OS. An extensive number of modules, packages and libraries make Python a respected and usable language across industries with companies such as Spotify, Netflix, Dropbox and Reddit as the most popular ones that use this language in their operations. With features such as text mining and machine learning, Python is becoming a respected authority for advanced analysis processes.

**Practical No.02**

**Aim : Design and develop at least 10 problem statements which demonstrate the use of data structure, functions, Importing / Exporting Data in any data analytics tool.**

**Code :**

import numpy as np

import pandas as pd

list1 = [1,2,3,4]

array1 = np.array(list1)

print(array1)

print("\nAnother dataSet")

ages = np.array([13,25,19])

series1 = pd.Series(ages,index=['Emma', 'Swetha', 'Serajh'])

print(series1)

**O/P**

**[1 2 3 4]**

Another dataSet

Emma 13

Swetha 25

Serajh 19

dtype: int64

**Practical No.03**

**Aim : Design and develop at least 5 problem statements which demonstrate the use of Control Structures of any data analytics tool.**

**Code :**

def digitSum(n):

dsum = 0

for ele in str(n):

dsum += int(ele)

return dsum

List = [367, 111, 562, 945, 6726, 873]

newList = [digitSum(i) for i in List if i & 1]

print(newList)

a = [1, 2, 3, 4]

while a:

print(a.pop())

for i in range(10):

print(i)

if i == 2:

break

**Output :**

[16, 3, 18, 18]

4

3

2

1

0

1

2

**Practical No.04**

**Aim : Implement any 2 Classification techniques using any data analytics tool.**

**Decision Tree Classification**

**Code :**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.svm import SVC

print(fruits.head())

print(fruits.tail())

print(fruits.describe())

feature\_names = ['mass', 'width', 'height', 'color\_score']

X = fruits[feature\_names]

y = fruits['fruit\_label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

scaler = MinMaxScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

c = DecisionTreeClassifier().fit(X\_train, y\_train)

print('Accuracy of Decision Tree classifier on training set: {:.2f}'

.format(c.score(X\_train, y\_train)))

print('Accuracy of Decision Tree classifier on test set: {:.2f}'

.format(c.score(X\_test, y\_test)))

**Output :**

fruit\_label fruit\_name fruit\_subtype mass width height color\_score

0 1 apple granny\_smith 192 8.4 7.3 0.55

1 1 apple granny\_smith 180 8.0 6.8 0.59

2 1 apple granny\_smith 176 7.4 7.2 0.60

3 2 mandarin mandarin 86 6.2 4.7 0.80

4 2 mandarin mandarin 84 6.0 4.6 0.79

fruit\_label fruit\_name fruit\_subtype mass width height color\_score

54 4 lemon unknown 116 6.1 8.5 0.71

55 4 lemon unknown 116 6.3 7.7 0.72

56 4 lemon unknown 116 5.9 8.1 0.73

57 4 lemon unknown 152 6.5 8.5 0.72

58 4 lemon unknown 118 6.1 8.1 0.70

fruit\_label mass width height color\_score

count 59.000000 59.000000 59.000000 59.000000 59.000000

mean 2.542373 163.118644 7.105085 7.693220 0.762881

std 1.208048 55.018832 0.816938 1.361017 0.076857

min 1.000000 76.000000 5.800000 4.000000 0.550000

25% 1.000000 140.000000 6.600000 7.200000 0.720000

50% 3.000000 158.000000 7.200000 7.600000 0.750000

75% 4.000000 177.000000 7.500000 8.200000 0.810000

max 4.000000 362.000000 9.600000 10.500000 0.930000

Accuracy of Decision Tree classifier on training set: 1.00

Accuracy of Decision Tree classifier on test set: 0.73

**KNN**

**Code:**

knn = KNeighborsClassifier()

knn.fit(X\_train, y\_train)

print('Accuracy of K-NN classifier on training set: {:.2f}'

.format(knn.score(X\_train, y\_train)))

print('Accuracy of K-NN classifier on test set: {:.2f}'

.format(knn.score(X\_test, y\_test)))

**Output :**

Accuracy of K-NN classifier on training set: 0.95

Accuracy of K-NN classifier on test set: 1.00

**SVM :**

**Code :**

svm = SVC()

svm.fit(X\_train, y\_train)

print('Accuracy of SVM classifier on training set: {:.2f}'

.format(svm.score(X\_train, y\_train)))

print('Accuracy of SVM classifier on test set: {:.2f}'

