U23CBT41

FOUNDATIONS OF DATA SCIENCE

Comprehensive Learning Material

All Five Units — 45 Periods

Course Code	Credits	L	T	P
U23CBT41	3	3	0	0

Topics Covered

Unit	Title	Periods
Unit I	Introduction to Data Science	9
Unit II	Describing Data	9
Unit III	Describing Relationships	9
Unit IV	Python Libraries for Data Wrangling	9
Unit V	Data Visualization	9

Course Objectives

The main learning objective of this course is to prepare the students to become proficient in the fundamental concepts, tools, and techniques of Data Science. Upon completing this course, students will be able to:

1. Understand the data science fundamentals and process — including the complete pipeline from data acquisition to presentation.
2. Describe data for the data science process — using appropriate statistical measures, charts, and tables.
3. Describe the relationship between data — using correlation, regression, and scatter plots.
4. Utilize Python libraries for Data Wrangling — leveraging NumPy and Pandas for efficient data manipulation.
5. Present and interpret data using visualization libraries in Python — creating insightful plots with Matplotlib and Seaborn.

How to Use This Material

This learning material is organized unit-by-unit, following the course syllabus. Each unit begins with a learning overview, proceeds through detailed topic explanations with definitions, examples, and tables, and ends with a summary and review questions. Readers are encouraged to:

- Read each section carefully before attending the lecture.
- Attempt the review questions at the end of each unit.
- Practice Python examples using Jupyter Notebook or Google Colab.
- Refer to the key terms highlighted in red throughout the material.

UNIT I
INTRODUCTION TO DATA SCIENCE

1.1 What is Data Science?

Data Science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data. It combines statistics, computer science, and domain expertise to analyze and interpret complex datasets.

Data science is not a single discipline — it is a convergence of skills including mathematics, statistics, programming, database management, and subject-matter knowledge. The goal of data science is to turn raw data into actionable insights that drive decisions.

1.1.1 Benefits and Uses of Data Science

Data science has transformed virtually every industry and field. Its key benefits and applications include:

- Business Intelligence: Companies use data science to analyze customer behavior, forecast demand, personalize marketing, and optimize operations.
- Healthcare: Predictive models help diagnose diseases early, optimize treatment plans, and manage hospital resources.
- Finance: Fraud detection systems, risk assessment models, and algorithmic trading rely heavily on data science.
- Social Media: Recommendation systems on platforms like Netflix and YouTube are driven by data science algorithms.
- Government and Policy: Data science enables evidence-based policy making, smart city planning, and public health monitoring.
- Science and Research: Analyzing experimental data, identifying patterns in astronomical observations, and genomics research all rely on data science methods.

1.1.2 Facets of Data

Data can be characterized along several important dimensions — together these are called the facets of data:

Facet	Description	Example
Volume	The amount of data generated and stored	Terabytes of social media posts
Variety	Different types and forms of data	Text, images, videos, sensor readings
Velocity	Speed at which new data is generated	Real-time stock market feeds
Veracity	Accuracy and trustworthiness of data	Removing duplicates and errors
Value	Usefulness of data for decision	Identifying buying patterns

	making	

1.2 The Data Science Process

The data science process is a structured workflow that transforms raw data into meaningful insights. It consists of several well-defined stages:

1.2.1 Overview of the Data Science Process

Stage	Description
1. Define Research Goals	Identify the problem, formulate hypotheses, and set measurable objectives.
2. Retrieve Data	Collect data from databases, APIs, web scraping, surveys, or sensors.
3. Data Preparation	Clean, transform, and format data for analysis (also called data wrangling).
4. Exploratory Data Analysis (EDA)	Summarize, visualize, and understand the data's structure and patterns.
5. Build the Model	Apply machine learning or statistical models to generate predictions or insights.
6. Present Findings	Communicate results clearly through reports, dashboards, or visualizations.
7. Build Applications	Deploy the model into production for automated decision-making.

1.2.2 Defining Research Goals

The most critical and often underestimated step is defining clear research goals. A well-defined goal answers:

- What is the business or scientific problem we are trying to solve?
- What data do we need? Is it available?
- What does success look like? How will we measure it?
- What are the constraints (time, budget, privacy)?

 Note: A poorly defined problem leads to wasted effort and irrelevant results, even if the analysis is technically correct.

1.2.3 Retrieving Data

Data can be retrieved from many sources. Common sources include:

- Relational databases (SQL): MySQL, PostgreSQL, Oracle.
- APIs: Twitter API, Google Maps API, financial data APIs.
- Web scraping: Using libraries like BeautifulSoup or Scrapy.

- Public datasets: UCI Machine Learning Repository, Kaggle, government data portals.
- Flat files: CSV, JSON, XML, Excel spreadsheets.

1.2.4 Data Preparation

Raw data is rarely analysis-ready. Data preparation (or data wrangling) involves:

- Handling missing values — imputation or removal.
- Removing duplicates.
- Correcting data types (e.g., dates stored as strings).
- Normalizing or standardizing numerical features.
- Encoding categorical variables.
- Removing or capping outliers.

Data preparation typically consumes 60–80% of a data scientist's time.

1.2.5 Exploratory Data Analysis (EDA)

EDA is the process of exploring data to understand its structure, spot anomalies, test assumptions, and discover patterns. Key techniques include:

- Summary statistics: mean, median, standard deviation, range.
- Visualizations: histograms, box plots, scatter plots, heatmaps.
- Correlation analysis to identify relationships between variables.
- Distribution checks: are variables normally distributed?

