Name: Manav Shah
Roll No: 231070902
Second Year CS
Subject: **Programming Lab1**

Experiment No. 10

AIM: To perform analysis of data using NumPy and Pandas libraries.

THEORY:

Numpy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

Features of NumPy:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Pandas: Pandas is a powerful and versatile library that simplifies tasks of data manipulation in Python . Pandas is built on top of the NumPy library and is particularly well-suited for working with tabular data, such as spreadsheets or SQL tables. Its versatility and ease of use make it an essential tool for data analysts, scientists, and engineers working with structured data in Python.

It is built on the top of the NumPy library which means that a lot of structures of NumPy are used or replicated in Pandas. The data produced by Pandas are often used as input for plotting functions of Matplotlib, statistical analysis in SciPy, and machine learning algorithms in Scikit-learn. Here is a list of things that we can do using Pandas.

CODE:

To perform analysis of data using NumPy and Pandas libraries.

```
import pandas as pd
cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", names=cols)
df.head()
```

Output:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
df.to_csv("iris.csv", index=False)
df.head()                    # shows top 5 rows
```

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
df.tail()                    # shows bottom 5 rows
```

	sepal_length	sepal_width	petal_length	petal_width	class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
df.describe()                    # provides info about the df
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
df.dtypes                        # shows the datatypes of columns
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
class            object
dtype: object
```

```
new_df = df.copy()               # creates a copy
new_df.to_numpy()
```

```
array([[5.1, 3.5, 1.4, 0.2, 'Iris-setosa'],
       [4.9, 3.0, 1.4, 0.2, 'Iris-setosa'],
       [4.7, 3.2, 1.3, 0.2, 'Iris-setosa'],
       [4.6, 3.1, 1.5, 0.2, 'Iris-setosa'],
       [5.0, 3.6, 1.4, 0.2, 'Iris-setosa'],
       [5.4, 3.9, 1.7, 0.4, 'Iris-setosa'],
       [4.6, 3.4, 1.4, 0.3, 'Iris-setosa'],
       [5.0, 3.4, 1.5, 0.2, 'Iris-setosa'],
       [4.4, 2.9, 1.4, 0.2, 'Iris-setosa'],
       [4.9, 3.1, 1.5, 0.1, 'Iris-setosa'],
       [5.4, 3.7, 1.5, 0.2, 'Iris-setosa'],
       [4.8, 3.4, 1.6, 0.2, 'Iris-setosa'],
       [4.8, 3.0, 1.4, 0.1, 'Iris-setosa'],
       [4.3, 3.0, 1.1, 0.1, 'Iris-setosa'],
       [5.8, 4.0, 1.2, 0.2, 'Iris-setosa'],
       [5.7, 4.4, 1.5, 0.4, 'Iris-setosa'],
       [5.4, 3.9, 1.3, 0.4, 'Iris-setosa'],
       [5.1, 3.5, 1.4, 0.3, 'Iris-setosa'],
       [5.7, 3.8, 1.7, 0.3, 'Iris-setosa'],
```

```
new_df.T                                # Transposes the df
```

	1	2	3	4	5	6	7	8	9	10	...	141	142	143	144	145	146	147
sepal_length	5.1	4.9	4.7	4.6	5.0	5.4	4.6	5.0	4.4	4.9	...	6.7	6.9	5.8	6.8	6.7	6.7	6.3
sepal_width	3.5	3.0	3.2	3.1	3.6	3.9	3.4	3.4	2.9	3.1	...	3.1	3.1	2.7	3.2	3.3	3.0	2.5
petal_length	1.4	1.4	1.3	1.5	1.4	1.7	1.4	1.5	1.4	1.5	...	5.6	5.1	5.1	5.9	5.7	5.2	5.0
petal_width	0.2	0.2	0.2	0.2	0.2	0.4	0.3	0.2	0.2	0.1	...	2.4	2.3	1.9	2.3	2.5	2.3	1.9
class	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	...	Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica

5 rows × 150 columns

```
new_df.sort_index(axis=0, ascending=False)          # sorts the df
```

	sepal_length	sepal_width	petal_length	petal_width	class
150	5.9	3.0	5.1	1.8	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
148	6.5	3.0	5.2	2.0	Iris-virginica
147	6.3	2.5	5.0	1.9	Iris-virginica
146	6.7	3.0	5.2	2.3	Iris-virginica
...	...	...	...	...	...
5	5.0	3.6	1.4	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa

```
new_df.loc[[1, 2], :]                    # shows specified cells
```

	1	2	3	4	5
1	123.0	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa

```
new_df.loc[(new_df[1] < 5) & (new_df[2] > 2)]      # shows specified
cells with condition
```

	1	2	3	4	5
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa

```
new_df.drop([5], axis=1, inplace=True)        # used to drop rows and
columns
```

```
new_df
```

	1	2	3	4
1	123.0	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
6	5.4	3.9	1.7	0.4
...	...	...	...	...

```
new_df.drop_duplicates()                    # drops the duplicate records
```

	1	2	3	4
1	123.0	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
6	5.4	3.9	1.7	0.4
...	...	...	...	...

```
new_df.info()                    # shows info about df
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 149 entries, 1 to 150
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   1       149 non-null    float64
 1   2       149 non-null    float64
 2   3       149 non-null    float64
 3   4       149 non-null    float64
dtypes: float64(4)
memory usage: 9.9+ KB
```

```
df.min()                # shows min value from each columns
```

```
sepal_length            4.3
sepal_width             2.0
petal_length            1.0
petal_width             0.1
class           Iris-setosa
dtype: object
```

```
df.max()                # shows maxvalue from each columns
```

```
sepal_length               7.9
sepal_width                4.4
petal_length               6.9
petal_width                2.5
class           Iris-virginica
dtype: object
```

```
df.median()             # shows median value from each columns
```

```
sepal_length    5.80
sepal_width     3.00
petal_length    4.35
petal_width     1.30
dtype: float64
```

```
df.std()                # shows standard deviation value from each columns
```

```
sepal_length    0.828066
sepal_width     0.433594
petal_length    1.764420
petal_width     0.763161
dtype: float64
```

```
df.corr()               # shows correlation between each column with every
other column
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

CONCLUSION : From this experiment, we learnt about the Numpy and Pandas library in python which are used for data processing and manipulation. Pandas is built on top of Numpy and is well-suited for working with tabular data, such as spreadsheets or SQL tables. In this experiment, we imported the iris dataset and performed various operations on the dataframe like sorting, slicing, transposing, etc. We also cleaned the dataframe by removing the duplicate values.