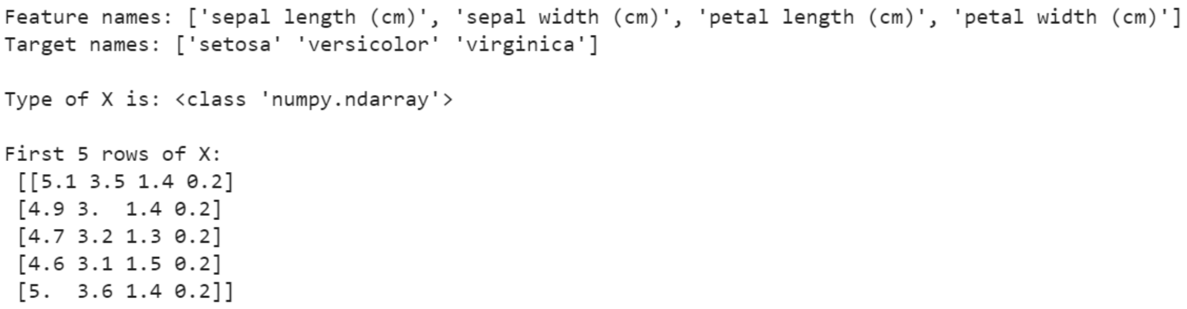
SCIKIT-LEARN:

A) IMPORTING LIBRARIES, DATASET & READING DATASET:

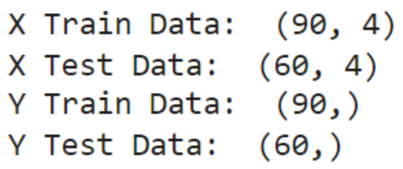
```
import sklearn
# load the iris dataset as an example
from sklearn.datasets import load_iris
iris = load_iris()

# 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])
```



```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']

Type of X is: <class 'numpy.ndarray'>

First 5 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

B) SPLITTING DATASET INTO TRAINING & TESTING DATASET:

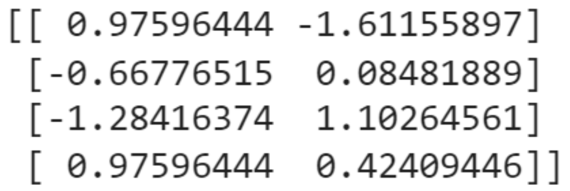
```
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# 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)
```



```
X Train Data:  (90, 4)
X Test Data:  (60, 4)
Y Train Data:  (90,)
Y Test Data:  (60,)
```

C) 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)
```

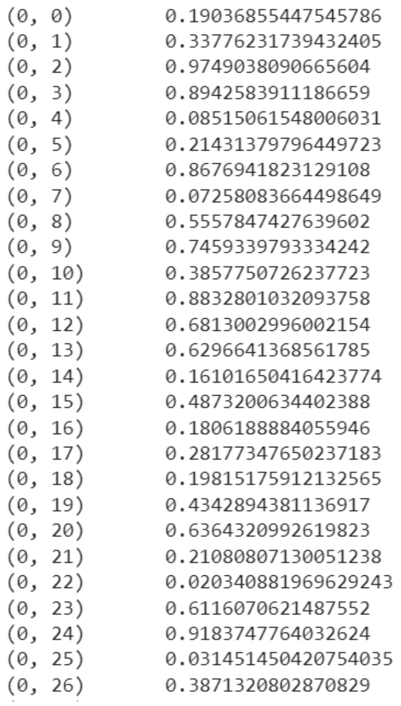


```
[[ 0.97596444 -1.61155897]
 [-0.66776515  0.08481889]
 [-1.28416374  1.10264561]
 [ 0.97596444  0.42409446]]
```

SCIPY:

A) SPARSE MATRIX:

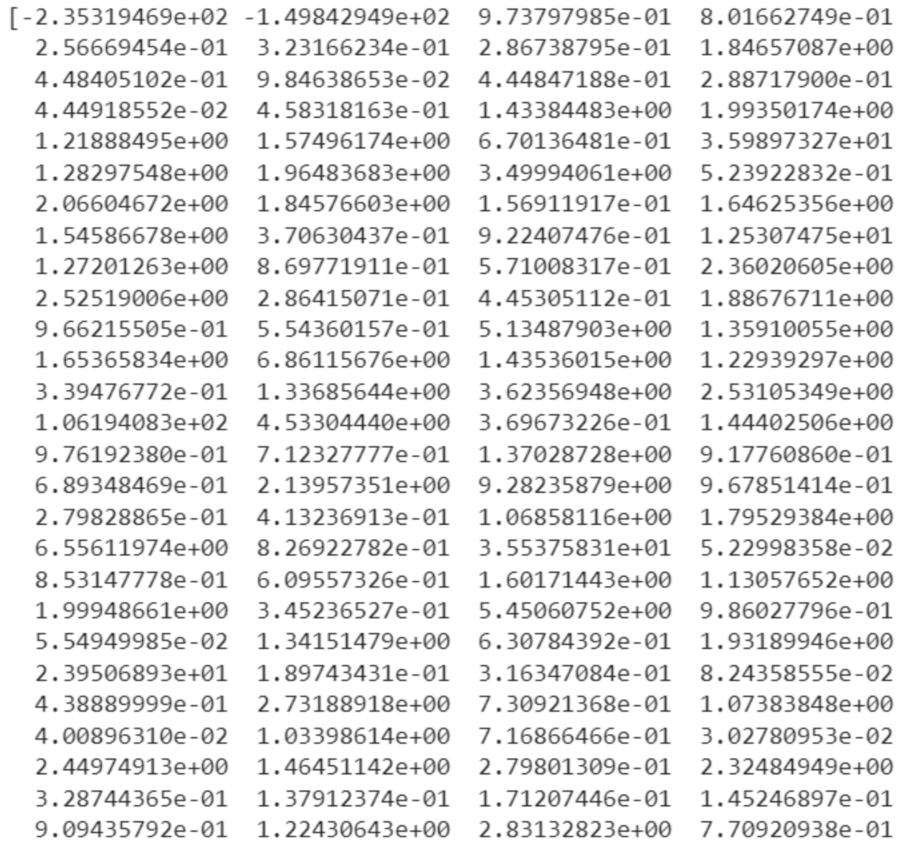
```
# import necessary modules
from scipy import sparse
# Row-based linked list sparse matrix
A = sparse.lil_matrix((1000, 1000))
print(A)

A[0,:100] = np.random.rand(100)
A[1,100:200] = A[0,:100]
A.setdiag(np.random.rand(1000))
print(A)
```



```
  (0, 0)	0.19036855447545786
  (0, 1)	0.33776231739432405
  (0, 2)	0.9749038090665604
  (0, 3)	0.8942583911186659
  (0, 4)	0.08515061548006031
  (0, 5)	0.21431379796449723
  (0, 6)	0.8676941823129108
  (0, 7)	0.07258083664498649
  (0, 8)	0.5557847427639602
  (0, 9)	0.7459339793334242
  (0, 10)	0.3857750726237723
  (0, 11)	0.8832801032093758
  (0, 12)	0.6813002996002154
  (0, 13)	0.6296641368561785
  (0, 14)	0.16101650416423774
  (0, 15)	0.4873200634402388
  (0, 16)	0.1806188884055946
  (0, 17)	0.28177347650237183
  (0, 18)	0.19815175912132565
  (0, 19)	0.4342894381136917
  (0, 20)	0.6364320992619823
  (0, 21)	0.21080807130051238
  (0, 22)	0.020340881969629243
  (0, 23)	0.6116070621487552
  (0, 24)	0.9183747764032624
  (0, 25)	0.031451450420754035
  (0, 26)	0.3871320802870829
```

