SCIPY

- SciPy is a python library that is useful in solving many mathematical equations and algorithms.
- It is designed on the top of Numpy library that gives more extension of finding scientific mathematical formulae like Matrix Rank, Inverse, polynomial equations, LU Decomposition, etc.
- Using its high level functions will significantly reduce the complexity of the code and helps in better analyzing the data. SciPy is an interactive Python session used as a data-processing library that is made to compete with its rivalries such as MATLAB, Octave, R-Lab,etc.
- It has many user-friendly, efficient and easy-to-use functions that helps to solve problems like numerical integration, interpolation, optimization, linear algebra and statistics.
- The benefit of using SciPy library in Python while making ML models is that it also makes a strong programming language available for use in developing less complex programs and applications.

Linear Algebra

Determinant of a Matrix

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

Compute pivoted LU decomposition of a matrix

- LU decomposition is a method that reduce matrix into constituent parts that helps in easier calculation of complex matrix operations.
- The decomposition methods are also called matrix factorization methods, are base of linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.
  The decomposition is:
  A = P L U
  where P is a permutation matrix, L lower triangular with unit diagonal elements, and U upper triangular.

```
from scipy import linalg
P, L, U = linalg.lu(A)
print(P)
print(L)
print(U)
print(np.dot(L,U))
```

Eigen values and eigen vectors of this matrix

```
import numpy as np
A = np.array([[1,2,3],[4,5,6],[7,8,8
eigen_values, eigen_vectors = linalg.eig(A)
print(eigen_values)
print(eigen_vectors)
```

Solving systems of linear equations can also be done

```
v = np.array([[2],[3],[5]])
print(v)
s = linalg.solve(A,v)
print(s)
```

Sparse Linear Algebra

SciPy has some routines for computing with sparse and potentially very large matrices. The necessary tools are in the submodule scipy.sparse.

```
from scipy import sparse
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)
```

Linear Algebra for Sparse Matrices

```
from scipy.sparse import linalg
A.tocsr()
A = A.tocsr()
b = np.random.rand(1000)
ans = linalg.spsolve(A, b)
print(ans)
```

Integration

- When a function is very difficult to integrate analytically, on ıd a solution through numerical integration methods. SciPy has a capabil g numerical integration also. Scipy has integration methods in scipy.inte ıle.

$$\int_a^b f(x)dx$$

Single Integrals

- The Quad routine is the important function out of SciPy's integration functions. If integration in over f(x) function where x ranges from a to b, then integral looks like this.
- The parameters of quad is scipy.integrate.quad(f, a, b), Where 'f' is the function to be integrated. Whereas, 'a' and 'b' are the lower and upper ranges of x limit. Let us see an example of integrating e^{-x^2} over the range of 0 and 1 with respect to dx.

We will first define the function f(x)=e^(-x^2) , this is done using a lambda expression and then use quad routine.

```
import scipy.integrate
f= lambda x:np.exp(-x**2)
i = scipy.integrate.quad(f, 0, 1)
print(i)
```

Double Integrals

- The parameters of dblquad function is scipy.integr $\int_0^1 \int_0^2 x * y^2 \, dx \, dy$, a, b, g, h).
- Where, ‘f’ is the function to be integrated, ‘a’ and ‘b r and upper ranges of the x variable, respectively, while ‘g’ and ‘h’ are the functions that tells the lower and upper limits of y variable.
- As an example, let us perform the double integral of x*y^2 over x range from 0 to 2 and y ranges from 0 to 1.
- We define the functions f, g, and h, using the lambda expressions. Note that even if g and h are constants, as they may be in many cases, they must be defined as functions, as we have done here for the lower limit.

```
from scipy import integrate
f = lambda y, x: x*y**2
i = integrate.dblquad(f, 0, 2, lambda x: 0, lambda x: 1)
print(i)
```