1.2.6 Building the Model

Model building involves selecting and applying an appropriate algorithm. Common approaches include:

- Regression models for predicting continuous outcomes.
- Classification models for predicting categories.
- Clustering for grouping similar data points.
- Neural networks for complex pattern recognition.

1.2.7 Presenting Findings and Building Applications

Results must be communicated clearly to both technical and non-technical stakeholders. Tools include:

- Data visualization libraries (Matplotlib, Seaborn, Plotly).
- Business intelligence dashboards (Tableau, Power BI).
- Automated reports and notebooks (Jupyter Notebook).
- Deployed APIs (Flask, FastAPI) for real-time predictions.

1.3 Data Mining

Data Mining is the computational process of discovering patterns in large datasets. It involves methods from statistics, machine learning, and database systems to extract useful information from data.

Data Mining: The practice of automatically searching large stores of data to discover patterns and trends that go beyond simple analysis.

Key data mining tasks include:

- Classification: Assigning items to predefined categories (e.g., spam detection).
- Regression: Predicting a continuous value (e.g., house prices).
- Clustering: Grouping data without predefined labels (e.g., customer segmentation).
- Association Rule Mining: Finding items that frequently co-occur (e.g., market basket analysis).
- Anomaly Detection: Identifying unusual data points (e.g., fraud detection).

1.4 Data Warehousing

A Data Warehouse is a central repository of integrated data from one or more disparate sources. It is designed for query and analysis rather than transaction processing.

Data Warehouse: A large collection of data gathered from multiple sources and organized for business intelligence and analytical reporting.

Feature	Data Warehouse	Operational Database
Purpose	Analysis and reporting	Day-to-day transactions
Data Type	Historical, aggregated	Current, detailed
Users	Analysts, executives	Operational staff
Queries	Complex analytical	Simple, repetitive
Update Frequency	Periodic (batch)	Continuous (real-time)

Key Components of a Data Warehouse

- ETL Process (Extract, Transform, Load): Moves data from source systems to the warehouse.
- Data Marts: Subsets of the data warehouse for specific departments.
- OLAP (Online Analytical Processing): Enables multidimensional analysis.
- Metadata Repository: Stores information about the data.

1.5 Basic Statistical Descriptions of Data

Statistical descriptions help us summarize and understand datasets quickly. They fall into several categories:

1.5.1 Measures of Central Tendency

Mean: The arithmetic average of all values in a dataset. Calculated as sum of values divided by count.

Median: The middle value when data is sorted. Resistant to outliers.

Mode: The most frequently occurring value. Useful for categorical data.

1.5.2 Measures of Dispersion

Range: The difference between the maximum and minimum values.

Variance: The average of squared deviations from the mean. Measures spread.

Standard Deviation: The square root of variance. Expressed in the same units as the data.

Interquartile Range (IQR): The range of the middle 50% of data (Q3 - Q1). Robust to outliers.

1.5.3 Measures of Shape

Skewness: Measures asymmetry of the distribution. Positive skew = tail on the right.

Kurtosis: Measures peakedness of the distribution. High kurtosis = heavy tails.

Unit I — Review Questions

6. Define Data Science and explain its interdisciplinary nature.
7. List and explain the seven stages of the Data Science Process.
8. What is EDA? Why is it important before building a model?
9. Distinguish between Data Mining and Data Warehousing.
10. What are the five Vs of data? Give one example for each.
11. Define mean, median, and mode. When would you prefer median over mean?

UNIT II
DESCRIBING DATA

2.1 Types of Data

Understanding the types of data is fundamental to choosing appropriate statistical methods and visualizations. Data can be broadly classified into two main types:

Category	Type	Description	Example
Qualitative (Categorical)	Nominal	Categories with no natural order	Colors, blood types, gender
Qualitative (Categorical)	Ordinal	Categories with a meaningful order	Education level, satisfaction rating
Quantitative (Numerical)	Discrete	Countable, integer values	Number of students, goals scored
Quantitative (Numerical)	Continuous	Measurable, any value in a range	Height, weight, temperature

2.1.1 Qualitative Data

Qualitative (or categorical) data represents characteristics or attributes that cannot be measured numerically. It describes qualities or properties of an item.

- Nominal data: The categories have no inherent order. For example, types of cars (sedan, SUV, truck) or country of birth. Mathematical operations like addition or subtraction are meaningless for nominal data.
- Ordinal data: Categories have a meaningful order or rank, but the intervals between categories are not equal. For example, a customer satisfaction scale (Poor, Fair, Good, Excellent) is ordinal.

2.1.2 Quantitative Data

Quantitative data represents numerical measurements or counts. Arithmetic operations are meaningful on quantitative data.

- Discrete data: Can only take specific, separate values (usually whole numbers). For example, the number of cars in a parking lot.
- Continuous data: Can take any value within a range. For example, the exact temperature measured to many decimal places.

2.2 Types of Variables

In statistics and data science, the variables used in a study are classified based on their role:

Independent Variable: The variable that is manipulated or selected to observe its effect. Also called the explanatory or predictor variable.

Dependent Variable: The variable that is measured or observed. It depends on the independent variable. Also called the response variable.

Control Variable: A variable that is kept constant to prevent it from influencing the results.

Confounding Variable: An outside variable that influences both the independent and dependent variable, potentially distorting results.

