Day 1: Foundations of Data Science

---

1. Overview of Data Science

Introduction

Data Science is an interdisciplinary field focused on extracting knowledge and insights from data, both structured and unstructured. It integrates statistical analysis, computer science, and domain expertise to address real-world challenges and enable data-driven decision-making.

Technical Explanation

Core Components:

Data Collection:

Gathering data through APIs, databases, web scraping, or sensors.

Example tools: Selenium (web scraping), PostgreSQL (databases).

Python package: pip install requests (for API calls).

Data Cleaning:

Techniques for handling missing values, duplicates, and outliers.

Libraries: pandas, scikit-learn for imputation.

Python package: pip install pandas scikit-learn.

Data Analysis:

Using descriptive and inferential statistics to uncover patterns.

Tools: scipy, statsmodels.

Python package: pip install scipy statsmodels.

Modeling:

Applying machine learning models for predictions or classifications.

Libraries: scikit-learn, xgboost.

Python package: pip install scikit-learn xgboost.

Visualization:

Libraries: matplotlib, seaborn, plotly, dash.

Python package: pip install matplotlib seaborn plotly dash.

Key Technologies:

Programming Languages: Python, R, SQL.

Tools: Jupyter Notebook, Tableau, Power BI.

Libraries: Pandas, NumPy, Scikit-learn, TensorFlow.

Realtime Example

An online education platform analyzes student activity logs to predict the likelihood of course completion and provide personalized recommendations.

Practical Implementation in Python

```
# Install required libraries
# pip install pandas

Import pandas as pd

# Example: Reading sales data
Data = pd.read_csv('sales_data.csv')
Print("Sample Data:")
Print(data.head())

# Summary statistics
Print("\nSummary Statistics:")
Print(data.describe())
```

---

2. Core Python for Data Science

Introduction

Python is a versatile programming language, particularly favored in Data Science for its simplicity, extensive libraries, and vibrant community support. Its ecosystem supports tasks like data manipulation, numerical computation, and data visualization.

Technical Explanation

NumPy: Handles numerical operations efficiently with arrays.

Installation: pip install numpy.

Key Features: Element-wise operations, matrix manipulations.

Pandas: Provides high-performance data manipulation tools.

Installation: pip install pandas.

Key Features: DataFrames for structured data, groupby, and merging datasets.

Matplotlib/Seaborn: Used for static and advanced visualizations.

Installation: pip install matplotlib seaborn.

Realtime Example

A healthcare provider uses Pandas to clean patient data and Seaborn to visualize disease trends.

Practical Implementation in Python

```
# Install required libraries
# pip install numpy pandas matplotlib seaborn

Import numpy as np
Import pandas as pd
Import matplotlib.pyplot as plt
Import seaborn as sns

# NumPy Example
Arr = np.array([1, 2, 3, 4, 5])
Print("NumPy Array:", arr)

# Pandas Example
Data = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie'], 'Score': [85, 92, 78]})
Print("\nPandas DataFrame:\n", data)

# Seaborn Visualization
Plt.figure(figsize=(8, 5))
Sns.barplot(data=data, x='Name', y='Score')
Plt.title("Scores by Name")
Plt.show()
```

3. Exploratory Data Analysis (EDA)

Introduction

EDA involves summarizing and visualizing data to identify patterns, detect anomalies, and extract useful insights. It lays the foundation for effective modeling and decision-making.

Technical Explanation

Univariate Analysis:

Focus on one variable (e.g., distribution, outliers).

Tools: Histogram, box plot.

Bivariate Analysis:

Examines relationships between two variables (e.g., scatterplot, correlation).

Tools: Heatmaps, pairplots.

Multivariate Analysis:

Explores interactions between multiple variables (e.g., PCA, clustering).

Realtime Example

Analyzing COVID-19 data to understand the relationship between infection rates and vaccination coverage across regions.

Practical Implementation in Python

```
# Install required libraries
# pip install pandas seaborn matplotlib

Import pandas as pd
Import seaborn as sns
Import matplotlib.pyplot as plt

# Load a dataset
Df = pd.read_csv('covid_data.csv')

# Univariate Analysis: Histogram
Sns.histplot(df['Cases'], bins=30, kde=True)
Plt.title("Distribution of COVID-19 Cases")
```

```
Plt.show()

# Bivariate Analysis: Scatter Plot
Sns.scatterplot(data=df, x='VaccinationRate', y='Cases', hue='Region')
Plt.title("Vaccination Rate vs COVID-19 Cases")
Plt.show()

# Multivariate Analysis: Heatmap
Correlation = df.corr()
Sns.heatmap(correlation, annot=True, cmap='coolwarm')
Plt.title("Correlation Heatmap")
Plt.show()
```

---

4. Interactive Dashboards with Plotly

Introduction

Plotly is a versatile Python library for creating interactive visualizations. Dashboards created with Plotly allow users to dynamically explore data through zooming, filtering, and hovering.

Technical Explanation

Core Features:

Supports multiple chart types: Line, scatter, bar, pie, maps.

Offers interactivity: Zoom, hover tooltips, filters.

Compatible with Dash for building web apps.

Realtime Example

An investment firm creates an interactive dashboard to track stock price trends and portfolio performance.

Practical Implementation in Python

```
# Install required libraries
# pip install pandas plotly

Import pandas as pd
Import plotly.express as px

# Load a dataset
Df = pd.read_csv('stock_data.csv')

# Create an interactive line chart
```

```
Fig = px.line(df, x='Date', y='Close', title='Stock Price Over Time')

Fig.show()


# Create an interactive bar chart

Sector_perf = df.groupby('Sector')['Performance'].mean().reset_index()

Fig = px.bar(sector_perf, x='Sector', y='Performance', title='Average Performance by Sector')

Fig.show()
```