.format(svm.score(X\_test, y\_test)))

**Output :**

Accuracy of SVM classifier on training set: 0.9

1Accuracy of SVM classifier on test set: 0.80

**Practical No.5**

**Aim : Implement any 2 Clustering techniques using any data analytics tool.**

**Implementation of the k-means clustering algorithm**

**Code :**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_digits

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

data, labels = load\_digits(return\_X\_y=True)

(n\_samples, n\_features), n\_digits = data.shape, np.unique(labels).size

reduced\_data = PCA(n\_components=2).fit\_transform(data)

kmeans = KMeans(init="k-means++", n\_clusters=n\_digits, n\_init=4)

kmeans.fit(reduced\_data)

# Step size of the mesh. Decrease to increase the quality of the VQ.

h = 0.02 # point in the mesh [x\_min, x\_max]x[y\_min, y\_max].

# Plot the decision boundary. For that, we will assign a color to each

x\_min, x\_max = reduced\_data[:, 0].min() - 1, reduced\_data[:, 0].max() + 1

y\_min, y\_max = reduced\_data[:, 1].min() - 1, reduced\_data[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

# Obtain labels for each point in mesh. Use last trained model.

Z = kmeans.predict(np.c\_[xx.ravel(), yy.ravel()])

# Put the result into a color plot

Z = Z.reshape(xx.shape)

plt.figure(1)

plt.clf()

plt.imshow(

Z,

interpolation="nearest",

extent=(xx.min(), xx.max(), yy.min(), yy.max()),

cmap=plt.cm.Paired,

aspect="auto",

origin="lower",

)

plt.plot(reduced\_data[:, 0], reduced\_data[:, 1], "k.", markersize=2)

# Plot the centroids as a white X

centroids = kmeans.cluster\_centers\_

plt.scatter(

centroids[:, 0],

centroids[:, 1],

marker="x",

s=169,

linewidths=3,

color="w",

zorder=10,

)

plt.title(

"K-means clustering on the digits dataset (PCA-reduced data)\n"

"Centroids are marked with white cross"

)

plt.xlim(x\_min, x\_max)

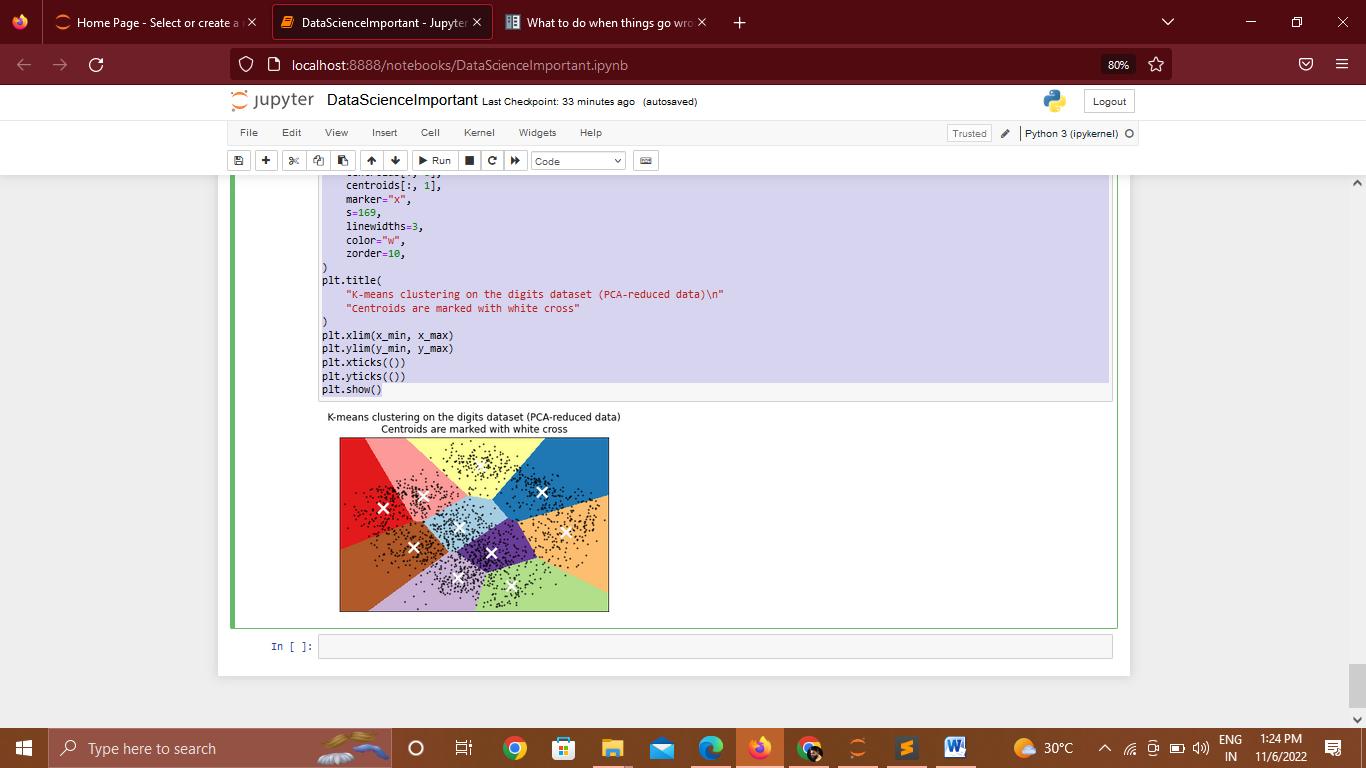
plt.ylim(y\_min, y\_max)