Additional classification of variables:

Variable Type	Scale	Properties	Examples
Nominal	Categorical	Categories, no order, no arithmetic	Gender, religion, nationality
Ordinal	Categorical	Ordered categories, unequal intervals	Pain rating, rank in class
Interval	Numerical	Equal intervals, no true zero	Temperature in Celsius, IQ score
Ratio	Numerical	Equal intervals, true zero exists	Age, weight, income, distance

2.3 Describing Data with Tables and Graphs

Once data is collected, it must be organized and displayed effectively. Tables and graphs summarize data and reveal patterns that raw numbers cannot show.

2.3.1 Frequency Tables

A frequency table organizes data into categories and shows how often each category appears. It typically includes:

- Class/Category: The group or value.
- Frequency (f): The count of observations in each class.
- Relative Frequency (rf): The proportion of total observations — calculated as f / n.
- Cumulative Frequency: The running total of frequencies up to and including each class.

2.3.2 Types of Graphs

Graph Type	Best Used For	Key Feature
Bar Chart	Comparing categories of nominal/ordinal data	Bars represent frequency or value for each category
Histogram	Showing distribution of continuous data	Bars represent frequency within each interval; no gaps
Pie Chart	Showing parts of a whole	Each slice represents a percentage of the total
Line Graph	Showing trends over time	Points connected by lines to show change
Box Plot	Displaying distribution and	Shows min, Q1, median, Q3,

	outliers	max
Scatter Plot	Exploring relationship between two variables	Each point represents one observation (x, y)

 Note: Always label axes, provide a title, and cite the data source when creating graphs.

2.4 Describing Data with Averages

Averages (measures of central tendency) describe the center of a dataset. The three most important averages are the mean, median, and mode.

2.4.1 Arithmetic Mean

The arithmetic mean is calculated by summing all values and dividing by the number of observations. Formula: Mean = (Sum of all values) / n

The mean is sensitive to outliers. A single extremely large or small value can pull the mean significantly toward it.

2.4.2 Median

The median is the middle value of an ordered dataset. For an odd number of observations, the median is the value at position (n+1)/2. For an even number, it is the average of the two middle values.

The median is a robust measure — it is not affected by extreme values, making it preferred for skewed distributions (e.g., income data).

2.4.3 Mode

The mode is the value that occurs most frequently. A dataset can have:

- No mode: if all values occur equally often.
- One mode (unimodal): one value occurs most frequently.
- Two modes (bimodal): two values occur with the same highest frequency.
- Multiple modes (multimodal): more than two values share the highest frequency.

2.4.4 When to Use Each Average

Measure	Use When	Avoid When
Mean	Data is symmetrical and no extreme outliers	Data is heavily skewed or has outliers
Median	Data is skewed or has outliers	You need to use it in further calculations
Mode	Data is categorical or finding the most common value	Data is continuous with no repeats

2.5 Describing Variability

Measures of variability describe how spread out the data values are from the center. Two datasets can have the same mean but very different spreads.

2.5.1 Range

Range = Maximum value - Minimum value. Simple to calculate but highly sensitive to outliers — even one extreme value dramatically changes the range.

2.5.2 Variance

Variance measures the average squared deviation from the mean. A higher variance indicates that data points are more spread out from the mean.

Population Variance: The average of squared differences from the population mean (divide by N).

Sample Variance: When working with a sample (divide by n-1, applying Bessel's correction for unbiased estimation).

2.5.3 Standard Deviation

Standard deviation is the square root of variance. It is expressed in the same units as the original data, making it more interpretable than variance.

- A small standard deviation indicates data points cluster closely around the mean.
- A large standard deviation indicates data points are spread widely.

2.5.4 Interquartile Range (IQR)

IQR = Q3 - Q1, where Q1 is the 25th percentile and Q3 is the 75th percentile. IQR is resistant to outliers and represents the spread of the middle 50% of the data.

IQR is used to detect outliers: values below Q1 - 1.5*IQR or above Q3 + 1.5*IQR are considered outliers (Tukey's rule).

2.6 Normal Distribution and Standard (z) Scores

The Normal Distribution is one of the most important probability distributions in statistics. It is a symmetric, bell-shaped curve defined entirely by its mean and standard deviation.

2.6.1 Properties of the Normal Distribution

- Symmetrical: The left and right halves are mirror images.
- Mean = Median = Mode: All three measures of central tendency coincide at the center.
- 68-95-99.7 Rule (Empirical Rule):
- 68% of data falls within 1 standard deviation of the mean.
- 95% of data falls within 2 standard deviations.
- 99.7% of data falls within 3 standard deviations.
- The curve asymptotically approaches the x-axis but never touches it.

2.6.2 Standard (z) Scores

A z-score represents how many standard deviations a particular value is from the mean. Formula: z = (x - mean) / standard deviation

Z-scores are useful for:

- Comparing values from different distributions.
- Finding probabilities using the standard normal table.
- Identifying outliers (values with |z| > 3 are often considered extreme outliers).

z-Score	Interpretation
z = 0	Value equals the mean
z = 1	Value is 1 standard deviation above the mean
z = -2	Value is 2 standard deviations below the mean
|z| > 3	Potential outlier — very unusual observation

Unit II — Review Questions

12. Distinguish between quantitative and qualitative data with examples.
13. What is the difference between nominal and ordinal scales?
14. Explain the difference between a histogram and a bar chart.
15. When is the median a better measure of central tendency than the mean?
16. Define variance and standard deviation. Why is standard deviation more useful?
17. What does a z-score of -1.5 mean? Explain the 68-95-99.7 empirical rule.

UNIT III
DESCRIBING RELATIONSHIPS

3.1 Correlation

Correlation measures the strength and direction of the linear relationship between two quantitative variables. It is one of the most widely used statistical tools in data science.

