Machine Learning Lab (AI332L)

Class:	BS Artificial Intelligence	**Semester:**	5th (Fall 2023)
Instructor:	Dr. M. Ehatisham-ul-Haq	**Email:**	ehtisham@mail.au.edu.pk

| Lab 01 |

INTRODUCTION TO PYTHON WITH BASIC MACHINE LIBRARIES

Lab Objectives:

- To understand the basics of Python and fundamental libraries
- To understand multidimensional array and matrix processing using the Numpy library
- To understand scientific computations and linear algebra functions using the Scipy library
- To understand data preprocessing and analysis using the Pandas library
- To understand data visualization in the form of 2D graphs and plots using Matplotlib library

Introduction to Python and Setup using Google Colab:

This section provides an overview of Python and guides participants through setting up Python with Google Colab.

Setting up Python with Google Colab

1. Go to Google Colab.
2. Create a new notebook by clicking on **File** > **New Notebook**.
3. You're now ready to start writing and executing Python code in the notebook!

Setting up Python with Jupyter:

- Download and install Python from python.org
- Install Jupyter Notebook with Anaconda: **conda install -c anaconda jupyter**

Python Fundamentals

1. Variables:

Variables in Python are used to store data values. They are like containers that can hold different types of data.

```
name = "John" # A string variable
age = 30 # An integer variable
height = 5.9 # A float variable
is_student = True # A boolean variable
```

2. Data Types:

a. Numeric Types:

```
integer_number = 42
float_number = 3.14
complex_number = 2 + 3j
```

b. String:

```
name = "John Doe"
message = 'Hello, World!'
```

c. Boolean:

```
is_adult = True
is_student = False
```

d. List:

```
numbers = [1, 2, 3, 4, 5]
names = ['John', 'Jane', 'Doe']
mixed_list = [1, 'John', True]
```

e. Tuple:

```
coordinates = (10, 20)
names = ('John', 'Jane', 'Doe')
```

f. Dictionary:

```
person = {'name': 'John', 'age': 30, 'is_student': False}
```

3. Control Flow:

a. If-Else Statements:

```
age = 20
if age >= 18:
  print("You are an adult.")
else:
  print("You are a minor.")
```

b. Loops (for and while):

```
for i in range(5):
  print(i)
count = 0
while count < 5:
  print(count)
  count += 1
```

4. Functions:

Functions are blocks of code that perform a specific task. They promote code reusability.

```
def greet(name):
  print(f"Hello, {name}!")
```

Putting it All Together:

```
name = "John Doe"
age = 25
is_student = True
if age >= 18:
  print(f"{name} is an adult.")
else:
  print(f"{name} is a minor.")
def say_hello():
  print(f"Hello, my name is {name} and I'm {age} years old.")
say_hello()
```

List of important Python Libraries

- Python Libraries for Data Collection
  - Beautiful Soup
  - Scrapy
  - Selenium
- Python Libraries for Data Cleaning and Manipulation
  - Pandas
  - PyOD
  - NumPy
  - Scipy
  - Spacy
- Python Libraries for Data Visualization
  - Matplotlib
  - Seaborn
  - Bokeh
- Python Libraries for Modeling
  - Scikit-learn
  - TensorFlow
  - PyTorch
- Python Libraries for Model Interpretability
  - Lime
  - H2O
- Python Libraries for Audio Processing

- Librosa
- Madmom
- pyAudioAnalysis

- Python Libraries for Image Processing
  - OpenCV-Python
  - Scikit-image
  - Pillow
- Python Libraries for Database
  - Psycopg
  - SQLAlchemy
- Python Libraries for Deployment
  - Flask

List of Python Math Libraries:

List of Functions in Python Math Module	
Function	**Description**
ceil(x)	Returns the smallest integer greater than or equal to x.
copysign(x,y)	Returns x with the sign of y
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
fmod(x, y)	Returns the remainder when x is divided by y
frexp(x)	Returns the mantissa and exponent of x as the pair (m, e)
fsum(iterable)	Returns an accurate floating point sum of values in the iterable
isfinite(x)	Returns True if x is neither an infinity nor a NaN (Not a Number)
isinf(x)	Returns True if x is a positive or negative infinity
isnan(x)	Returns True if x is a NaN
ldexp(x, i)	Returns x * (2**i)
modf(x)	Returns the fractional and integer parts of x
trunc(x)	Returns the truncated integer value of x
exp(x)	Returns e**x
expm1(x)	Returns e**x - 1
log(x[, base])	Returns the logarithm of x to the base (defaults to e)
log1p(x)	Returns the natural logarithm of 1+x
log2(x)	Returns the base-2 logarithm of x
log10(x)	Returns the base-10 logarithm of x
pow(x, y)	Returns x raised to the power y