plt.xticks(())

plt.yticks(())

plt.show()

**Output :**



**Implementation of the DBSCAN(Density Based) clustering algorithm**

**Code:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

from sklearn import metrics

from sklearn.datasets import make\_blobs

from sklearn.preprocessing import StandardScaler

centers = [[1, 1], [-1, -1], [1, -1]]

X, labels\_true = make\_blobs(

n\_samples=750, centers=centers, cluster\_std=0.4, random\_state=0

)

X = StandardScaler().fit\_transform(X)

db = DBSCAN(eps=0.3, min\_samples=10).fit(X)

core\_samples\_mask = np.zeros\_like(db.labels\_, dtype=bool)

core\_samples\_mask[db.core\_sample\_indices\_] = True

labels = db.labels\_

# Number of clusters in labels, ignoring noise if present.

n\_clusters\_ = len(set(labels)) - (1 if -1 in labels else 0)

n\_noise\_ = list(labels).count(-1)

print("Estimated number of clusters: %d" % n\_clusters\_)

print("Estimated number of noise points: %d" % n\_noise\_)

print("Homogeneity: %0.3f" % metrics.homogeneity\_score(labels\_true, labels))

print("Completeness: %0.3f" % metrics.completeness\_score(labels\_true, labels))

print("V-measure: %0.3f" % metrics.v\_measure\_score(labels\_true, labels))

print("Adjusted Rand Index: %0.3f" % metrics.adjusted\_rand\_score(labels\_true, labels))

print(

"Adjusted Mutual Information: %0.3f"

% metrics.adjusted\_mutual\_info\_score(labels\_true, labels)

)

print("Silhouette Coefficient: %0.3f" % metrics.silhouette\_score(X, labels))

# Black removed and is used for noise instead.

unique\_labels = set(labels)

colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique\_labels))]

for k, col in zip(unique\_labels, colors):

if k == -1:

# Black used for noise.

col = [0, 0, 0, 1]

class\_member\_mask = labels == k

xy = X[class\_member\_mask & core\_samples\_mask]

plt.plot(

xy[:, 0],

xy[:, 1],

"o",

markerfacecolor=tuple(col),

markeredgecolor="k",

markersize=14,

)

xy = X[class\_member\_mask & ~core\_samples\_mask]

plt.plot(

xy[:, 0],

xy[:, 1],

"o",

markerfacecolor=tuple(col),

markeredgecolor="k",

markersize=6,

)

plt.title("Estimated number of clusters: %d" % n\_clusters\_)

plt.show()