Correlation: A statistical measure that expresses the extent to which two variables change together. A positive correlation means both increase together; a negative correlation means one increases as the other decreases.

3.1.1 Types of Correlation

Type	Description	Example
Positive Correlation	Both variables increase together	Height and weight; study time and grades
Negative Correlation	One variable increases as the other decreases	Speed and travel time; stress and sleep
Zero / No Correlation	No linear relationship between the variables	Shoe size and intelligence
Perfect Positive (r=+1)	All data points lie exactly on an upward line	Rare in real data
Perfect Negative (r=-1)	All data points lie exactly on a downward line	Rare in real data

3.2 Scatter Plots

A scatter plot is a graph that displays the relationship between two quantitative variables. Each point on the plot represents one observation, with the x-axis representing one variable and the y-axis representing the other.

How to Read a Scatter Plot

- Direction: Does the cluster of points slope upward (positive) or downward (negative)?
- Strength: How tightly do the points cluster around an imaginary straight line? Tight cluster = strong relationship.
- Form: Is the pattern linear or curved?
- Outliers: Are there any points that deviate dramatically from the overall pattern?

 Note: Scatter plots can reveal non-linear patterns that correlation coefficients (which measure only linear relationships) would miss.

3.3 Correlation Coefficient for Quantitative Data

The Pearson Correlation Coefficient (r) is a numerical measure of the linear relationship between two quantitative variables. It ranges from -1 to +1.

Value of r	Strength of Relationship	Direction
+0.90 to +1.00	Very Strong	Positive
+0.70 to +0.89	Strong	Positive
+0.40 to +0.69	Moderate	Positive
+0.10 to +0.39	Weak	Positive
0.00 to +0.09	Negligible / None	—
-0.10 to -0.39	Weak	Negative
-0.40 to -0.69	Moderate	Negative
-0.70 to -0.89	Strong	Negative
-0.90 to -1.00	Very Strong	Negative

Important Caution: Correlation vs. Causation

A high correlation between two variables does NOT imply that one causes the other. This is one of the most important principles in data science and statistics.

Example: Ice cream sales and drowning rates are positively correlated — but ice cream does not cause drowning. Both are driven by a third variable: hot weather (a confounding variable).

> *Note: Always look for possible confounding variables and use controlled experiments to establish causation.*

3.4 Computational Formula for Correlation Coefficient

The Pearson correlation coefficient r is calculated using the following steps:

18. Calculate the mean of X and Y.
19. For each pair (xi, yi), compute the deviation from the mean: (xi - x_bar) and (yi - y_bar).
20. Multiply the deviations for each pair and sum them to get the cross-product sum.
21. Calculate the sum of squared deviations for X and Y separately.
22. Divide the cross-product sum by the square root of the product of the two sums of squares.

This calculation produces r, bounded between -1 and +1.

3.5 Regression

While correlation measures the strength of a relationship, regression goes further — it allows us to predict the value of one variable based on the value of another.

Regression: A statistical method for modeling the relationship between a dependent variable (outcome) and one or more independent variables (predictors).

3.5.1 The Regression Line

The regression line (or line of best fit) is the straight line that best summarizes the linear relationship between two variables. It is expressed as:

y = a + bx where y = predicted dependent variable, x = independent variable, b = slope, a = y-intercept.

- The slope (b) tells us how much y changes for each one-unit increase in x.
- The y-intercept (a) tells us the predicted value of y when x = 0.

3.5.2 Least Squares Regression Line

The least squares method finds the regression line that minimizes the sum of squared vertical distances (residuals) between the observed data points and the line. This is the most common criterion for fitting a regression line.

Residual: The difference between an observed value and the value predicted by the regression line. Residual = y_observed - y_predicted.

The least squares regression line is the unique line for which the sum of all squared residuals is as small as possible.

3.5.3 Standard Error of Estimate

The standard error of estimate (Se) measures the typical size of prediction errors — how spread out the data points are around the regression line. A smaller Se indicates a better-fitting model with more accurate predictions.

3.6 Interpretation of r^2

The coefficient of determination (r^2) is the square of the correlation coefficient. It represents the proportion of the variance in the dependent variable that is explained by the independent variable.

r^2 Value	Interpretation
$r^2 = 1.00$	100% of variation in y is explained by x — perfect fit
$r^2 = 0.81$	81% of variation in y is explained by x — strong predictive model
$r^2 = 0.49$	49% of variation in y is explained by x — moderate model
$r^2 = 0.09$	Only 9% of variation in y is explained — weak model
$r^2 = 0.00$	x explains none of the variation in y

3.7 Multiple Regression Equations

Multiple regression extends simple regression to include two or more independent variables. The multiple regression equation is:

$\mathbf{y = a + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n}$ where each b_i is the partial regression coefficient for predictor x_i.

In multiple regression:

- Each coefficient b_i represents the effect of one predictor while holding all other predictors constant.
- R^2 (multiple coefficient of determination) measures the proportion of variance in y explained by all predictors combined.
- Adjusted R^2 accounts for the number of predictors, penalizing for adding variables that don't improve the model.

Assumptions of Linear Regression

- Linearity: The relationship between predictors and outcome is linear.
- Independence: Observations are independent of each other.
- Homoscedasticity: Residuals have constant variance.
- Normality: Residuals are approximately normally distributed.
- No Multicollinearity: Predictors are not highly correlated with each other.

3.8 Regression Towards the Mean

Regression towards the mean is a phenomenon where extreme values on a first measurement tend to be followed by less extreme values on a second measurement. This was first observed by Sir Francis Galton.

