Class 1: Introduction to Data Science

Objectives

- Understand Data Science prospects in Bangladesh (BD).
- Learn career guidelines for becoming a Data Scientist.
- Explore how this bootcamp accelerates your goals.
- Introduce Data Science, Jupyter Notebooks, and NumPy basics.

1 Prospects of Data Science in Bangladesh

Data Science is growing in BD due to:

- Booming tech sector (e.g., bKash, Pathao).
- High-demand roles: data analysts, data scientists.
- Digital Bangladesh initiatives driving data use.
- Freelancing opportunities on global platforms.

2 Career Guidelines

To become a Data Scientist:

1. Learn math (statistics, linear algebra) and Python.
2. Master tools: NumPy, Pandas, Tableau.
3. Gain experience via Kaggle, internships.
4. Study machine learning algorithms.
5. Develop communication and teamwork skills.
6. Stay updated with trends and certifications.

3 Bootcamp Benefits

This bootcamp offers:

- Structured curriculum: Python to machine learning.
- Hands-on projects with Jupyter and NumPy.
- Local context and career support.

4 Introduction to Data Science

Data Science extracts insights from raw data for decision-making. A Data Scientist:

1. Manages and preprocesses data.
2. Engineers features and selects models.
3. Evaluates and deploys models.
4. Monitors performance in production.

5 Jupyter Notebooks

Interactive tool for coding and visualization.

- Install: `pip install jupyter-notebook`.
- Alternative: Google Colab (cloud-based).
- Usage: Run `jupyter notebook`, execute cells with Shift+Enter.

6 NumPy Basics

NumPy (Numerical Python) is a Python library for efficient numerical computations with arrays. It is essential for data science because:

- **Performance**: Up to 50x faster than Python lists due to optimized C-based code.
- **Vectorization**: Allows operations on entire arrays without loops, simplifying code.
- **Interoperability**: Works seamlessly with Pandas, Scikit-learn, and other libraries.
- **Applications**: Used for data preprocessing, statistical analysis, and linear algebra.

Key Concepts

- **Arrays**: Multi-dimensional data structures for numerical data.
- **Properties**: `dtype` (data type, e.g., `int64`), `shape` (dimensions, e.g., `(6,)`), `ndim` (number of dimensions, e.g., 1 for a vector).
- **Operations**: Element-wise arithmetic, reshaping, and indexing.

Code Examples with Explanations

The following examples demonstrate how to use NumPy arrays. Each snippet includes comments to explain what the code does and why it's useful for data science.

```
import numpy as np
# Creating an array from a Python list
# Converts a list to a NumPy array for faster numerical operations
my_list = [1, 2, 5, 3, 7, 8]
my_np_array = np.array(my_list)
# Check type (numpy.ndarray), data type (int64), shape (6,), and dimensions (1)
print(my_np_array)  # Output: [1 2 5 3 7 8]

# Array of zeros
# Useful for initializing arrays, e.g., for placeholders in data processing
zeros = np.zeros(shape=(5,))  # Creates a 1D array of 5 zeros
print(zeros)  # Output: [0. 0. 0. 0. 0.]

# Array of ones
# Used in matrix operations or as a starting point for calculations
ones = np.ones(shape=(10,2), dtype=int)  # 10x2 array of 1s (integer type)
print(ones)  # Output: [[1 1], [1 1], ..., [1 1]]

# Array with a specific value
# Handy for initializing arrays with a constant, e.g., for bias terms
full = np.full(shape=(10,), fill_value=15)  # 1D array of 10 elements, all 15
print(full)  # Output: [15 15 ... 15]

# Identity matrix
# Essential for linear algebra, e.g., in machine learning algorithms
id_matrix = np.eye(4)  # 4x4 matrix with 1s on diagonal, 0s elsewhere
print(id_matrix)  # Output: [[1. 0. 0. 0.], [0. 1. 0. 0.], ...]

# Empty array
# Creates an array with uninitialized values (faster but contains random data)
empty = np.empty(shape=(5,))  # 1D array with 5 random values
print(empty)  # Output: [random values]

# Type casting
# Converts array to another data type, e.g., for memory efficiency
empty_int = empty.astype(int)  # Converts float64 to int64
print(empty_int)  # Output: [integer values]

# Sequential array
# Generates a sequence of numbers, useful for creating indices or ranges
seq = np.arange(start=1, stop=10, step=2)  # Numbers from 1 to 9, step 2
print(seq)  # Output: [1 3 5 7 9]

# Linearly spaced array
# Creates evenly spaced numbers, useful for plotting or simulations
lin = np.linspace(start=0, stop=10, num=5)  # 5 numbers from 0 to 10
print(lin)  # Output: [0. 2.5 5. 7.5 10.]

# Element-wise operations
# Performs calculations on all elements, e.g., for feature scaling
arr = np.array([1, 2, 3, 4])
```

```
squared = arr ** 2  # Squares each element
print(squared)  # Output: [1 4 9 16]

# Array aggregation
# Computes summary statistics, e.g., for data analysis
sum_arr = arr.sum()  # Sums all elements
print(sum_arr)  # Output: 10

# Reshaping arrays
# Changes array shape, e.g., for matrix operations in machine learning
matrix = np.arange(6).reshape(2, 3)  # Converts 1D array to 2x3 matrix
print(matrix)  # Output: [[0 1 2], [3 4 5]]
```

Summary

Data Science is thriving in BD. This bootcamp equips you with skills in Python, NumPy, and Jupyter Notebooks. Practice these tools and explore local opportunities.