**Output:**

Estimated number of clusters: 3

Estimated number of noise points: 18

Homogeneity: 0.953

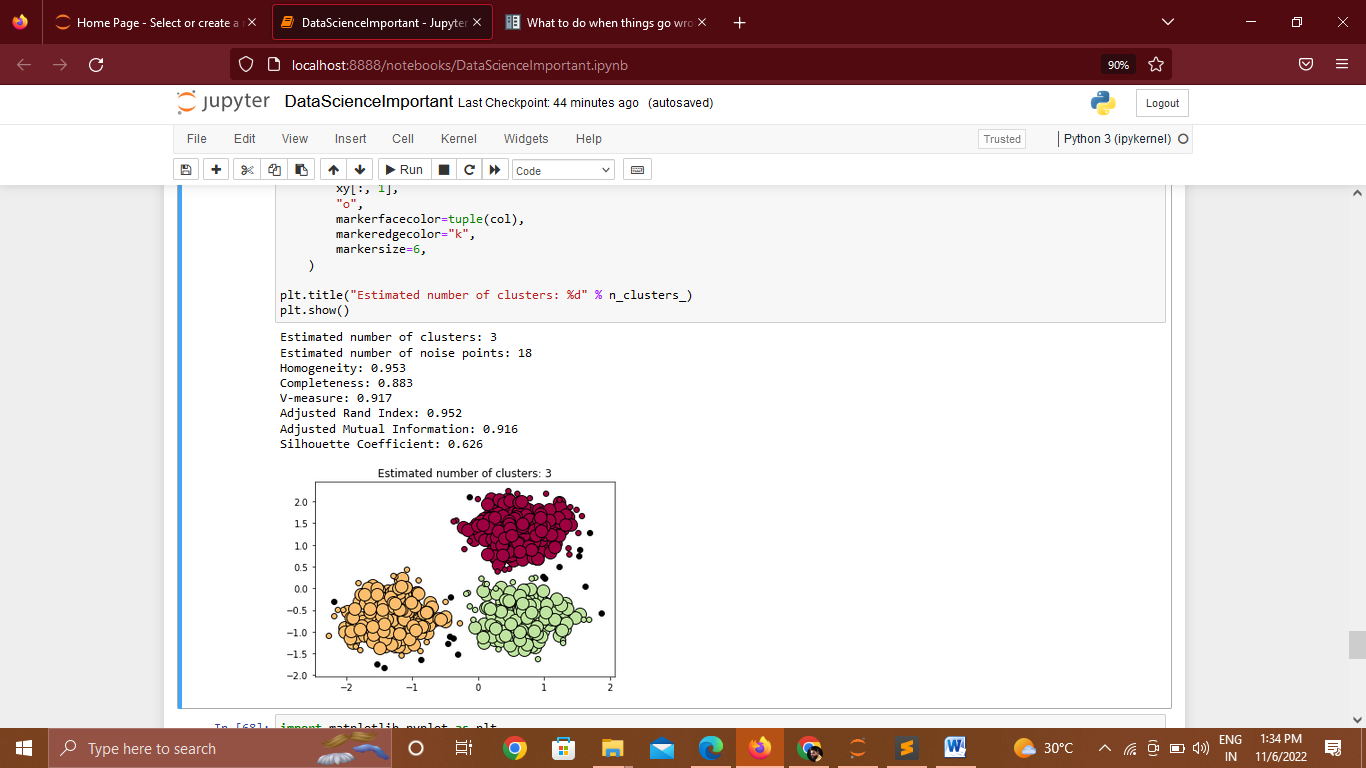
Completeness: 0.883

V-measure: 0.917

Adjusted Rand Index: 0.952

Adjusted Mutual Information: 0.916

Silhouette Coefficient: 0.626



**Practical No.06**

**Aim : Implement any 2 Association Rule Mining techniques using any data analytics tool.** **Code :**

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

data.columns

data.Country.unique()

data['Description'] = data['Description'].str.strip()

# Dropping the rows without any invoice number

data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

data['InvoiceNo'] = data['InvoiceNo'].astype('str')

# Dropping all transactions which were done on credit

data = data[~data['InvoiceNo'].str.contains('C')]

basket\_France = (data[data['Country'] =="France"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

# Transactions done in the United Kingdom

basket\_UK = (data[data['Country'] =="United Kingdom"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

# Transactions done in Portugal

basket\_Por = (data[data['Country'] =="Portugal"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

basket\_Sweden = (data[data['Country'] =="Sweden"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

frq\_items = apriori(basket\_France, min\_support = 0.05, use\_colnames = True)