Example: Parents who are very tall tend to have children who are tall, but on average slightly shorter than themselves — their children's heights regress towards the population mean.

> *Note: This phenomenon has important implications for medical trials and performance evaluation — improvement after treatment may partly reflect regression to the mean rather than a true treatment effect.*

Unit III — Review Questions

23. Define correlation. What is the difference between positive and negative correlation?
24. How do you read a scatter plot? What four features should you look for?
25. What does r = 0.85 tell you about a relationship? What about r = -0.30?
26. Explain the principle that correlation does not imply causation with an example.
27. What is the least squares regression line? What does it minimize?
28. If $r^2 = 0.64$, what percentage of variance in y is explained by x?
29. What is multiple regression? When would you use it over simple regression?

UNIT IV
PYTHON LIBRARIES FOR DATA WRANGLING

4.1 Introduction to NumPy

NumPy (Numerical Python) is the foundational library for numerical computing in Python. It provides efficient multi-dimensional array objects (ndarray), mathematical functions, and tools for working with large datasets.

ndarray: NumPy's primary data structure — an N-dimensional array of fixed-type elements stored contiguously in memory, enabling vectorized operations without explicit Python loops.

4.1.1 Why NumPy?

- Speed: NumPy operations are implemented in C and run up to 100x faster than equivalent Python loops.
- Memory Efficiency: NumPy arrays store data in a contiguous block, unlike Python lists.
- Vectorization: Operations are applied element-wise without explicit loops.
- Broadcasting: Operations between arrays of different shapes are handled automatically.

4.1.2 Creating NumPy Arrays

NumPy arrays can be created in several ways:

- np.array([1, 2, 3, 4, 5]) — Create a 1D array from a Python list.
- np.zeros((3, 4)) — Create a 3x4 array filled with zeros.
- np.ones((2, 3)) — Create a 2x3 array filled with ones.
- np.arange(0, 10, 2) — Create array [0, 2, 4, 6, 8] with step size 2.
- np.linspace(0, 1, 5) — Create 5 evenly-spaced values from 0 to 1.
- np.random.rand(3, 3) — Create a 3x3 array with random values from [0, 1).

4.1.3 Aggregations

NumPy provides efficient aggregation functions that summarize array data:

Function	Description	Example Output
np.sum(arr)	Sum of all elements	Sum = 15 for [1,2,3,4,5]
np.mean(arr)	Arithmetic mean	Mean = 3.0
np.std(arr)	Standard deviation	Spread of values
np.min(arr)	Minimum value	1
np.max(arr)	Maximum value	5
np.median(arr)	Median value	3.0
np.cumsum(arr)	Cumulative sum	[1, 3, 6, 10, 15]

 Note: Axis parameter: np.sum(arr, axis=0) sums along columns; axis=1 sums along rows.

4.1.4 Computations on Arrays

NumPy supports element-wise arithmetic operations and universal functions (ufuncs):

- Arithmetic: arr1 + arr2, arr1 * arr2 — applied element-wise.
- Exponentiation: np.power(arr, 2) squares each element.
- Square root: np.sqrt(arr).
- Trigonometric: np.sin(arr), np.cos(arr), np.tan(arr).
- Logarithmic: np.log(arr), np.log2(arr), np.log10(arr).

4.1.5 Comparisons, Masks, and Boolean Logic

NumPy supports element-wise comparisons that return Boolean arrays, which can be used as masks to filter data:

- arr > 3 returns array([False, False, False, True, True]) for [1,2,3,4,5].
- arr[arr > 3] returns array([4, 5]) — filtering using a Boolean mask.
- Logical operators: np.logical_and(mask1, mask2), np.logical_or(mask1, mask2).
- any() and all(): Check if any or all elements satisfy a condition.

4.1.6 Fancy Indexing

Fancy indexing allows selection of multiple arbitrary elements using index arrays:

- arr[[0, 2, 4]] — selects elements at indices 0, 2, and 4.
- arr[np.array([True, False, True, False, True])] — Boolean fancy indexing.
- For 2D arrays: arr[[0, 2], :][:, [1, 3]] — selects specific rows and columns.

4.1.7 Structured Arrays

Structured arrays allow storing heterogeneous data (different data types) in a NumPy array, similar to a spreadsheet with named columns:

- Define dtype: dtype = [('name', 'U10'), ('age', int), ('weight', float)]
- Create: data = np.zeros(3, dtype=dtype)
- Access by field name: data['name'], data['age']

4.2 Data Manipulation with Pandas

Pandas is a powerful data manipulation library built on top of NumPy. It provides two primary data structures — Series (1D) and DataFrame (2D) — that make working with structured data intuitive and efficient.

Series: A one-dimensional labeled array capable of holding any data type. The labels form the index.

DataFrame: A two-dimensional tabular data structure with labeled rows (index) and columns. Think of it as a spreadsheet or SQL table in Python.

4.2.1 Data Indexing and Selection

Pandas provides multiple ways to select and access data:

Method	Usage	Description
df['column']	Column selection	Returns a Series for the named column
df[['col1','col2']]	Multiple columns	Returns a DataFrame with selected columns
df.loc[label]	Label-based indexing	Select by row label and column name
df.iloc[index]	Integer-based indexing	Select by row/column integer position
df.at[row, col]	Single cell (label)	Fast access to a single element by label
df.iat[r, c]	Single cell (integer)	Fast access to single element by position

4.2.2 Operating on Data

- Arithmetic: df['col'] * 2, df['col1'] + df['col2'] — element-wise operations.
- Apply functions: df['col'].apply(lambda x: x**2) — apply a function to each element.
- Sorting: df.sort_values('column', ascending=True) — sort rows by column value.
- Filtering: df[df['age'] > 25] — filter rows based on a condition.
- String operations: df['name'].str.upper() — string methods via .str accessor.

