**SCIKIT-LEARN:**

1. **IMPORTING LIBRARIES, DATASET & READING DATASET:**

import sklearn

# load the iris dataset as an example

from sklearn.datasets import load\_iris

iris = load\_iris()

# store the feature matrix (X) and response vector (y)

X = iris.data

y = iris.target

# store the feature and target names

feature\_names = iris.feature\_names

target\_names = iris.target\_names

# printing features and target names of our dataset

print("Feature names:", feature\_names)

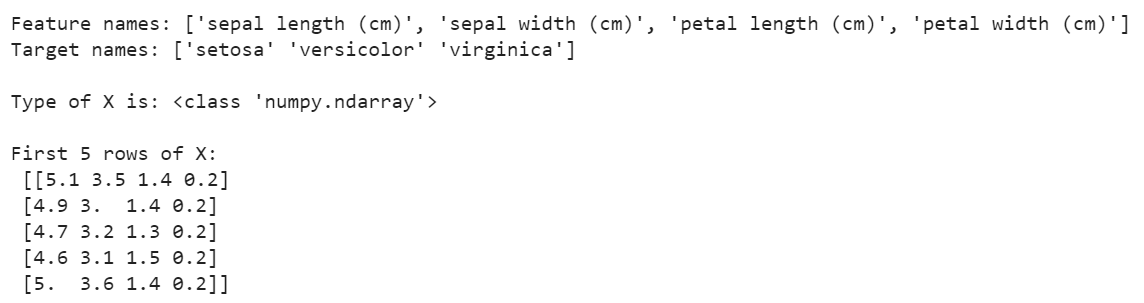
print("Target names:", target\_names)

# X and y are numpy arrays

print("\nType of X is:", type(X))

# printing first 5 input rows

print("\nFirst 5 rows of X:\n", X[:5])



1. **SPLITTING DATASET INTO TRAINING & TESTING DATASET:**

# splitting X and y into training and testing sets

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=1)

# printing the shapes of the new X objects

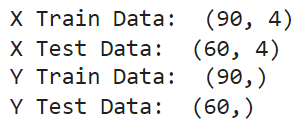
print('X Train Data: ',X\_train.shape)

print('X Test Data: ',X\_test.shape)

# printing the shapes of the new y objects

print('Y Train Data: ',y\_train.shape)

print('Y Test Data: ',y\_test.shape)



1. **NORMALIZATION:**

# import module

from sklearn.preprocessing import StandardScaler

# create data

data = [[11, 2], [3, 7], [0, 10], [11, 8]]

# compute required values

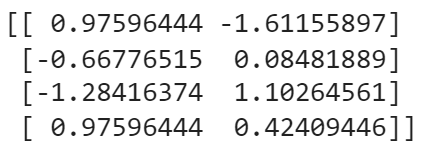
scaler = StandardScaler()

model = scaler.fit(data)

scaled\_data = model.transform(data)

# print scaled data

print(scaled\_data)



**SCIPY:**

1. **SPARSE MATRIX:**

# import necessary modules

from scipy import sparse

# Row-based linked list sparse matrix

A = sparse.lil\_matrix((1000, 1000))

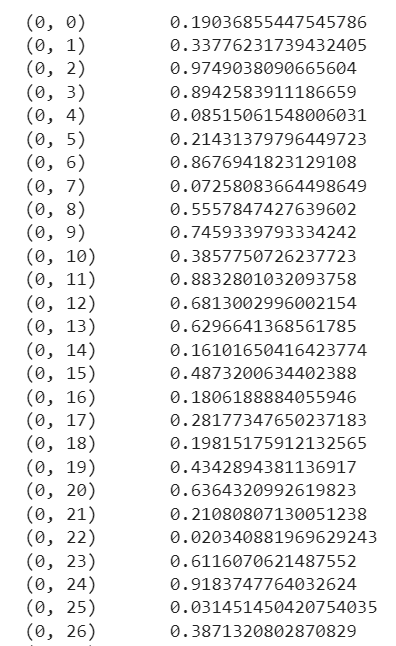
print(A)

A[0,:100] = np.random.rand(100)

A[1,100:200] = A[0,:100]

A.setdiag(np.random.rand(1000))

print(A)

****

1. **CONVERT THIS MATRIX TO COMPRESSED SPARSE ROW FORMAT**

from scipy.sparse import linalg

# Convert this matrix to Compressed Sparse Row format.

A.tocsr()

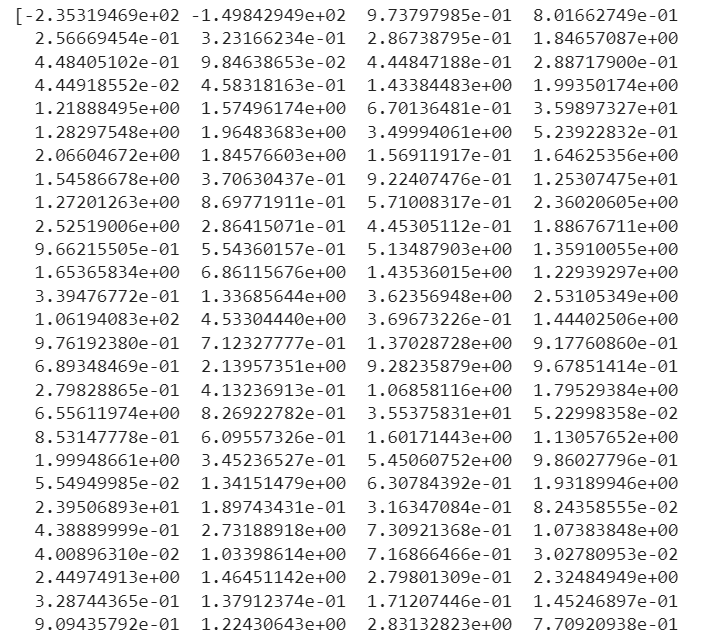
A = A.tocsr()

b = np.random.rand(1000)

ans = linalg.spsolve(A, b)

# it will print ans array of 1000 size

print(ans)

****

1. **DETEMINANT:**

# import numpy library

import numpy as np

A = np.array([[1,2,3],[4,5,6],[7,8,8]])

# importing linalg function from scipy

from scipy import linalg

# Compute the determinant of a matrix

linalg.det(A)



1. **DOT PRODUCT:**

P, L, U = linalg.lu(A)

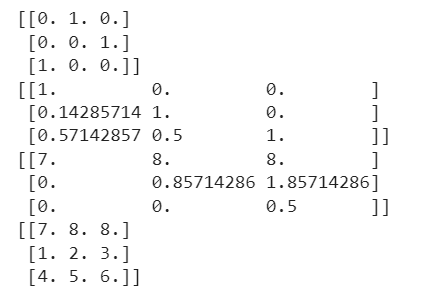
print(P)

print(L)

print(U)

# print LU decomposition

print(np.dot(L,U))

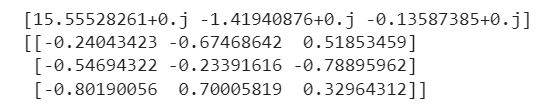
****

1. **EIGEN VECTOR:**

eigen\_values, eigen\_vectors = linalg.eig(A)

print(eigen\_values)

print(eigen\_vectors)

****

1. **INTEGRATION:**

from scipy import integrate

f = lambda y, x: x\*y\*\*2

i = integrate.dblquad(f, 0, 2, lambda x: 0, lambda x: 1)

# print the results

print(i)

****

**CONCLUSION:**

From this practical, I have learned and implemented scikit-learn & scipy libraries in python.