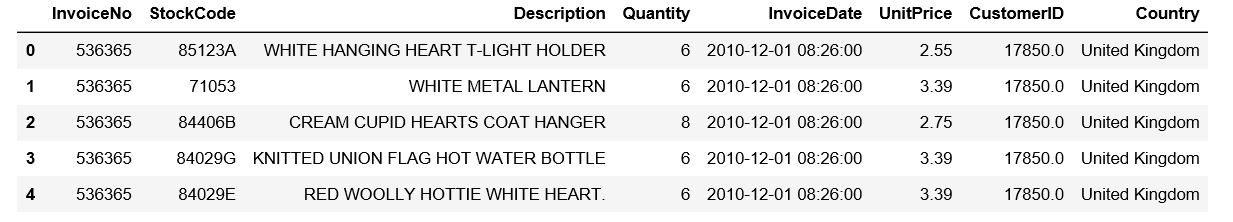
# Collecting the inferred rules in a dataframe

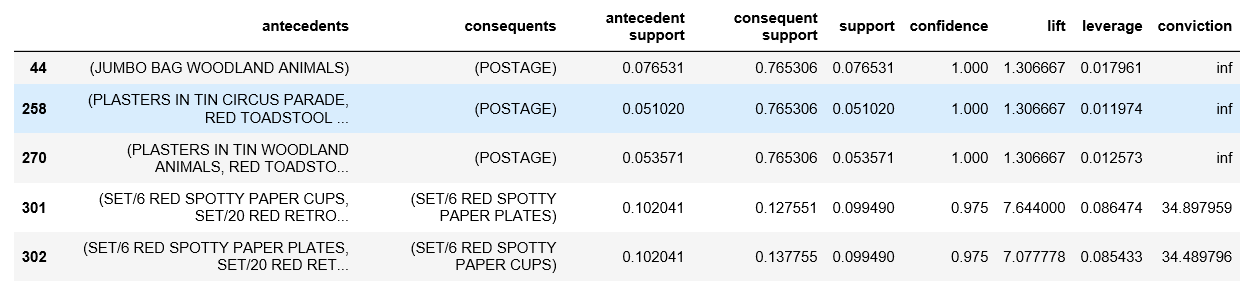
rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

print(rules.head())

**Output :**





**Practical No. 07**

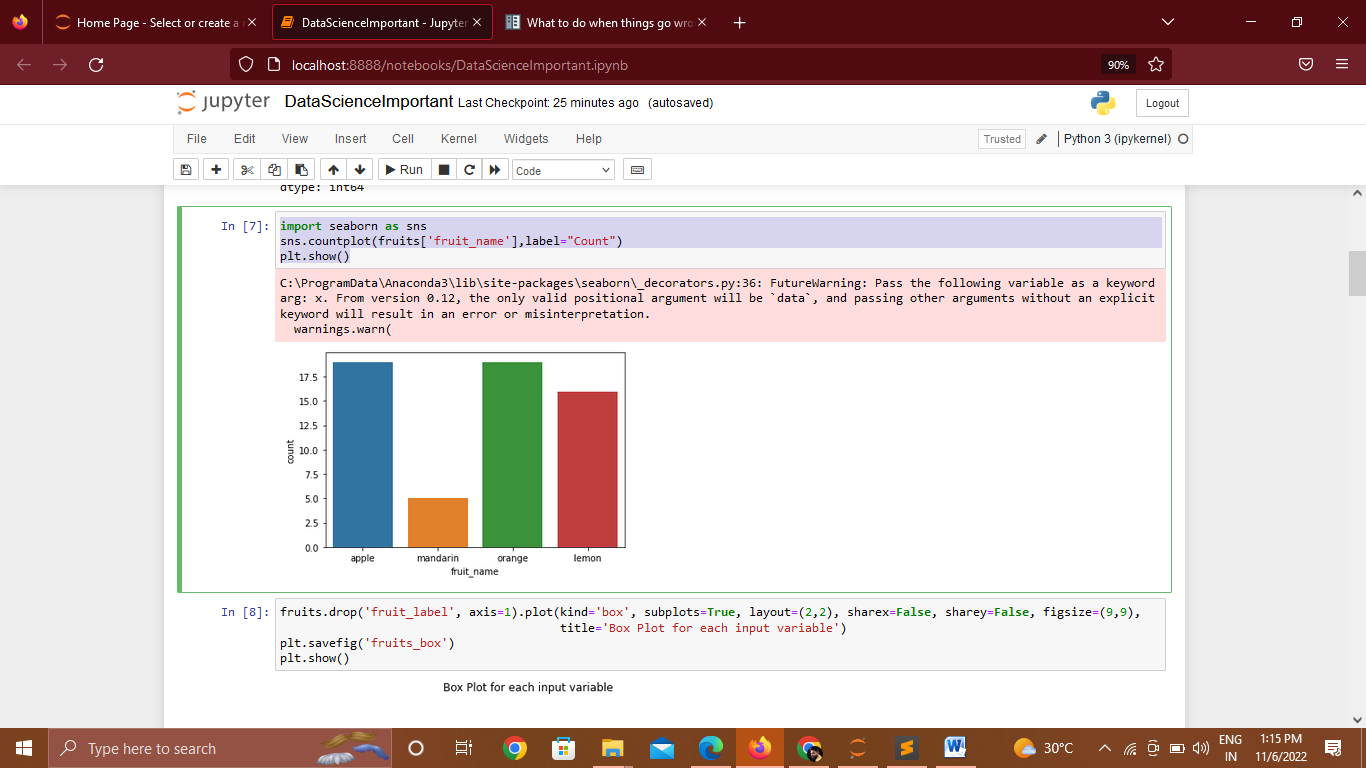
**Aim : Visualize all the statistical measures (mean, mode, median, range, inter quartile range, etc.) using Histograms, Boxplots, scatter plots, etc.**