4.2.3 Missing Data

Real-world datasets almost always contain missing values, represented in Pandas as NaN (Not a Number). Handling missing data is a critical step in data preparation.

Method	Description
df.isnull()	Returns a Boolean DataFrame marking missing values as True
df.isnull().sum()	Count of missing values per column
df.dropna()	Remove rows with any missing values
df.dropna(axis=1)	Remove columns with any missing values
df.fillna(value)	Fill missing values with a specified value
df.fillna(method='ffill')	Forward fill — propagate last valid value
df.fillna(df.mean())	Fill missing values with column mean
df.interpolate()	Interpolate missing values (useful for time series)

4.2.4 Hierarchical Indexing (MultiIndex)

Hierarchical indexing allows multiple levels of indexing on a single axis, enabling representation of higher-dimensional data in a 2D DataFrame.

- Created with pd.MultiIndex.from_tuples() or using groupby followed by aggregation.
- Access with df.loc[('level1_label', 'level2_label')].
- Reset index with df.reset_index() to convert index levels back to columns.
- Useful for panel data, time-series data with multiple entities.

4.2.5 Combining Datasets

Pandas provides powerful tools for combining multiple DataFrames:

Method	Description	SQL Equivalent
pd.concat([df1, df2])	Stack DataFrames vertically or horizontally	UNION ALL
pd.merge(df1, df2, on='key')	Join DataFrames on a common column	JOIN
df1.join(df2)	Join on index by default	JOIN on index
df.append(df2)	Append rows to a DataFrame (deprecated)	INSERT

Types of merges (SQL-style joins):

- inner: Only matching rows from both DataFrames (default).
- outer: All rows from both DataFrames; NaN where no match.
- left: All rows from the left DataFrame; NaN for missing right.
- right: All rows from the right DataFrame; NaN for missing left.

4.2.6 Aggregation and Grouping

The groupby operation follows a split-apply-combine strategy:

- Split: Divide the DataFrame into groups based on one or more columns.
- Apply: Apply a function to each group independently (e.g., sum, mean, count).
- Combine: Merge the results back into a single DataFrame.

Example: df.groupby('Department')['Salary'].mean() — average salary per department.

Multiple aggregations: df.groupby('City').agg({'Sales': 'sum', 'Profit': 'mean'})

4.2.7 Pivot Tables

Pivot tables summarize data by reorganizing it into a two-dimensional format with aggregations. They are similar to Excel pivot tables.

pd.pivot_table(df, values='Sales', index='Region', columns='Product', aggfunc='sum', fill_value=0)

- values: The column to aggregate.
- index: The column(s) to use as row labels.
- columns: The column(s) to use as column headers.
- aggfunc: The aggregation function (default is mean).

- fill_value: Value to use for missing cells.

Unit IV — Review Questions

30. What is NumPy? How does it differ from a Python list?
31. Explain fancy indexing with an example. How is it different from regular indexing?
32. What is a Boolean mask? How is it used to filter data in NumPy?
33. What is a Pandas DataFrame? How do loc and iloc differ?
34. List four strategies for handling missing data in Pandas.
35. Explain the groupby split-apply-combine strategy with an example.
36. What are the four types of joins in Pandas merge? Describe each.

UNIT V
DATA VISUALIZATION

5.1 Introduction to Matplotlib

Matplotlib is Python's foundational data visualization library. It provides a MATLAB-like interface and allows creation of a wide variety of static, animated, and interactive plots.

Figure: The top-level container for all plot elements — the overall window or page.

Axes: The actual plot area within a Figure where data is plotted. A Figure can have multiple Axes (subplots).

5.1.1 Importing Matplotlib

The standard import convention:

- import matplotlib.pyplot as plt — provides the plt interface for creating plots.
- import numpy as np — used alongside Matplotlib to generate data.
- %matplotlib inline — in Jupyter Notebook, displays plots directly below the cell.

5.1.2 Basic Plotting Workflow

The standard Matplotlib workflow involves:

37. Create a Figure and Axes: fig, ax = plt.subplots()
38. Plot the data: ax.plot(x, y, color='blue', linestyle='--', marker='o')
39. Add labels and title: ax.set_xlabel(), ax.set_ylabel(), ax.set_title()
40. Add legend if needed: ax.legend()
41. Display or save: plt.show() or plt.savefig('plot.png', dpi=300)

5.2 Line Plots

Line plots are the most basic type of plot, connecting data points with straight lines. They are ideal for showing continuous data and trends over time.

- plt.plot(x, y) — basic line plot.
- Key parameters: color, linewidth, linestyle ('-', '--', ':', '-.'), marker ('o', 's', '^', 'D').
- Multiple lines: call plt.plot() multiple times or provide a 2D array.
- Use plt.xlim() and plt.ylim() to set axis ranges.

 Note: Line plots should only be used for continuous data. Connecting discrete categories with lines can be misleading.

5.3 Scatter Plots

Scatter plots display the relationship between two quantitative variables by placing dots at each (x, y) pair.

- plt.scatter(x, y) — basic scatter plot.

- c parameter: color each point based on a third variable (color-coded scatter).
- s parameter: size of each point — can encode a fourth variable (bubble chart).
- alpha parameter: transparency (0 = invisible, 1 = opaque) — useful for overplotting.
- cmap parameter: specify a colormap for continuous color encoding.