sqrt(x)	Returns the square root of x
acos(x)	Returns the arc cosine of x
asin(x)	Returns the arc sine of x
atan(x)	Returns the arc tangent of x
atan2(y, x)	Returns atan(y / x)
cos(x)	Returns the cosine of x
hypot(x, y)	Returns the Euclidean norm, sqrt(x*x + y*y)
sin(x)	Returns the sine of x
tan(x)	Returns the tangent of x
degrees(x)	Converts angle x from radians to degrees
radians(x)	Converts angle x from degrees to radians
acosh(x)	Returns the inverse hyperbolic cosine of x
asinh(x)	Returns the inverse hyperbolic sine of x
atanh(x)	Returns the inverse hyperbolic tangent of x
cosh(x)	Returns the hyperbolic cosine of x
sinh(x)	Returns the hyperbolic cosine of x
tanh(x)	Returns the hyperbolic tangent of x
erf(x)	Returns the error function at x
erfc(x)	Returns the complementary error function at x
gamma(x)	Returns the Gamma function at x
lgamma(x)	Returns the natural logarithm of the absolute value of the Gamma function at x
Pi	Mathematical constant, the ratio of the circumference of a circle to its diameter (3.14159...)
E	Mathematical constant e (2.71828...)

Python NumPy Libraries:

NumPy is a powerful Python library for numerical computations. It supports working with arrays, matrices, and a wide range of mathematical functions to operate on these data structures. NumPy is a fundamental library for scientific computing in Python.

Key Features of NumPy:

- **Arrays:** NumPy provides the ndarray (n-dimensional array) object, which allows for efficient storage and manipulation of arrays of numbers.
- **Vectorized Operations:** NumPy enables vectorized operations, meaning that you can perform operations on entire arrays simultaneously, which is much more efficient than loops.
- **Broadcasting:** NumPy automatically handles operations between arrays of different shapes and dimensions through a process called broadcasting.
- **Linear Algebra:** NumPy provides a wide range of linear algebra functions, including matrix operations, eigenvalues, and more.

- **Random Number Generation:** It includes tools for generating random numbers and distributions.
- **FFT (Fast Fourier Transform):** NumPy provides functions for performing fast Fourier transforms.
- **Statistical Functions:** NumPy has a range of functions for performing statistical calculations.

Basic Examples: Creating Arrays

```
import numpy as np
# Creating a 1-dimensional array
arr_1d = np.array([1, 2, 3, 4, 5])
# Creating a 2-dimensional array
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
# Creating an array with specified range
arr_range = np.arange(0, 10, 2)  # Generates [0, 2, 4, 6, 8]
# Creating an array with specified shape filled with zeros
arr_zeros = np.zeros((2, 3))  # Generates a 2x3 matrix of zeros
# Creating an array with specified shape filled with ones
arr_ones = np.ones((3, 2))  # Generates a 3x2 matrix of ones
```

Basic Examples: Operations and Broadcasting

```
# Adding two arrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
result = arr1 + arr2  # [5, 7, 9]
# Scalar operations
arr = np.array([1, 2, 3])
result = arr * 2  # [2, 4, 6]
# Broadcasting
arr = np.array([[1, 2], [3, 4]])
result = arr * 2  # [[2, 4], [6, 8]]
```

Basic Examples: Mathematical Functions

```
# Sine function
arr = np.array([0, np.pi/2, np.pi])
result = np.sin(arr)  # [0. 1. 1.2246468e-16]
```

```
# Exponential function
arr = np.array([1, 2, 3])
result = np.exp(arr)  # [ 2.71828183  7.3890561  20.08553692]
# Sum, mean, and standard deviation
arr = np.array([1, 2, 3, 4, 5])
sum_arr = np.sum(arr)  # 15
mean_arr = np.mean(arr)  # 3.0
std_arr = np.std(arr)  # 1.4142135
```

Basic Examples: Linear Algebra

```
# Matrix multiplication
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
result = np.dot(A, B)  # [[19, 22], [43, 50]]
# Eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)
```

Here is a list of some useful NumPy functions and methods names ordered in categories.