**Code :**

import seaborn as sns

sns.countplot(fruits['fruit\_name'],label="Count")

plt.show()

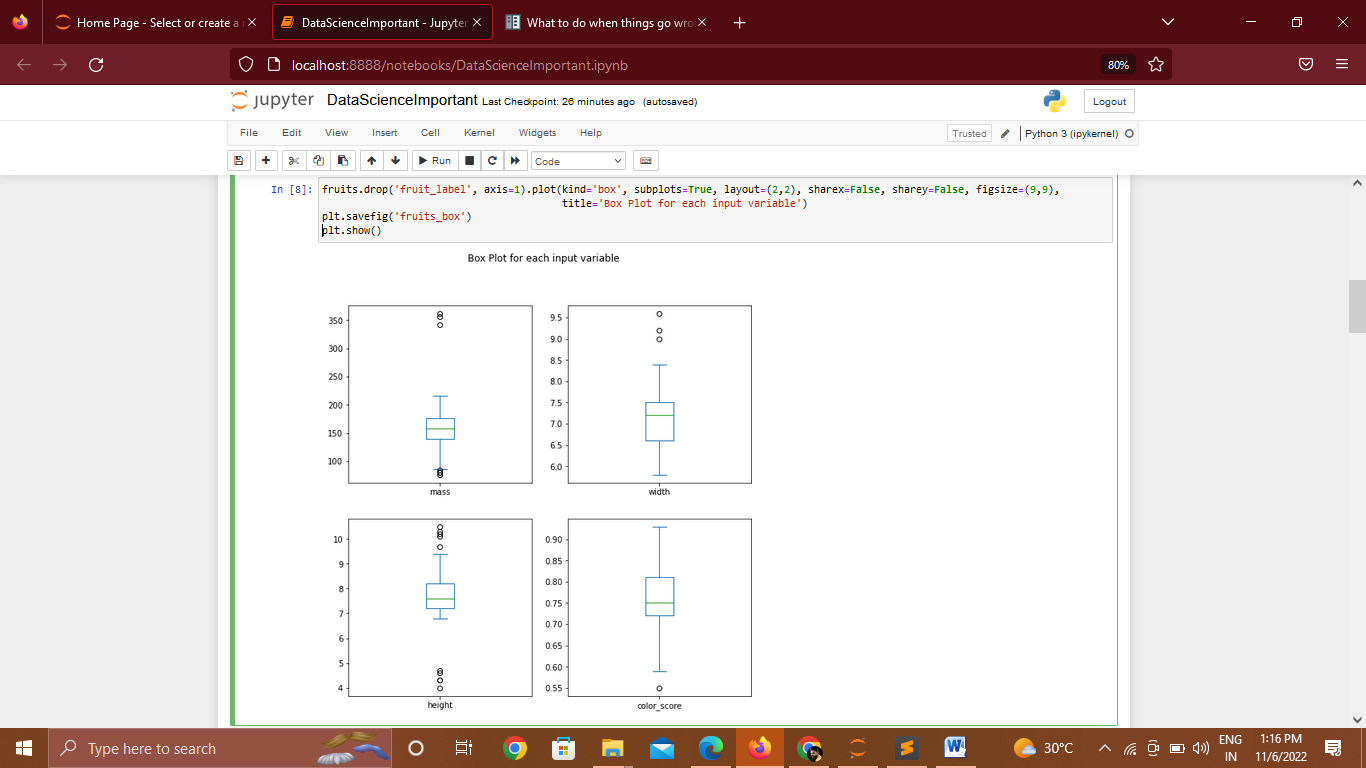


fruits.drop('fruit\_label', axis=1).plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, figsize=(9,9),

title='Box Plot for each input variable')

plt.savefig('fruits\_box')

plt.show()