5.4 Visualizing Errors

Error bars show the uncertainty or variability in measurements. In data science, they represent confidence intervals, standard deviations, or standard errors.

- plt.errorbar(x, y, yerr=error) — add vertical error bars.
- xerr parameter for horizontal error bars.
- fmt parameter: format string for the data points (e.g., 'o' for circles).
- capsize parameter: length of the caps at the end of error bars.

Continuous error regions (shaded bands) are often preferred over error bars for continuous data:

- plt.fill_between(x, y_lower, y_upper, alpha=0.3) — shaded region between two curves.

5.5 Density and Contour Plots

Density and contour plots are used to visualize three-dimensional data on a two-dimensional plane.

5.5.1 Contour Plots

Contour plots display lines of constant z-value over an (x, y) plane — like topographic maps.

- plt.contour(X, Y, Z) — contour lines plot.
- plt.contourf(X, Y, Z) — filled contour plot (colors between lines).
- levels parameter: specify the number or values of contour lines.
- plt.colorbar() — add a color scale bar.

5.5.2 Density Plots (KDE)

Kernel Density Estimation (KDE) plots estimate the probability density function of a dataset, providing a smooth curve representation of the data distribution.

- Created with Seaborn: sns.kdeplot(data=df, x='column')
- Can be overlaid on histograms for comparison.
- Bandwidth parameter controls smoothness of the density curve.

5.6 Histograms

Histograms display the distribution of a single quantitative variable by dividing data into bins and showing the frequency or density of observations in each bin.

- plt.hist(data, bins=30) — basic histogram with 30 bins.
- density=True: Normalize so the total area equals 1 (shows probability density).
- histtype='stepfilled': Filled step histogram.
- Multiple histograms: plt.hist(data1, alpha=0.5) then plt.hist(data2, alpha=0.5) — overlapping histograms.

Histogram Property	What It Reveals
Shape (symmetric vs skewed)	Nature of the distribution
Number of peaks (modes)	Unimodal, bimodal, or multimodal distribution
Spread (narrow vs wide)	Variability of the data
Gaps or outliers	Unusual values or separated clusters

5.7 Legends, Colors, and Customization

5.7.1 Legends

Legends identify the different elements of a plot. Best practices:

- Add descriptive labels: plt.plot(x, y, label='Training Set')
- Display legend: plt.legend() or plt.legend(loc='upper right').
- loc options: 'best', 'upper left', 'upper right', 'lower left', 'lower right', 'center'.
- bbox_to_anchor parameter: Place legend outside the plot area.

5.7.2 Colors

Matplotlib supports multiple ways to specify colors:

- Named colors: 'blue', 'red', 'green', 'black', 'orange', 'purple'.
- Short codes: 'b' (blue), 'r' (red), 'g' (green), 'k' (black), 'w' (white).
- Hex codes: '#FF5733' for a specific shade of red-orange.
- RGB tuples: (0.1, 0.2, 0.5) for a dark blue.
- Colormaps (for continuous data): 'viridis', 'plasma', 'inferno', 'coolwarm', 'RdBu'.

5.7.3 Customization

Matplotlib provides extensive customization options:

- Figure size: plt.figure(figsize=(10, 6)) — width and height in inches.
- Font size: plt.rcParams['font.size'] = 14 — set globally.
- Grid: plt.grid(True, alpha=0.3) — add a light grid.
- Spine removal: ax.spines['top'].set_visible(False) — clean minimalist style.
- Tick formatting: ax.xaxis.set_major_formatter(plt.FuncFormatter(...))
- Style sheets: plt.style.use('seaborn-v0_8'), plt.style.use('ggplot')

5.8 Subplots

Subplots allow multiple plots to be displayed in a grid layout within a single figure.

- fig, axes = plt.subplots(nrows=2, ncols=3) — create a 2x3 grid of subplots.
- Access individual subplots: axes[0, 0], axes[1, 2].
- figsize=(12, 8) — set total figure size.
- plt.tight_layout() — automatically adjust spacing between subplots.

- plt.subplots_adjust(hspace=0.4, wspace=0.3) — manually control spacing.
- plt.subplot2grid((3,3), (0,0), colspan=3) — unequal-sized subplots.

5.9 Text and Annotation

Annotations add context and highlight important features of a plot.

- plt.text(x, y, 'Text') — add text at a specific (x, y) position.
- ax.annotate('Peak', xy=(x_peak, y_peak), xytext=(x_text, y_text), arrowprops=dict(arrowstyle='->')) — annotation with arrow.
- plt.title(), plt.xlabel(), plt.ylabel() — title and axis labels.
- fontsize, fontweight, color, ha (horizontal alignment), va (vertical alignment) — text formatting options.

5.10 Three-Dimensional Plotting

Matplotlib supports 3D plots through the mpl_toolkits.mplot3d module:

- from mpl_toolkits.mplot3d import Axes3D — import the 3D toolkit.
- fig = plt.figure(); ax = fig.add_subplot(111, projection='3d') — create 3D axes.
- ax.plot3D(x, y, z) — 3D line plot.
- ax.scatter3D(x, y, z, c=z, cmap='viridis') — 3D scatter plot.
- ax.plot_surface(X, Y, Z, cmap='coolwarm') — 3D surface plot from meshgrid data.
- ax.plot_wireframe(X, Y, Z) — wireframe 3D surface.
- ax.contour3D(X, Y, Z, 50) — 3D contour plot.