B) <u>CONVERT THIS MATRIX TO COMPRESSED SPARSE ROW FORMAT</u>

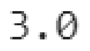
```
from scipy.sparse import linalg

# 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)
```



```
[-2.35319469e+02 -1.49842949e+02  9.73797985e-01  8.01662749e-01
  2.56669454e-01  3.23166234e-01  2.86738795e-01  1.84657087e+00
  4.48405102e-01  9.84638653e-02  4.44847188e-01  2.88717900e-01
  4.44918552e-02  4.58318163e-01  1.43384483e+00  1.99350174e+00
  1.21888495e+00  1.57496174e+00  6.70136481e-01  3.59897327e+01
  1.28297548e+00  1.96483683e+00  3.49994061e+00  5.23922832e-01
  2.06604672e+00  1.84576603e+00  1.56911917e-01  1.64625356e+00
  1.54586678e+00  3.70630437e-01  9.22407476e-01  1.25307475e+01
  1.27201263e+00  8.69771911e-01  5.71008317e-01  2.36020605e+00
  2.52519006e+00  2.86415071e-01  4.45305112e-01  1.88676711e+00
  9.66215505e-01  5.54360157e-01  5.13487903e+00  1.35910055e+00
  1.65365834e+00  6.86115676e+00  1.43536015e+00  1.22939297e+00
  3.39476772e-01  1.33685644e+00  3.62356948e+00  2.53105349e+00
  1.06194083e+02  4.53304440e+00  3.69673226e-01  1.44402506e+00
  9.76192380e-01  7.12327777e-01  1.37028728e+00  9.17760860e-01
  6.89348469e-01  2.13957351e+00  9.28235879e+00  9.67851414e-01
  2.79828865e-01  4.13236913e-01  1.06858116e+00  1.79529384e+00
  6.55611974e+00  8.26922782e-01  3.55375831e+01  5.22998358e-02
  8.53147778e-01  6.09557326e-01  1.60171443e+00  1.13057652e+00
  1.99948661e+00  3.45236527e-01  5.45060752e+00  9.86027796e-01
  5.54949985e-02  1.34151479e+00  6.30784392e-01  1.93189946e+00
  2.39506893e+01  1.89743431e-01  3.16347084e-01  8.24358555e-02
  4.38889999e-01  2.73188918e+00  7.30921368e-01  1.07383848e+00
  4.00896310e-02  1.03398614e+00  7.16866466e-01  3.02780953e-02
  2.44974913e+00  1.46451142e+00  2.79801309e-01  2.32484949e+00
  3.28744365e-01  1.37912374e-01  1.71207446e-01  1.45246897e-01
  9.09435792e-01  1.22430643e+00  2.83132823e+00  7.70920938e-01
```

C) DETEMINANT:

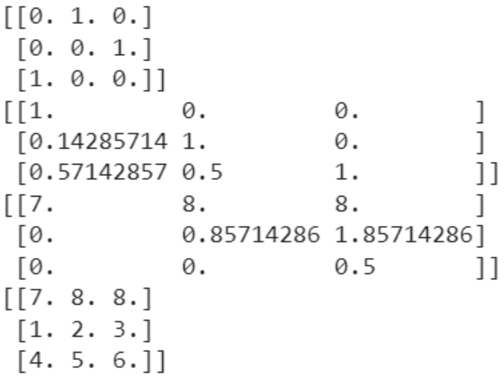
```
# import numpy library
import numpy as np
A = np.array([[1,2,3],[4,5,6],[7,8,8]])
# importing linalg function from scipy
from scipy import linalg

# Compute the determinant of a matrix
linalg.det(A)
```



```
3.0
```

D) DOT PRODUCT:

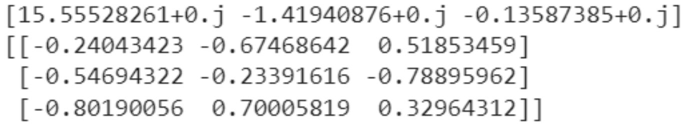
```
P, L, U = linalg.lu(A)
print(P)
print(L)
print(U)
# print LU decomposition
print(np.dot(L,U))
```



```
[[0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]]
[[1.         0.         0.        ]
 [0.14285714 1.         0.        ]
 [0.57142857 0.5        1.        ]]
[[7.         8.         8.        ]
 [0.         0.85714286 1.85714286]
 [0.         0.         0.5       ]]
[[7. 8. 8.]
 [1. 2. 3.]
 [4. 5. 6.]]
```

E) EIGEN VECTOR:

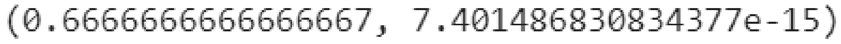
```
eigen_values, eigen_vectors = linalg.eig(A)
print(eigen_values)
print(eigen_vectors)
```



```
[15.55528261+0.j -1.41940876+0.j -0.13587385+0.j]
[[-0.24043423 -0.67468642  0.51853459]
 [-0.54694322 -0.23391616 -0.78895962]
 [-0.80190056  0.70005819  0.32964312]]
```

F) 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)
```



```
(0.6666666666666667, 7.401486830834377e-15)
```

CONCLUSION:

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