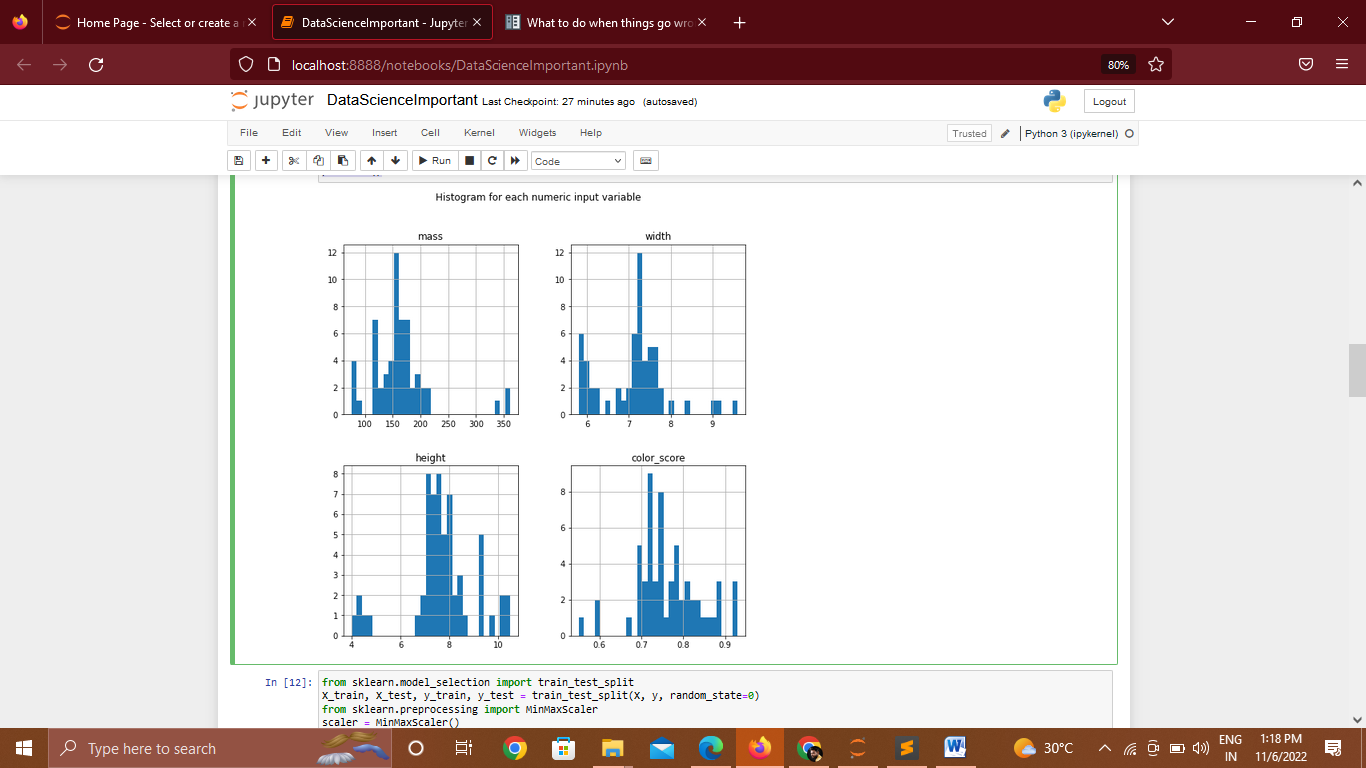
import pylab as pl

fruits.drop('fruit\_label' ,axis=1).hist(bins=30, figsize=(9,9))

pl.suptitle("Histogram for each numeric input variable")

plt.savefig('fruits\_hist')

plt.show()



k\_range = range(1, 20)

scores = []

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors = k)

knn.fit(X\_train, y\_train)

scores.append(knn.score(X\_test, y\_test))