5.11 Geographic Data with Basemap

The Basemap toolkit (from mpl_toolkits.basemap) allows plotting geographic data on maps.

- Supports various map projections: Mercator, Lambert, Orthographic, etc.
- m = Basemap(projection='merc', llcrnrlat=-80, urcrnrlat=80, llcrnrlon=-180, urcrnrlon=180) — create a Mercator map.
- m.drawcoastlines(), m.drawcountries() — draw geographic boundaries.
- m.fillcontinents(color='lightgreen') — fill land areas with color.
- Convert lat/lon to map coordinates: x, y = m(longitude, latitude)
- m.scatter(x, y, c=values, cmap='Reds') — overlay data on the map.

 Note: Basemap is being deprecated in favor of Cartopy (import cartopy.crs as ccrs) for modern geographic visualization.

5.12 Visualization with Seaborn

Seaborn is a statistical visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics with less code.

Seaborn: A Python data visualization library based on Matplotlib that provides a higher-level interface and built-in statistical plot types.

Key Seaborn Plot Types

Plot Type	Function	Use Case
Distribution	sns.histplot(), sns.kdeplot()	Show distribution of a single variable
Categorical	sns.boxplot(), sns.violinplot()	Compare distributions across categories
Categorical	sns.barplot(), sns.countplot()	Compare summary statistics or counts
Relational	sns.scatterplot(), sns.lineplot()	Show relationships between variables
Matrix	sns.heatmap()	Correlation matrices, confusion matrices
Regression	sns.regplot(), sns.lmplot()	Show data with regression line
Pair Grid	sns.pairplot(df)	All pairwise relationships in a DataFrame

5.12.1 Seaborn Advantages over Matplotlib

- Built-in themes: sns.set_theme() automatically applies a clean, professional look.
- Color palettes: sns.color_palette('husl', 8) — beautiful, accessible color schemes.
- Statistical estimation: Seaborn automatically computes and displays confidence intervals.
- DataFrame integration: Seaborn accepts Pandas DataFrames directly, using column names for labels.
- FacetGrid: sns.FacetGrid(df, col='category') — create grids of plots conditioned on a variable.

5.12.2 Heatmaps

Heatmaps are particularly useful for visualizing correlation matrices between multiple variables:

- corr = df.corr() — compute the correlation matrix.
- sns.heatmap(corr, annot=True, cmap='coolwarm', vmin=-1, vmax=1) — display with color coding.
- annot=True: Show the correlation value in each cell.
- mask parameter: Hide the upper triangle for a cleaner look.

Unit V — Review Questions

42. What is the difference between a Figure and Axes in Matplotlib?
43. How would you create a scatter plot with color-coded points representing a third variable?
44. Explain how to create a 2x2 grid of subplots in Matplotlib.
45. What is the advantage of using plt.fill_between() over error bars?
46. Compare Matplotlib and Seaborn. When would you prefer Seaborn?

47. How do you create a correlation heatmap using Seaborn?
48. What is a KDE plot? How does it differ from a histogram?

Course Summary

This learning material has covered all five units of the Foundations of Data Science course (U23CBT41). Here is a concise summary of key concepts from each unit:

Unit I — Introduction

Data Science is an interdisciplinary field combining statistics, programming, and domain expertise to extract insights from data. The data science process includes seven stages: defining goals, retrieving data, preparation, EDA, model building, presenting findings, and building applications. Key components include data mining, data warehousing, and basic statistical descriptions.

Unit II — Describing Data

Data is classified as qualitative (nominal, ordinal) or quantitative (discrete, continuous), and measured on nominal, ordinal, interval, or ratio scales. Data is described using frequency tables and graphs, measures of central tendency (mean, median, mode), measures of variability (range, variance, standard deviation, IQR), and the normal distribution with z-scores.

Unit III — Describing Relationships

Correlation measures the strength and direction of linear relationships between variables, quantified by Pearson's r (-1 to +1). Scatter plots visualize these relationships. Regression analysis models and predicts relationships: simple regression uses one predictor; multiple regression uses many. R-squared measures the proportion of variance explained, and regression towards the mean is a key statistical phenomenon.

Unit IV — Python Libraries for Data Wrangling

NumPy provides efficient N-dimensional arrays with vectorized operations, aggregations, boolean masking, fancy indexing, and structured arrays. Pandas extends this with DataFrames and Series, offering powerful tools for data selection (loc/iloc), missing data handling, hierarchical indexing, combining datasets (concat/merge/join), groupby aggregation, and pivot tables.

Unit V — Data Visualization

Matplotlib is Python's core visualization library, supporting line plots, scatter plots, histograms, contour plots, error visualization, subplots, 3D plotting, and geographic maps. Seaborn provides a high-level, aesthetically superior interface with built-in statistical plots including heatmaps, violin plots, pair plots, and KDE plots. Effective visualization requires careful attention to labels, colors, annotations, and appropriate plot selection.

Recommended References

49. McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). O'Reilly Media.

50. VanderPlas, J. (2016). Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media.
51. Wackerly, D., Mendenhall, W., & Scheaffer, R. (2008). Mathematical Statistics with Applications (7th ed.). Cengage Learning.
52. Field, A. (2018). Discovering Statistics Using IBM SPSS Statistics (5th ed.). SAGE Publications.
53. Matplotlib Documentation: https://matplotlib.org/stable/index.html
54. Pandas Documentation: https://pandas.pydata.org/docs/
55. Seaborn Documentation: https://seaborn.pydata.org/
56. NumPy Documentation: https://numpy.org/doc/