Array Creation
arange, array, copy, empty, empty_like, eye, fromfile, fromfunction, identity, linspace, logspace, mgrid, ogrid, ones, ones_like, *r*, zeros, zeros_like

Conversions
ndarray.astype, atleast_1d, atleast_2d, atleast_3d, mat

Manipulations
array_split, column_stack, concatenate, diagonal, dsplit, dstack, hsplit, hstack, ndarray.item, newaxis, ravel, repeat, reshape, resize, squeeze, swapaxes, take, transpose, vsplit, vstack

Questions
all, any, nonzero, where

Ordering
argmax, argmin, argsort, max, min, ptp, searchsorted, sort

Operations
choose, compress, cumprod, cumsum, inner, ndarray.fill, imag, prod, put, putmask, real, sum

Basic Statistics
cov, mean, std, var

Basic Linear Algebra

cross, dot, outer, linalg.svd, vdot

Python SciPy Libraries:

SciPy is a Python library built on top of NumPy that provides a wide range of scientific computing and optimization functions. It extends the functionality of NumPy and includes additional modules for tasks such as numerical integration, optimization, signal processing, statistics, and more.

Key Features of SciPy:

- **Optimization and Root Finding:** SciPy includes functions for finding minima or maxima of functions and for solving systems of nonlinear equations.
- **Integration:** It provides methods for numerical integration, including both definite and indefinite integrals.
- **Interpolation:** SciPy offers functions for interpolating data, enabling you to estimate values between known data points.
- **Signal and Image Processing**: It includes modules for tasks like filtering, image manipulation, and spectral analysis.
- **Statistical Functions:** SciPy extends the statistical capabilities of NumPy with additional functions for hypothesis testing, probability distributions, and more.
- **Linear Algebra:** SciPy builds upon NumPy's linear algebra capabilities and adds more advanced functions for solving linear systems, eigenvalues, and matrix decompositions.
- **Special Functions:** It provides a wide range of specialized mathematical functions not available in NumPy.

Python Machine Learning Libraries:

Pandas:

Pandas is a powerful Python library used for data manipulation and analysis. It provides high-level data structures like Series (1-dimensional) and DataFrame (2-dimensional) that are optimized for performance and ease of use. Pandas is widely used in data science, finance, and other fields where data processing and analysis are crucial.

Key Features of Pandas:

- **DataFrame:** A 2-dimensional labeled data structure with columns of potentially different types. It's similar to a spreadsheet or SQL table.
- **Series:** A 1-dimensional labeled array capable of holding any data type.
- **Data Alignment:** Automatically aligns data based on label (row and column names).
- **Missing Data Handling:** Provides functions for detecting and dealing with missing values.

- **Group By:** Allows for the grouping of data for aggregation and transformation operations.
- **Merge and Join:** Provides tools for combining datasets based on keys.
- **Time Series Functionality:** Supports time-series data.
- **Reshaping and Pivoting:** Allows for reshaping and pivoting of data.

Basic Examples:

```
import pandas as pd

# Create a DataFrame from a dictionary

data = {'Name': ['John', 'Jane', 'Jim', 'Jill'],

        'Age': [25, 30, 35, 40]}

df = pd.DataFrame(data)

print(df)


# Create a Series from a list

s = pd.Series([1, 3, 5, np.nan, 6, 8])

print(s)


# Load data from a CSV file

df = pd.read_csv('data.csv')

df.head()

df.shape()

temp_df=df.append(df)

temp_df.shape

temp_df = temp_df.drop_duplicates()

temp_df.shape


col1 = df['column1']

type(col1)


# Accessing columns

names = df['Name']
```

```
# Filtering data
young_people = df[df['Age'] < 30]
# Adding a new column
df['Gender'] = ['M', 'F', 'M', 'F']

# Load data from an Excel file
df = pd.read_excel('data.xlsx')

# Accessing columns
names = df['Name']
# Filtering data
young_people = df[df['Age'] < 30]

# Adding a new column
df['Gender'] = ['M', 'F', 'M', 'F']
# Fill missing values with a specific value
df.fillna(0, inplace=True)
# Drop rows with missing values
df.dropna(inplace=True)
```

Matplotlib:

Matplotlib is a comprehensive Python library for creating static, animated, and interactive visualizations in Python. It is widely used for generating plots, charts, histograms, and other types of visualizations to help in data exploration and presentation.

```
import matplotlib.pyplot as plt
```

List of functions available:

- plot(x-axis values, y-axis values) — plots a simple line graph with x-axis values against y-axis values
- show() — displays the graph

- title(—string‖) — set the title of the plot as specified by the string
- xlabel(—string‖) — set the label for x-axis as specified by the string
- ylabel(—string‖) — set the label for y-axis as specified by the string
- figure() — used to control a figure level attributes
- subplot(nrows, ncols, index) — Add a subplot to the current figure
- suptitle(—string‖) — It adds a common title to the figure specified by the string
- subplots(nrows, ncols, figsize) — a convenient way to create subplots, in a single call. It returns a tuple of a figure and number of axes
- set_title(—string‖) — an axes level method used to set the title of subplots in a figure
- bar(categorical variables, values, color) — used to create vertical bar graphs
- barh(categorical variables, values, color) — used to create horizontal bar graphs
- legend(loc) — used to make legend of the graph
- xticks(index, categorical variables) — Get or set the current tick locations and labels
- of the x-axis
- pie(value, categorical variables) — used to create a pie chart
- hist(values, number of bins) — used to create a histogram
- xlim(start value, end value) — used to set the limit of values of the x-axis
- ylim(start value, end value) — used to set the limit of values of the y-axis
- scatter(x-axis values, y-axis values) — plots a scatter plot with x-axis values against y-axis values
- axes() — adds an axes to the current figure
- set_xlabel(—string‖) — axes level method used to set the x-label of the plot specified as a string
- set_ylabel(—string‖) — axes level method used to set the y-label of the plot specified as a string
- scatter3D(x-axis values, y-axis values) — plots a three-dimensional scatter plot with x-axis values against y-axis values
- plot3D(x-axis values, y-axis values) — plots a three-dimensional line graph with x-axis values against y-axis values

Basic Examples:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

plt.plot(x, y)
plt.xlabel('X-axis')
```

```
plt.ylabel('Y-axis')
plt.title('Line Plot')
plt.show()

plt.scatter(x, y, color='red', marker='x')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.show()

x = ['A', 'B', 'C', 'D', 'E']
y = [10, 20, 15, 25, 30]

plt.bar(x, y, color='blue')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot')
plt.show()

import numpy as np
data = np.random.normal(0, 1, 1000)
plt.hist(data, bins=30, color='green', alpha=0.7)
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Histogram')
plt.show()

from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

```
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
z = [1, 2, 3, 4, 5]

ax.scatter(x, y, z, c='r', marker='o')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')

plt.show()
```

Some Important Machine Learning Libraries in Python:

There are several popular Python libraries and frameworks specifically designed for machine learning. Here is a list of some widely used ones:

Scikit-learn:

Website: https://scikit-learn.org/

Description: Scikit-learn is one of the most popular and widely used libraries for machine learning. It provides a wide range of algorithms for tasks like classification, regression, clustering, dimensionality reduction, and more.

TensorFlow:

Website: https://www.tensorflow.org/

Description: Developed by Google, TensorFlow is an open-source library primarily used for deep learning. It provides a flexible architecture for building and training various types of neural networks.

Keras:

Website: https://keras.io/

Description: Keras is a high-level neural networks API that runs on top of TensorFlow. It provides an easy-to-use interface for building and training deep learning models.

PyTorch:

Website: https://pytorch.org/

Description: PyTorch is an open-source deep learning library that provides a flexible and dynamic computational graph system. It's known for its ease of use and dynamic computation capabilities.

XGBoost:

Website: https://xgboost.ai/

Description: XGBoost is an optimized and efficient gradient boosting library. It's particularly effective for handling structured data and is widely used in machine learning competitions.

LightGBM:

Website: https://lightgbm.readthedocs.io/

Description: LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It's designed for distributed and efficient training of large datasets.

CatBoost:

Website: https://catboost.ai/

Description: CatBoost is a gradient boosting library that's well-suited for categorical data. It automatically handles categorical features, reducing the need for preprocessing.

NLTK (Natural Language Toolkit):

Website: https://www.nltk.org/

Description: NLTK is a leading platform for building Python programs to work with human language data. It's widely used in natural language processing (NLP) tasks, which are a subset of machine learning.