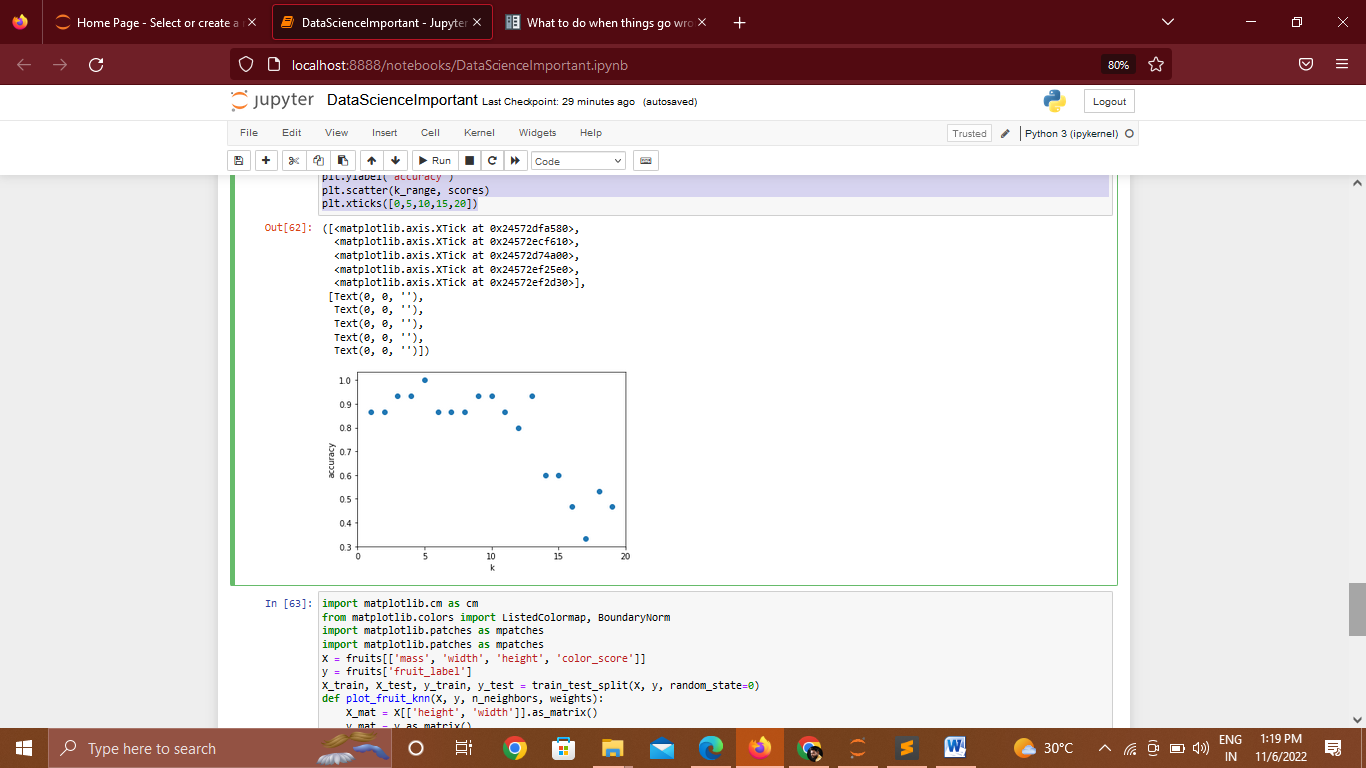
plt.figure()

plt.xlabel('k')

plt.ylabel('accuracy')

plt.scatter(k\_range, scores)

plt.xticks([0,5,10,15,20])



**Practical No.08**

**Aim : Design and Develop real-time Data Science Application (e.g. Image Recognition/ Intelligent Assistant/ Recommendation System/ Fake News Detection/Emotion Recognition/Chatbot/Other)**

**Code : Chatbat**

import time

import random

name = input("Hello, what is your name? ")

time.sleep(2)

print("Hello " + name)

feeling = input("How are you today? ")

time.sleep(2)

if "good" in feeling:

print("I'm feeling good too!")

else:

print("I'm sorry to hear that!")

time.sleep(2)

favcolour = input("What is your favourite colour? ")

colours = ["Red","Green","Blue"]

time.sleep(2)

print("My favourite colour is " + random.choice(colours))

**OutPut:**

**Hello, what is your name? Mr.Pramod**

**Hello Mr.Pramod**

**How are you today? I'm feeling good too!**

**I'm feeling good too!**

**What is your favourite colour? Blue**

**My favourite colour is Blue**